Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка програми для моделювання та розрахунку гнучких потокових технологій ремонту рухомого складу»
за освітньою програмою: «12 Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1911»

_____ /Георгій СІНЬКОВ/
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Вадим ГОРЯЧКІН/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Світлана ВОЛКОВА/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Development of a program for modeling and calculation of flexible flow technologies rolling stock repair»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1911: /Heorhii SINKOV/

Scientific Supervisor: /Vadym HORIACHKIN/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Факультет «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Сінькову Георгію Олексійовичу
1. Тема роботи: «Розробка програми для моделювання та розрахунку гнучких потокових технологій ремонту рухомого складу»
Керівник роботи: Горячкін Вадим Миколайович, завідувач кафедрою, к.т.н., доцент
затверджені наказом № 1209 ст від 07.12.2022
2. Строк подання студентом роботи: 16.05.2023р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
Вступ
Збір та аналіз вимог до веб додатку
Проектування веб додатку
Проектування прототипу веб додатку
Розробка прототипу веб додатку
Розробка веб додатку
Тестування веб додатку
Загальні висновки
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):
Презентація
Відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.23 -18.02.23
Робочий проект	Програмування та відлагодження програми.	19.02.23 - 20.05.23
	Тестування програми	20.05.23 - 27.05.23
	Розробка, узгодження і затвердження програмної документації.	27.05.23 - 12.06.23
	Подання кваліфікаційної роботи до кафедри	16.05.23
	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	26.06.23

Студент _____ Георгій СІНЬКОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Вадим ГОРЯЧКІН
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 8 розділів:
– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;
– збір вимог до програмного забезпечення – у цьому розділі описуються аналоги програми та література по даній предметній області, а також проводиться опитування зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 6 сторінок;
– зовнішнє і внутрішнє проектування – у цьому розділі проведений огляд функціональне та експлуатаційне призначення, вимоги до функціональних характеристик, вхідних і вихідних даних, опис зовнішнього інформаційних характеристик, проектування архітектури, проектування інтерфейсу користувача, проектування динаміки системи. Складається з 31 сторінок;
– розробка програми – включає в себе вибір мови програмування. Складається з 1 сторінок;
– тестування та налагодження – містить опис процесу відлагодження та тестування додатку. Складається з 1 сторінок;
– висновки. Складається з 1 сторінки;
– список літератури – включає в себе бібліографічний список використаної літератури. Складає 1 сторінки;
– додатки – містить текст програми.
Кількість таблиць: 16 штук.
Кількість рисунків: 25 штук.
Ключові слова: трансбордер, модуль, платформа, вагон.

Зміст

Вступ

Метою даної роботи є розробка програмного забезпечення, яке здійснює моделювання процесу ремонту на гнучких потокових виробництвах. Основна увага приділяється виведенню модулів на позиціях, вагонів різних типів, які надходять до вільних модулів для проходження процесу ремонту, та транспортуванню цих вагонів між модулями.

Кожен вагон потребує різного часу для виконання робіт на кожному модулі. Наприклад, просте миття чистого вагона займе менше часу, ніж зварювання складної деталі на старому вагоні. Однак необхідно уникати ситуації, коли вагони, для яких робота вже завершена, простоюють на модулі, затримуючи загальний потік робіт.

Отже, кожен модуль на кожній позиції повинен працювати з максимальною ефективністю, уникати непотрібних простоїв. Це означає, що на позиціях, де робота займає багато часу, повинно бути більше модулів, а на позиціях з швидким проходженням процесу ремонту - менше модулів, щоб уникнути незайнятості простору в будівлі та недооптимізованого використання працівників та обладнання.

Практичне застосування розробленого програмного забезпечення полягає у навчанні студентів процесу ремонту на гнучких потокових лініях та в поясненні важливості балансування кількості модулів на позиціях.

Розділ 1. Аналіз та вимоги

Процес аналізу вимог пов'язаний з установленням цілей та задоволенням потреб замовника. Під час проведення аналізу вимог необхідно визначити наступне:

- спосіб використання програмного забезпечення, тобто як замовник планує використовувати продукт;

- вхідні та вихідні дані, які потрібні для взаємодії з програмним продуктом;

- спосіб представлення інформації, який найбільше відповідає потребам замовника;

- вимоги до продуктивності, тобто як швидко та ефективно має працювати додаток;

- надійність додатка, тобто його стабільність та відсутність помилок під час роботи;

- інструменти, які будуть використовуватись для реалізації проекту.

Отже, результатом проведення збору та аналізу вимог має бути визначено, які функції має виконувати програмний продукт, як його зовнішній вигляд повинен бути спроектований та з яких інструментів він буде реалізований.

1.1 Огляд аналогів

Під час проведення дослідження був ідентифікований лише один аналоговий продукт, що використовується для моделювання гнучких потокових виробництв для ремонту рухомого складу - Conveyers Model.

Замовник надав інформацію щодо функціональності додатка. Давайте розглянемо отриману інформацію.

Conveyers Model - це перша версія додатку, яка була розроблена для моделювання гнучких потокових виробництв для ремонту рухомого складу. Проте цей додаток дозволяє лише розрахувати коефіцієнти використання та навантаження модулів.

На рисунку 1.1 зображена форма для введення конфігурації моделі, включаючи вибір методів, кількості позицій, кількості модулів на позиціях, часу переміщення до них та коефіцієнта надійності (з можливістю виходу з ладу). Також є можливість розгляду моделі з більш ніж одним трансбордером, які обслуговують різні позиції.

На рисунку 1.2 представлені результати розрахунку коефіцієнтів використання та навантаження для всіх модулів.

А на вкладці "Программа ремонта" (рисунок 1.3) є можливість переглянути номери вагонів, коли вони надійшли на ремонт, під яким номером

вони були відремонтовані, їх тип, вид та тривалість ремонту, час простою та їх розташування у цеху.

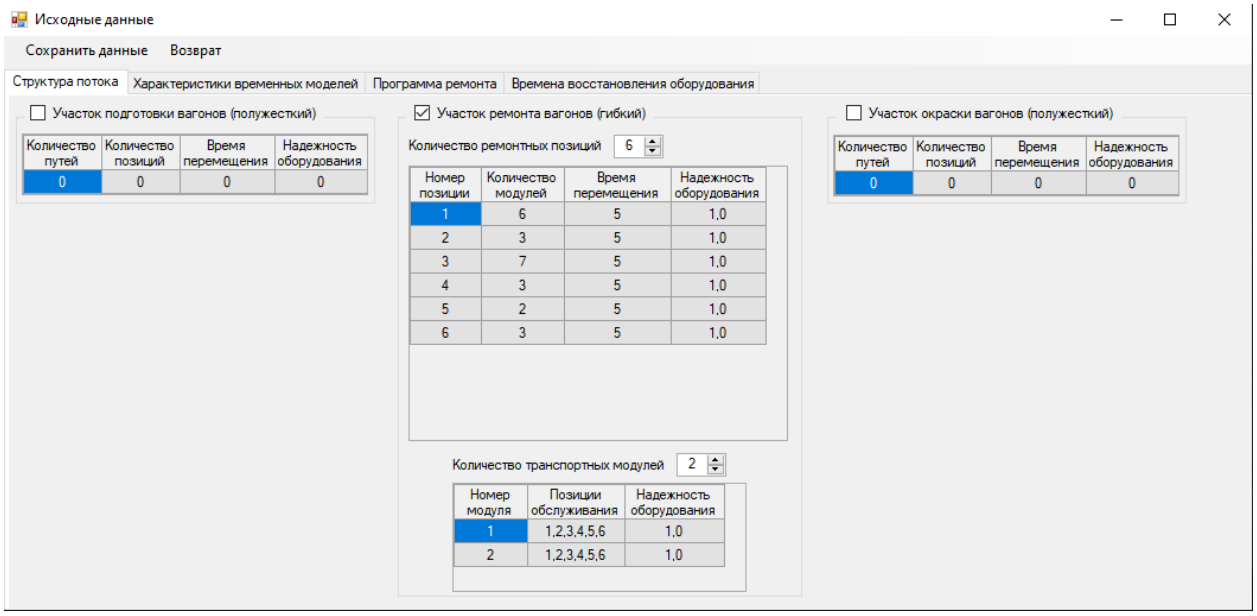

Рисунок 1.1 – Введення значень

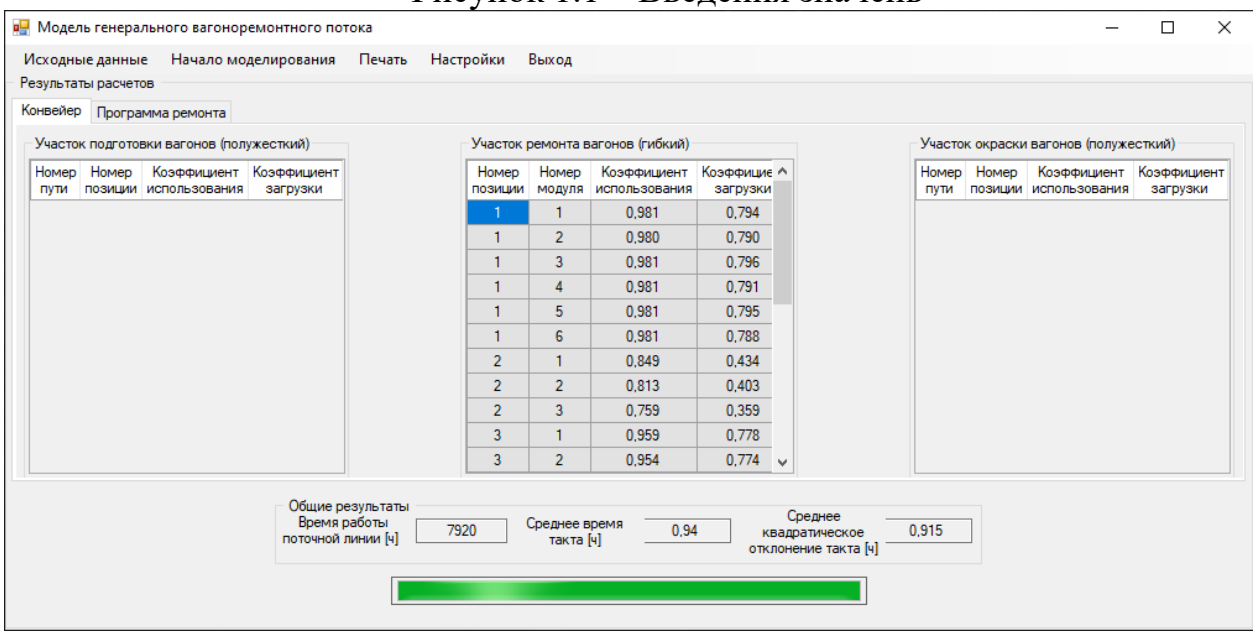

Рисунок 1.2 – Результат розрахунків для конвеєру

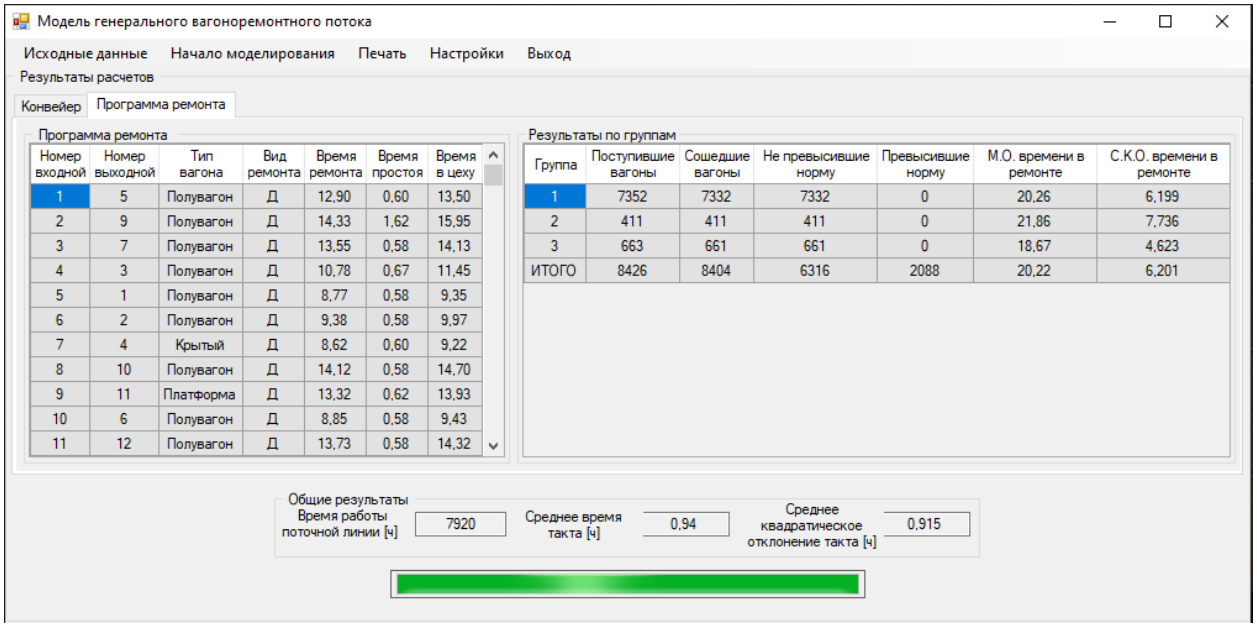

Рисунок 1.3 – Результати розрахунків програми ремонту

1.2 Огляд літератури

Далі розглянемо які бувають методи ремонту рухомих складів за матеріалами.

1.2.1 Стаціонарний метод ремонту

У стаціонарному методі ремонту весь комплекс робіт проводиться на одному робочому місці, за винятком операцій, які вимагають спеціального обладнання. Цей метод характеризується низькою продуктивністю, що призводить до значної тривалості ремонтного циклу. Можна виділити два види стаціонарного методу ремонту: стаціонарно-бригадний та стаціонарно-вузловий.

Стаціонарно-бригадний метод базується на принципі концентрації операційного процесу на одному місці. За цим методом весь цикл ремонтних робіт для вагонів та їх компонентів виконується послідовно на одній позиції однією бригадою універсальних виконавців. Після ремонту всі вузли та деталі, які були зняті з вагона, повертаються на той самий вагон.

Стаціонарно-вузловий метод ґрунтується на розподілі операцій, тобто диференціації процесу за технічними вузлами. Таким чином, уся послідовність ремонтно-збіральних операцій розділяється на вузлову та загальну збірку. Це дозволяє скоротити тривалість робіт шляхом ущільнення та одночасного виконання декількох операцій. Стаціонарно-вузловий метод отримав більше визнання порівняно зі стаціонарно-бригадним.

Проте обидва види стаціонарного методу ремонту мають недоліки, пов'язані з потребою висококваліфікованих виконавців та спеціального механізованого обладнання. При стаціонарному методі ремонту вагони непотрібно переміщати, через те, що усі роботи виконуються в одному місці.

Стосовно стаціонарного методу, його реалізація є досить простою, але

він не є продуктивним, оскільки не дозволяє використовувати весь необхідний комплекс технічного обладнання в одній позиції, починаючи від мийної машини і закінчуючи фарбувальною та сушильною камерами. Єдиним позитивним аспектом є повна незалежність від ремонту останніх вагонів, що означає, що об'єкт ремонту буде відремонтований протягом необхідного часу. Проте, в контексті єдиного залізничного шляху для подачі-видачі вагонів, згадана "незалежність" набуває залежного характеру. На рисунку 1.4 зображено структурну схему вагонної збірної ділянки, де використовується стаціонарний метод ремонту вагонів.

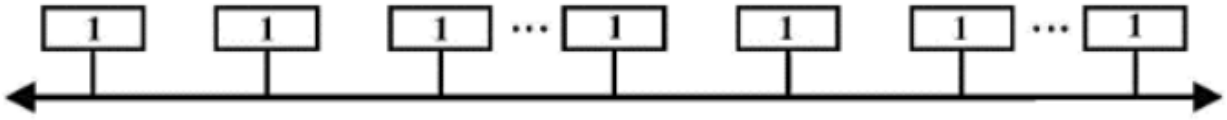

Рисунок 1.4 – Структурна схема розміщення стійл при стаціонарному методі ремонту вагонів

1.2.2 Жорсткий потік

У відміну від стаціонарної форми виробництва, потокова форма має численні недвозначні переваги, особливо явні при масовому виробництві нових продуктів.

Зазвичай, потокова форма організації виробництва включає спеціалізовані позиції, розташовані в послідовності, відповідній технічному процесу, а предмети праці постійно переміщуються між цими позиціями до повного завершення циклу. Цей метод також відомий як "поточно-предметний". Існує варіант, коли засоби праці знаходяться на одному місці, а переміщуються лише виконавці. Це відомо як "поточно-бригадний" метод організації виробництва. Навіть найпростіший потоковий метод є більш ефективним у порівнянні з не-потоковим, оскільки за допомогою спеціалізації позицій та оснащення їх необхідним технологічним обладнанням можна підвищити продуктивність праці.

На рисунку 1.5 зображена структурна схема потокової лінії з жорстким зв'язком між позиціями.

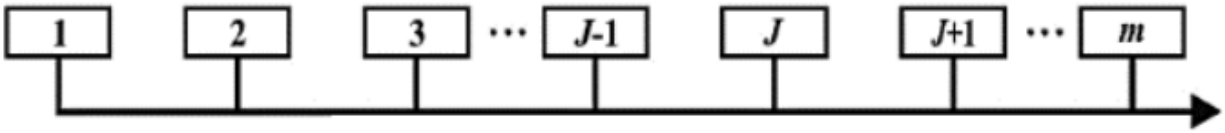

Рисунок 1.5 – Структурна схема потокової лінії з жорстким зв’язуванням між позиціями

У випадку жорсткої структури потоку, коли шлях руху предметів праці заздалегідь визначений і вони переміщуються "в одному ланцюжку" по одному і тому самому маршруту, дотримання регламенту часу виконання ремонтних операцій на кожній позиції має вирішальне значення.

З теорії надійності відомо, що найменш надійною є система, складена з послідовно з'єднаних елементів. Відмова будь-якого елемента в такому

ланцюжку призводить до відмови всієї системи в цілому. Порушення регламентованого такту на одній з позицій, фактично, становить відмову всього потоку.

1.2.3 Напівжорсткий потік

Крім широко застосовуваного "жорсткого" потоку у вагоноремонтному виробництві, існують і інші варіанти потокового виробництва. Одним з рішень, що дозволяє підвищити пропускну здатність потокової лінії без значних капітальних вкладень, є перехід від "жорсткого" потоку до "напівжорсткого" потоку. Суть цього рішення полягає у використанні окремих конвеєрів, які з'єднують лише дві сусідні позиції, замість одного вантажно-переміщувального конвеєру, що переміщає всі вагони одночасно між позиціями (рис. 1.6). Такий підхід дозволяє впроваджувати деякі елементи гнучкості, хоча не вирішує всіх проблем у повній мірі.

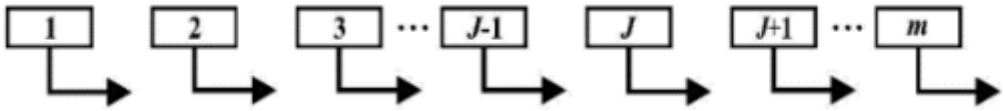

Рисунок 1.6 – Структурна схема потокової лінії з напівжорстким зв’язуванням між позиціями

1.2.4 Гнучкий варіант

Головна вагоноремонтна ділянка компактно організована в трьох паралельних будівельних прольотах, з яких два знаходяться на краю і використовуються для ремонту, а середній є транспортним.

Переміщення вагонів між позиціями здійснюється за допомогою транспортного агрегату. Вагони (модулі) розташовані поперек прольотів будівлі, а не вздовж них. Ремонтні прольоти розташовані з обох боків від транспортного прольоту. Таке розташування зумовлене, перш за все, тим, що транспортний проліт може одночасно обслуговувати ремонтні позиції, розташовані з обох його боків.

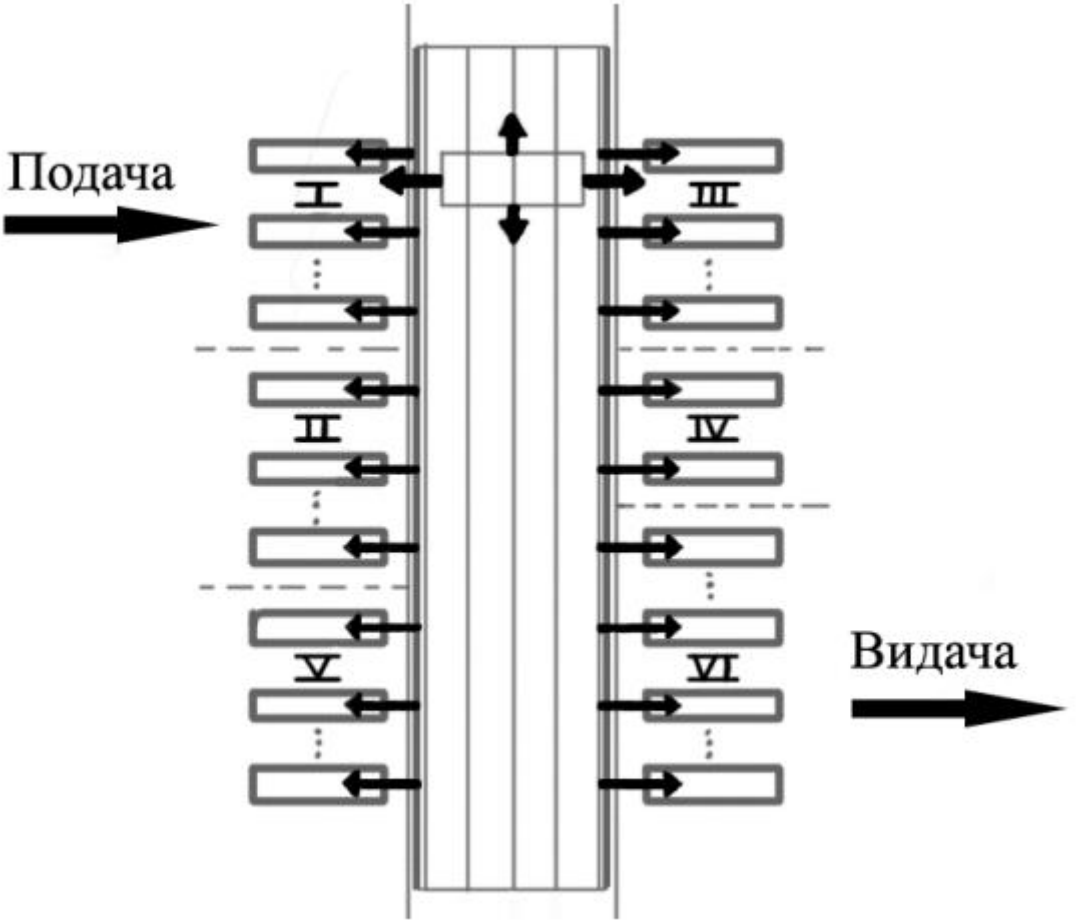

Рисунок 1.7 – Структурна схема потокової лінії з гнучкими зв’язками між позиціями

Використання окремого транспортного прольоту для переміщення вагонів між ремонтними позиціями має декілька переваг. Воно дозволяє забезпечити безпеку праці виконавців у ремонтних прольотах і уникнути їх відволікання від завдань на сусідніх модулях під час переміщення між позиціями. Крім того, така схема не впливає на роботу транспортних та вантажопідіймальних засобів, що безпосередньо обслуговують ремонтні позиції. Такий архітектурно-транспортний підхід, зі спеціалізацією ремонтних позицій, сприяє використанню потокового методу ремонту, оскільки при стаціонарному методі неможливо здійснити повний ремонтний цикл.

1.3 Опитування зацікавлених сторін

Метод опитування є психологічним вербально-комунікативним підходом, що передбачає взаємодію між інтерв'юером та респондентами з метою отримання від них відповідей на певні запитання. Опитування можна описати як комунікацію, де інтерв'юер ставить заздалегідь сформульовані запитання респонденту. У цьому процесі респондент виступає в якості замовника інформації.

1.3.1 Перелік питань

Нижче наведено список запитань, що мають на меті уточнення

поставленої задачі:

1. Який формат додатку необхідно використовувати - настільний чи веб-додаток?

2. Який метод ремонту слід застосувати для моделювання?

3. Яку інформацію слід відображати на формі додатку?

1.3.2 Відповіді респондента

Далі представлено перелік відповідей респондента на поставлені питання:

1. Настільний у вигляді додатку на комп’ютер;
2. За методом гнучких потокових виробництв;
3. Повинна бути можливість введення значень для моделювання.

1.4 Постановка задачі

Необхідно розробити програмний додаток з графічним інтерфейсом, що буде відображати перелік потрібних таблиць для вводу даних ремонту відповідно до моделі описаній у розділі 1.2.4. Також необхідна можливість задавати конфігурацію цієї моделі за допомогою файлової системи.

Висновки до розділу 1

Під час збору вимог було оглянуто й обрано необхідну схему виконання моделювання, визначено тип додатка та затверджено все, що необхідно прийняти на вхід та зобразити на екрані.

Найбільш інформативними джерелами інформації були огляд літератури та опитування зацікавлених сторін, саме за цими методами приймалося рішення, щодо того, який додаток повинен бути та який функціонал він повинен виконувати.

Розділ 2. Зовнішнє і внутрішнє проектування

2.1 Зовнішнє проектування

2.1.1 Функціональне призначення

Розроблюваний програмний продукт має на меті моделювання процесу ремонту рухомого складу на гнучких потокових виробництвах з використанням введених даних. Він надає можливість додавати та видаляти позиції, вводити кількість модулів для кожної позиції, встановлювати середній час виконання робіт на цих позиціях та нормальний допуск часу від середнього значення.

2.1.2 Експлуатаційне призначення

Завдяки цьому програмному продукту, викладачі отримають можливість ілюструвати роботу алгоритму гнучкого ремонту вагонів наочним способом перед студентами та абітурієнтами. Це сприятиме кращому розумінню процесів на гнучких потокових виробництвах з боку студентів.

Викладачі зможуть зрозуміло пояснити, що кількість модулів на позиції повинна змінюватися відповідно до часу, необхідного для виконання робіт на них.

2.1.3 Вимоги до функціональних характеристик

Повинна бути можливість задавати:
- кількість модулів на позиції;
- середній час необхідний для виконання робіт на позиції;
- нормальний відступ від середнього часу;
- кількість вагонів, що повинні пройти через усі позиції;
- швидкість відображення анімації.

Також необхідна можливість додавати та видаляти позиції, а також спосіб зупинити/почати програвання музики.

Програмний продукт повинен давати змогу вводити данні для моделюванню, а саме таблицю кількість ремонтних позицій, кількість транспортних моделей, кількість груп ремонту, обсягу ремонту, крок моделювання, тривалість зміни, кількість зміни, кількість робочих змін, фонд робочого часу, С.К.О., М.О.

2.1.4 Вхідні дані

Вхідними даними є:

1) Таблиця “Кількість ремонтних позицій”:
- positionNumber – зберігає номер позиції, значення береться з файлу(“.txt”), тип даних int.
- numberOfModules – зберігає кількість модулей, значення береться з файлу(“.txt”), тип даних int.
- movingHour – зберігає час переміщення, значення береться з файлу(“.txt”), тип даних int.
- confidenceOfPossession – зберігає надійність обладнання, значення береться з файлу(“.txt”), тип даних double.

2) Таблиця “Кількість транспортних модулей”:
- moduleNumber – зберігає позиція обслуговування, значення береться з файлу(“.txt”), тип даних int.
- servicePosition – зберігає кількість модулей, значення береться з файлу(“.txt”), тип даних int.
- confidenceOfPossession1 – зберігає надійність обладнання, значення береться з файлу(“.txt”), тип даних double.

3) Таблиця “Кількість груп ремонту”:
- groupColumn – зберігає групу, значення береться з файлу(“.txt”), тип даних int.
- wagonTypeColumn – зберігає тип вагону, значення береться з файлу(“.txt”), тип даних string.
- typeOfWorkColumn – зберігає вид робіт, значення береться з

файлу(“.txt”), тип даних string.

- normIsSimpleColumn – зберігає норма простою, значення береться з файлу(“.txt”), тип даних int.
- priority – зберігає пріоритет, значення береться з файлу(“.txt”), тип даних boolean.

4) Таблиця “Обсяг ремонту за рік”:

- groupColumn1 – зберігає групу, значення береться з файлу(“.txt”), тип даних int.
- wagonTypeColumn1 – зберігає тип вагону, значення береться з файлу(“.txt”), тип даних string.
- typeOfWorkColumn1 – зберігає вид робіт, значення береться з файлу(“.txt”), тип даних string.
- numberOfWagonsColumn – зберігає кількість вагонів, значення береться з файлу(“.txt”), тип даних int.
- interestColumn – зберігає відсоток, значення береться з файлу(“.txt”), тип даних double.

5) spinnerModelingStep – спинер вводу, що зберігає крок моделювання, тип даних int.

6) textFieldDurationOfChanges – поле вводу, що зберігає тривалість зміни, тип даних int.

7) textFieldNumberOfChanges – поле вводу, що зберігає кількість змін, тип даних int.

8) textFieldNumberOfWorkShifts – поле вводу, що зберігає кількість робочих змін, тип даних int.

9) textFieldWorkingTimeFund – поле вводу, що зберігає фонд робочого часу, тип даних int.

10) В таблицях “Тимчасові параметри ремонту об’єкту(Ділянка ремонту)”, “Час відновлення обладнання(Ділянка ремонту)”, “Час відновлення обладнання(Трансборденні візки)”стовбців група, С.К.О. та М.О. створюються динамічним, тому вони не мають назви, тип даних double.

2.1.5 Вихідні дані

Вихідними даними є:

1) Таблицю “Ділянка ремонту вагонів(гнучкий)”:
- номер позиції;
- номер модуля;
- коефіцієнт використання;
- коефіцієнт завантаження.

2) Таблицю “Програма ремонту”:
- номер вхідний;
- номер вихідний;

- тип вагону;
- вид ремонту;
- час ремонту;
- час простою;
- час в цеху.

3) Таблицю “Результаті по групам”:
- група;
- вагони надійшли;
- вагони зійшли;
- не перевищили норму;
- перевищили норму;
- М.О. часу у ремонті;
- С.К.О. часу у ремонті.

2.1.6 Опис зовнішнього інформаційного середовища

Всі дані для вводу отримуються із файлів(BogiesRecoveryData.txt, ObjData.txt, PosData.txt, ProgData.txt, RepRecoveryData.txt, TimeData.txt, TransData.txt, WorkTimeData.txt).

Користувач може перемикати всі вкладки програми за допомогою навігаційних кнопок або у списку назв цих вкладок.

Результат моделювання виводиться вигляді таблиць у окремому вікні.

Для функціонування ПЗ необхідно мати: встановлену операційну систему Windows 7 чи вище та встановлена JVM(Java Virtual Machine) на комп’ютері користувача.

2.2 Внутрішнє проектування

2.2.1 Проектування архітектури

У додатку якій я розробляв присутні дві частини розробки:

r
ackend (Логіка програми)

Нижче буде наведені діаграми класів до першої частини розробки, тобто Front
tend (Інтерфейс)

e
n
d
(Інтерфейс)

Рисунок 2.1 – Діаграма клас “MainController”

Далі наведена діаграма класів до другої частини розробки, тобто Backend (Логіка програми).

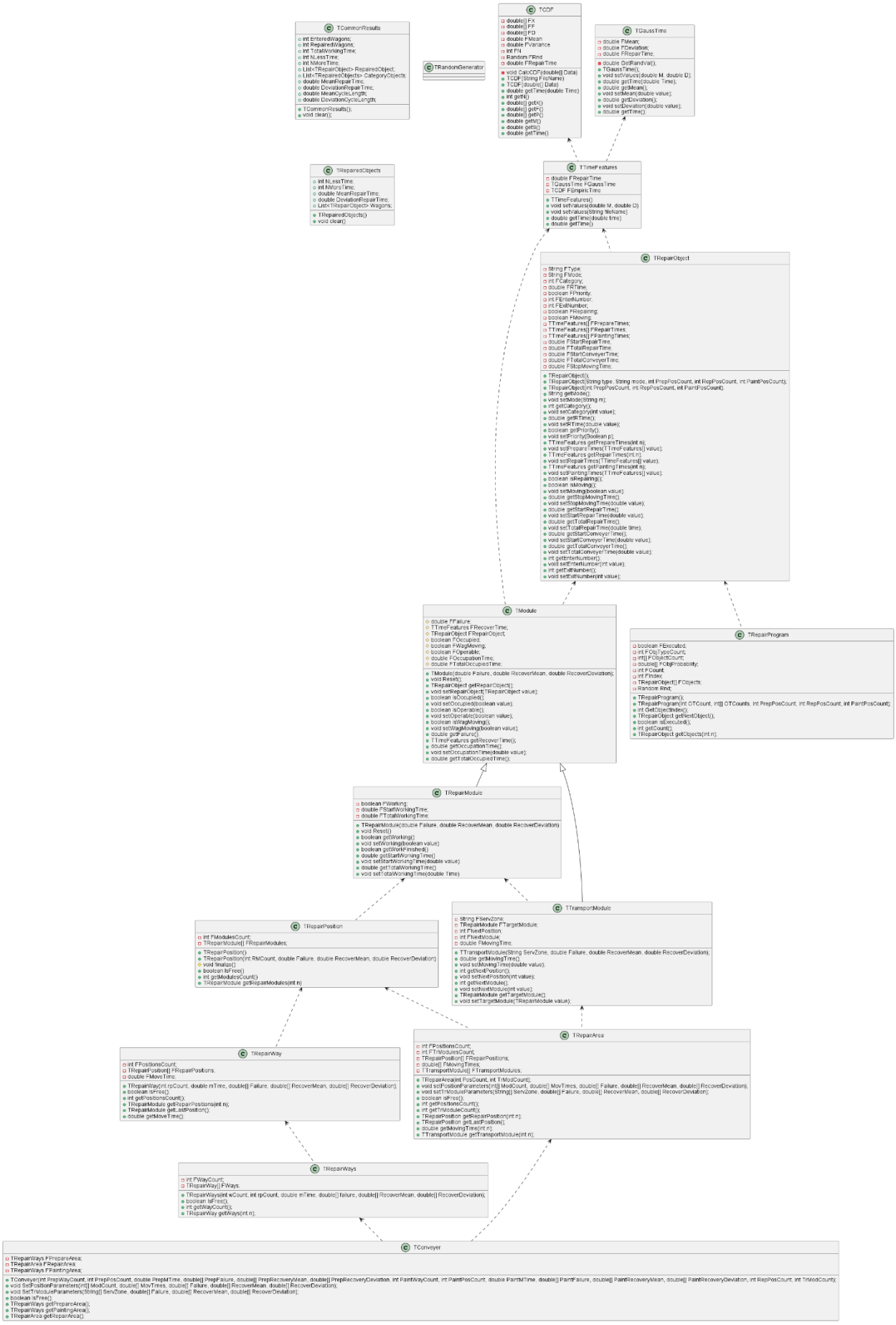

Рисунок 2.2 – Діаграма класів

У таблицях 2.1-2.16 зображені CRC картки для усіх класів.

Таблиця 2.1 - CRC картки для класу “TCDF”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати масив аргументів “FX” функції розподілу. • Зберігати масив значень “FF” функції розподілу. • Зберігати масив щільності “FD” розподілу. • Зберігати середнє значення “FMean” розподілу. • Зберігати дисперсію “FVariance” розподілу. • Зберігати довжину функції розподілу “FN”. • Зберігати генератор рівномірно розподілених випадкових чисел “FRnd”. • Зберігати час обслуговування “FRepairTime”. • Обчислювати функцію розподілу та щільність розподілу “CalcCDF()”. • Ініціалізувати об'єкт зі зчитуванням даних з файлу “TCDF(String FileName)”. • Ініціалізувати об'єкт з масивом даних “TCDF(double[] Data)”. • Отримувати час обслуговування “getTime(double Time)”. • Отримувати довжину функції	Використовує клас “Random” для генерації рівномірно розподілених випадкових чисел.

розподілу “getN()”. • Отримувати масив аргументів функції розподілу “getX()”. • Отримувати масив значень функції розподілу “getF()”. • Отримувати масив щільності розподілу “getP()”. • Отримувати середнє значення розподілу “getM()”. • Отримувати стандартне відхилення розподілу “getS()”. • Отримувати час обслуговування “getTime()”.	

Таблиця 2.2 – CRC-картки для класу “TCommonResults”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати кількість прибулих вагонів “EnteredWagons”. • Зберігати кількість відремонтованих вагонів “RepairedWagons”. • Зберігати загальний час роботи “TotalWorkingTime”. • Зберігати кількість вагонів з меншим часом обслуговування “NLessTime”. • Зберігати кількість вагонів з більшим часом обслуговування “NMoreTime”. • Зберігати список відремонтованих об'єктів “RepairedObject”. • Зберігати список об'єктів, розділених за категоріями “CategoryObjects”.	Використовує клас “TRepairObject” для зберігання відремонтованих об'єктів. Використовує клас “TRepairedObjects” для зберігання об'єктів, розділених за категоріями.

• Зберігати середній час ремонту “MeanRepairTime”. • Зберігати відхилення часу ремонту “DeviationRepairTime”. • Зберігати середню тривалість циклу “MeanCycleLength”. • Зберігати відхилення тривалості циклу “DeviationCycleLength”. • Очищувати дані “clear()”.	

Таблиця 2.3 – CRC-картки для класу “TConveyer”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати об'єкт участка підготовки “FPrepareArea”. • Зберігати об'єкт участка ремонту “FRepairArea”. • Зберігати об'єкт участка фарбування “FPaintingArea”. • Створювати об'єкти участків підготовки, ремонту та фарбування з заданими параметрами “TConveyer(int ”. • Встановлювати параметри р • Встановлювати параметри транспортних модулів “SetTrModuleParameters(String[] • Перевіряти, чи є конвеєр вільним “IsFree()”. • Повертати об'єкт участка підготовки “getPrepareArea()”. • Повертати об'єкт участка	Використовує клас “TRepairWays” для зберігання об'єктів участків підготовки та фарбування. Використовує клас “TRepairArea” для зберігання об'єкту участка ремонту.

фарбування “getPaintingArea()”. • Повертати об'єкт участка ремонту “getRepairArea()”.	

Таблиця 2.4 – CRC-картки для класу “TGaussTime”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати середнє значення часу обслуговування “FMean”. • Зберігати середнє квадратичне відхилення часу обслуговування “FDeviation”. • Зберігати поточний час обслуговування “FRepairTime”. • Генерувати випадкове число “GetRandVal()”. • Встановлювати значення параметрів “setValues(double ”. • Розраховувати момент закінчення обслуговування “getTime(double Time)”. • Повертати середнє значення часу обслуговування “getMean()”. • Встановлювати середнє значення часу обслуговування “setMean(double value)”. • Повертати середнє квадратичне відхилення часу обслуговування “getDeviation()”. • Встановлювати середнє квадратичне відхилення часу обслуговування “setDeviation(double value)”. • Повертати поточний час	Використовує клас “Math” для обчислення випадкових чисел та математичних операцій.

обслуговування “getTime()”.	

Таблиця 2.5 – CRC-картки для класу “TModule”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати ймовірність відмови модуля “FFailure”. • Зберігати тимчасові параметри відновлення працездатності модуля “FRecoverTime”. • Зберігати вагон, що знаходиться зараз у модулі “FRepairObject”. • Зберігати прапор зайнятості модуля “FOccupied”. • Зберігати прапор руху вагона до модуля “FWagMoving”. • Зберігати прапор працездатності модуля “FOperable”. • Зберігати момент переходу модуля у зайнятий стан “FOccupationTime”. • Зберігати загальний час зайнятості модуля “FTotalOccupiedTime”. • Скидати параметри модуля “Reset()”. • Повертати вагон, що знаходиться зараз у модулі “getRepairObject()”. • Встановлювати вагон, що иться зараз у модулі “setRepairObject(TRepairObject ”. • Повертати прапор зайнятості модуля “isOccupied()”.	Використовує клас “TRepairObject” для зберігання вагона, що знаходиться у модулі. Використовує клас “TTimeFeatures” для зберігання тимчасових параметрів відновлення працездатності.

• Встановлювати прапор зайнятості модуля “setOccupied(boolean value)”. • Повертати прапор працездатності модуля “isOperable()”. • Встановлювати прапор працездатності модуля “setOperable(boolean value)”. • Повертати прапор руху вагона до модуля “isWagMoving()”. • Встановлювати прапор руху вагона до модуля “setWagMoving(boolean ”. • Повертати ймовірність несправності модуля “getFailure()”. • Повертати тимчасові параметри відновлення працездатності модуля “getRecoverTime()”. • Повертати момент переходу модуля у зайнятий стан “getOccupationTime()”. • Встановлювати момент переходу модуля у зайнятий стан “setOccupationTime(double value)”. • Повертати загальний час зайнятості модуля “ getTotalOccupiedTime()”.	

Таблиця 2.6 – CRC-картки для класу “TRandomGenerator”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати випадковий	Використовує клас “Random” для

генератор “FRndX”. • Зберігати випадковий генератор “FRndY”. • Зберігати середнє значення “FMean”. • Зберігати стандартне відхилення “FDeviation”. • Зберігати випадкові значення в межах діапазону 0-1 “”FAuxRndVal. • Зберігати випадкові значення з середнім значенням FMean та стандартним відхиленням “”FDeviation FRndVal. • Зберігати прапори для типів випадкових значень: нормальний “”(FCNor), Релея “”(FCRel), експонентний “”(FCExp). • Генерувати випадкові числа “”Values().	створення випадкових генераторів “FRndX” та “FRndY”. Використовує клас “TRandomValues” для зберігання випадкових значень “ FAuxRndVal ” та “ FRndVal ”.

Таблиця 2.7 – CRC-картки для класу “TRepairArea”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати кількість ремонтних позицій “”FPositionsCount. • Зберігати кількість транспортних модулів “”FTrModulesCount. • Зберігати масив ремонтних позицій “”FRepairPositions. • Зберігати масив часів переміщення вагона між позиціями “”FMovingTimes. • Зберігати масив транспортних модулів “”FTransportModules.	Використовує клас “”TRepairPosition для створення та зберігання ремонтних позицій. Використовує клас “”TTransportModule для створення та зберігання транспортних модулів.

• Встановлювати параметри монтних позицій “”setPositionParameters(). • Встановлювати параметри портних модулів “”setTrModuleParameters(). • Перевіряти, чи всі позиції та транспортні модулі вільні “”isFree(). • Отримувати кількість ремонтних позицій “”getPositionsCount(). • Отримувати кількість транспортних модулів “”getTrModuleCount(). • Отримувати ремонтну позицію за індексом “”getRepairPosition(). • Отримувати останню ремонтну позицію “”getLastPosition(). • Отримувати час переміщення за індексом “”getMovingTime(). • Отримувати транспортний модуль за індексом “”getTransportModule().	

Таблиця 2.8 – CRC-картки для класу “TRepairedObjects”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати кількість відремонтованих вагонів з меншим часом ремонту “”NLessTime. • Зберігати кількість відремонтованих вагонів з більшим часом ремонту “”NMoreTime.	Використовує клас “”TRepairObject для зберігання відремонтованих вагонів у списку “”Wagons.

• Зберігати стандартне илення часу ремонту “”DeviationRepairTime. • Зберігати список відремонтованих вагонів “”Wagons. • Очищувати дані про відремонтовані вагони “”clear().	

Таблиця 2.9 – CRC-картки для класу “TRepairModule”

Базовий клас	Похідні класи
“”TModule	Відсутні
Обов’язки	Зв’язки
• Зберігати інформацію про стан роботи модуля: “”FWorking. • Зберігати час початку роботи модуля: “”FStartWorkingTime. • Зберігати загальний час роботи модуля: “”FTotalWorkingTime. • Скидати параметри модуля: “”Reset(). • П • Встановлювати стан роботи модуля: “”setWorking(). • Перевіряти, чи завершена робота модуля: “”getWorkFinished(). • Повертати час початку роботи модуля: “”getStartWorkingTime(). • Встановлювати час початку роботи модуля: “”setStartWorkingTime(). • Повертати загальний час роботи модуля:	Відсутні

“”getTotalWorkingTime(). • Встановлювати загальний час роботи модуля: “”setTotalWorkingTime().	

Таблиця 2.10 – CRC-картки для класу “TRepairObject”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати інформацію про тип об'єкта: “FType”. • Зберігати інформацію про режим об'єкта: “FMode”. • Зберігати категорію об'єкта: “FCategory”. • Зберігати час ремонту об'єкта: “FRTime”. • Зберігати пріоритет об'єкта: “FPriority”. • Зберігати номер входу об'єкта: “FEnterNumber”. • Зберігати номер виходу об'єкта: “FExitNumber”. • Зберігати стан ремонту об'єкта: “FRepairing”. • Зберігати стан переміщення об'єкта: “FMoving”. • Зберігати характеристики часу підготовки: “FPrepareTimes”. • Зберігати характеристики часу ремонту: “FRepairTimes”. • Зберігати характеристики часу фарбування: “FPaintingTimes”. • Зберігати час початку ремонту об'єкта: “FStartRepairTime”. • Зберігати загальний час ремонту об'єкта:	Клас “”TTimeFeatures для збереження характеристик часу підготовки, ремонту та фарбування.

“FTotalRepairTime”. • Зберігати час початку переміщення об'єкта: “FStartConveyerTime”. • Зберігати загальний час переміщення об'єкта: “FTotalConveyerTime”. • Зберігати час зупинки переміщення об'єкта: “FStopMovingTime”. • Отримувати/встановлювати режим об'єкта: “getMode()”, “setMode()”. • Отримувати/встановлювати категорію об'єкта: “getCategory()”, “setCategory()”. • Отримувати/встановлювати час ремонту об'єкта: “getRTime()”, “setRTime()”. • Отримувати/встановлювати пріоритет об'єкта: “getPriority()”,“setPriority()”. • Отримувати/встановлювати характеристики часу підготовки: “getPrepareTimes()”, “setPrepareTimes()”. • Отримувати/встановлювати характеристики часу ремонту: “getRepairTimes()”, “setRepairTimes()”. • Отримувати/встановлювати характеристики часу фарбування: “getPaintingTimes()”, “setPaintingTimes()”. • Перевіряти, чи об'єкт знаходиться у стані ремонту:	

“isRepairing()”. • Перевіряти, чи об'єкт знаходиться у стані переміщення: “isMoving()”. • Встановлювати стан переміщення об'єкта: “setMoving()”. • Отримувати/встановлювати час зупинки переміщення об'єкта: “getStopMovingTime()”, “setStopMovingTime()”. • Отримувати/встановлювати час початку ремонту об'єкта: “getStartRepairTime()”, “setStartRepairTime()”. • Отримувати/встановлювати загальний час ремонту об'єкта: “getTotalRepairTime()”, “setTotalRepairTime()”. • Отримувати/встановлювати час початку переміщення об'єкта: “getStartConveyerTime()”, “setStartConveyerTime()”. • Отримувати/встановлювати загальний час переміщення об'єкта: “getTotalConveyerTime()”, “setTotalConveyerTime()”. • Отримувати/встановлювати номер входу об'єкта: “getEnterNumber()”, “setEnterNumber()”. • Отримувати/встановлювати номер виходу об'єкта: “getExitNumber()”, “setExitNumber()”. • Отримувати/встановлювати тип об'єкта: “getFType()”,	

“setFType()”.	

Таблиця 2.11 – CRC-картки для класу “TRepairPosition”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати кількість модулів у ремонтній позиції: “”FModulesCount. • Зберігати масив модулів ремонтної позиції: “”FRepairModules. • Створювати ремонтні модулі та встановлювати їх параметри. • Скидати параметри всіх модулів ремонтної позиції. • Перевіряти, чи ремонтна позиція вільна: “”IsFree(). • Отримувати кількість модулів у ремонтній позиції: “”getModulesCount(). • Отримувати модуль ремонтної позиції за індексом: “”getRepairModules().	Клас “”TRepairModule для збереження параметрів модулів ремонтної позиції.

Таблиця 2.12 – CRC-картки для класу “TRepairProgram”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати стан виконання програми ремонту: “”FExecuted. • Зберігати кількість типів об'єктів ремонту: “”FObjTypeCount. • Зберігати масив кількостей об'єктів ремонту: “”FObjectCount.	Клас “”TRepairObject для збереження параметрів об'єктів ремонту. Клас “”TTimeFeatures для збереження часових характеристик ремонту. Клас “”Random для генерації випадкових номерів об'єктів ремонту.

• Зберігати масив частот появи об'єктів різних типів в обсязі ремонту: “”FObjProbability. • Зберігати обсяг ремонту (загальну кількість об'єктів ремонту): “”FCount. • Зберігати індекс вагона в початку черги (наступного об'єкта, що потрапить на конвеєр): “”FIndex. • Зберігати масив об'єктів ремонту: “”FObjects. • Зберігати генератор випадкових номерів об'єктів: “”Rnd. • Отримувати індекс об'єкта ремонту: “”GetObjectIndex(). • Отримувати наступний об'єкт ремонту: “”getNextObject(). • Перевіряти, чи програма ремонту виконана: “”isExecuted(). • Отримувати кількість об'єктів ремонту: “”getCount(). • Отримувати об'єкт ремонту за індексом: “”getObjects().	

Таблиця 2.13 – CRC-картки для класу “TRepairWay”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати кількість ремонтних позицій: “”FPositionsCount. • Зберігати масив ремонтних позицій: “”FRepairPositions. • Зберігати час переміщення між позиціями: “”FMoveTime. • Перевіряти, чи всі позиції	Клас “”TRepairPosition для збереження ремонтних модулів позиції.

вільні: “”IsFree(). • Отримувати кількість ремонтних позицій: “”getPositionsCount(). • Отримувати ремонтну позицію за індексом: “”getRepairPositions(). • Отримувати останню ремонтну позицію: “”getLastPosition(). • Отримувати час переміщення між позиціями: “”getMoveTime().	

Таблиця 2.14 – CRC-картки для класу “TRepairWays”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати кількість ремонтних шляхів: “”FWayCount. • Зберігати масив ремонтних шляхів: “”FWays. • Перевіряти, чи всі шляхи вільні: “”IsFree(). • Отримувати кількість ремонтних шляхів: “”getWayCount(). • Отримувати ремонтний шлях за індексом: “”getWays().	Клас “”TRepairWay для збереження ремонтних позицій шляху.

Таблиця 2.15 – CRC-картки для класу “TTimeFeatures”

Базовий клас	Похідні класи
Відсутні	Відсутні
Обов’язки	Зв’язки
• Зберігати значення параметрів обслуговування: “FRepairTime”, “FGaussTime”, “FEmpiricTime”. • Встановлювати значення	Клас “”TGaussTime для обробки нормального розподілу часу. Клас TCDF для обробки емпіричного розподілу часу.

параметрів для нормального розподілу часу: “setValues(double M, double • Встановлювати значення параметрів для емпіричного розподілу часу: “setValues(String fileName)”. • Розраховувати момент закінчення обслуговування: “getTime(double time)”. • Отримувати поточний час обслуговування: “getTime()”.	

Таблиця 2.16 – CRC-картки для класу “TTransportModule”

Базовий клас	Похідні класи
“TModule”	Відсутні
Обов’язки	Зв’язки
• Зберігати інформацію про зону обслуговування “FServZone”. • Зберігати посилання на модуль, до якого їде транспортний модуль “FTargetModule”. • Зберігати номер наступної позиції “FNextPosition”. • Зберігати номер вільного модуля наступної позиції “FNextModule”. • Зберігати час переміщення вагона до вільного модуля наступної позиції “FMovingTime”. • Наслідувати властивості та методи від класу “TModule”. • Повертати час переміщення до вільного модуля наступної позиції “getMovingTime()”. • Встановлювати час переміщення до вільного модуля наступної позиції “setMovingTime()”.	“TRepairModule”

• Повертати номер наступної позиції “getNextPosition()”. • Встановлювати номер наступної позиції “setNextPosition()”. • Повертати номер вільного модуля наступної позиції “getNextModule()”. • Встановлювати номер вільного модуля наступної позиції “setNextModule()”. • Повертати модуль призначення “getTargetModule()”. • Встановлювати модуль призначення “setTargetModule()”.	

2.2.2 Проектування інтерфейсу користувача

На формі необхідно відобразити:
- таблицю для введення конфігурації моделі;
- таблицю для виведення результату моделювання;
- кнопку для переходу на наступну сторінку;
- кнопку для переходу на попередню сторінку.

Було визначено взаємне розташування даних та елементів інтерфейсу на екрані, яке можна побачити на рисунку 2.3-2.8. Основна увага на формі була зосереджена на таблицях, де будуть зберігатися дані для заповнення моделі.

Кольорова палітра додатку:
- головний фон: #3C3F41;
- границя фону: #323232;
- фон кнопка вперед/моделювання: #365880;
- контур кнопка вперед/моделювання: #4C708C;
- фон кнопка назад/відміна: #4C5052;
- контур кнопка назад/відміна: #5E6060;
- фон для кнопка активної: #4B6EAF;
- основний текст: #BBBBBB;
- не активний текст: #808080.

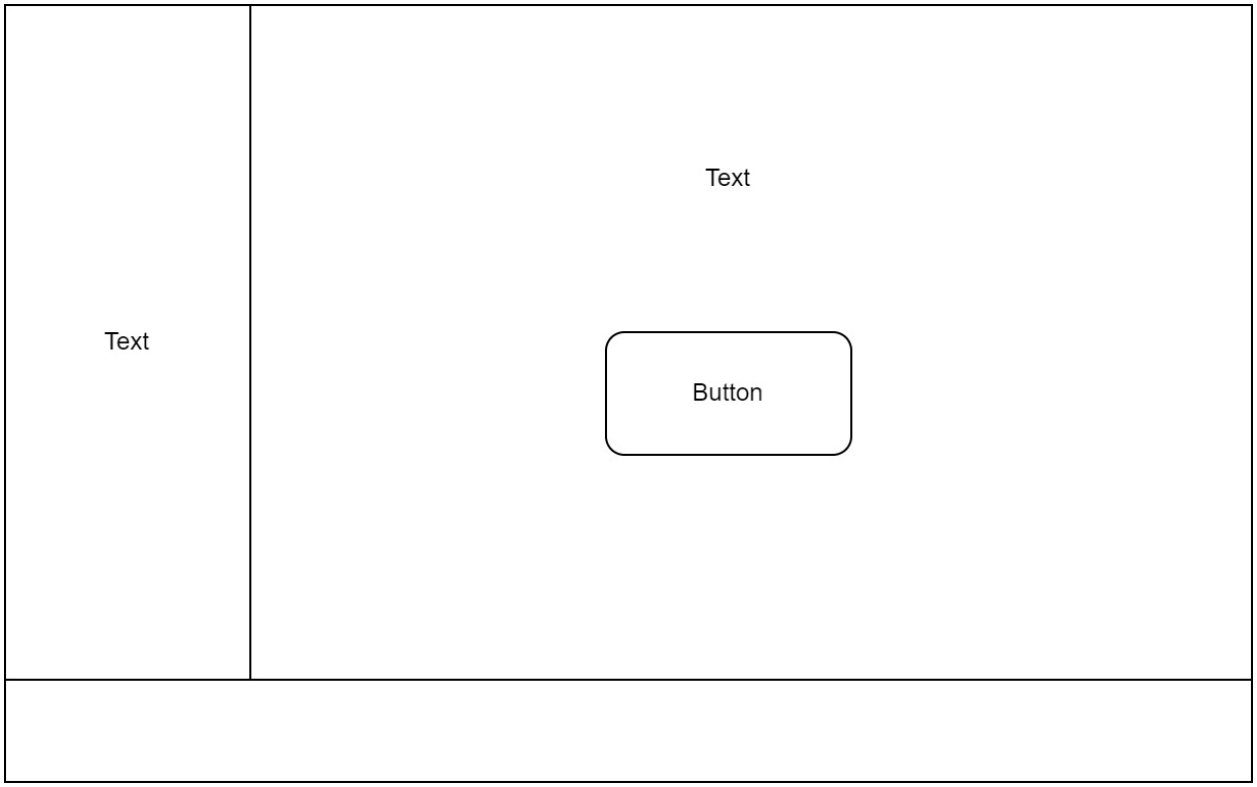

Рисунок 2.3 – Ескіз форми “Вступ”

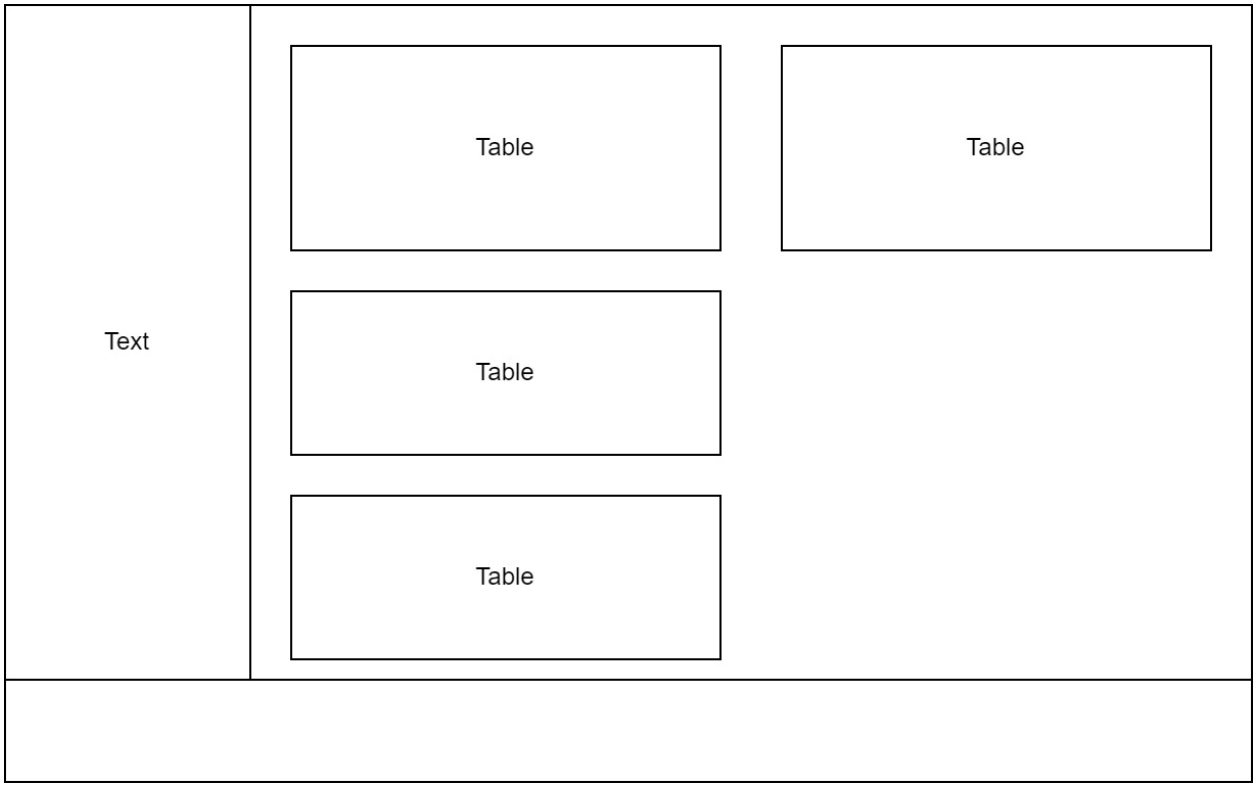

Рисунок 2.4 – Ескіз форми “Структура потоку”

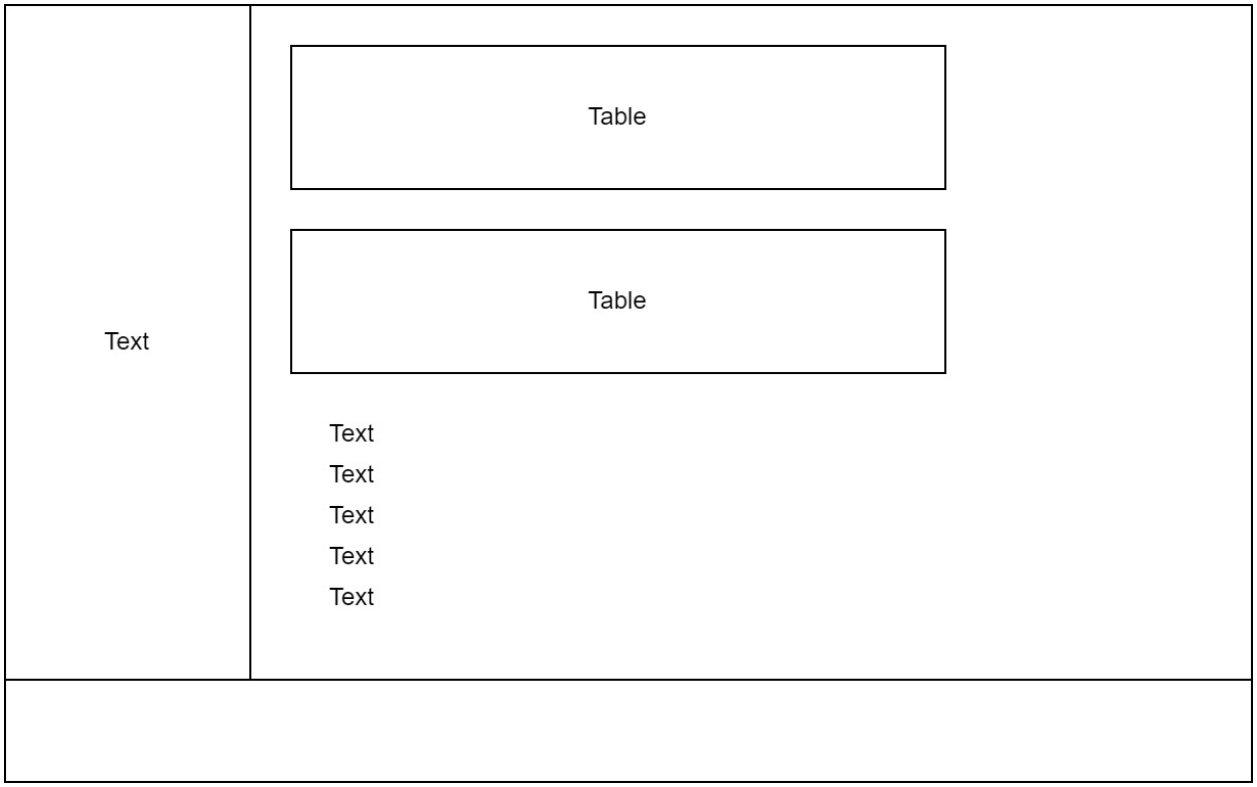

Рисунок 2.5 – Ескіз форми “Програма ремонту”

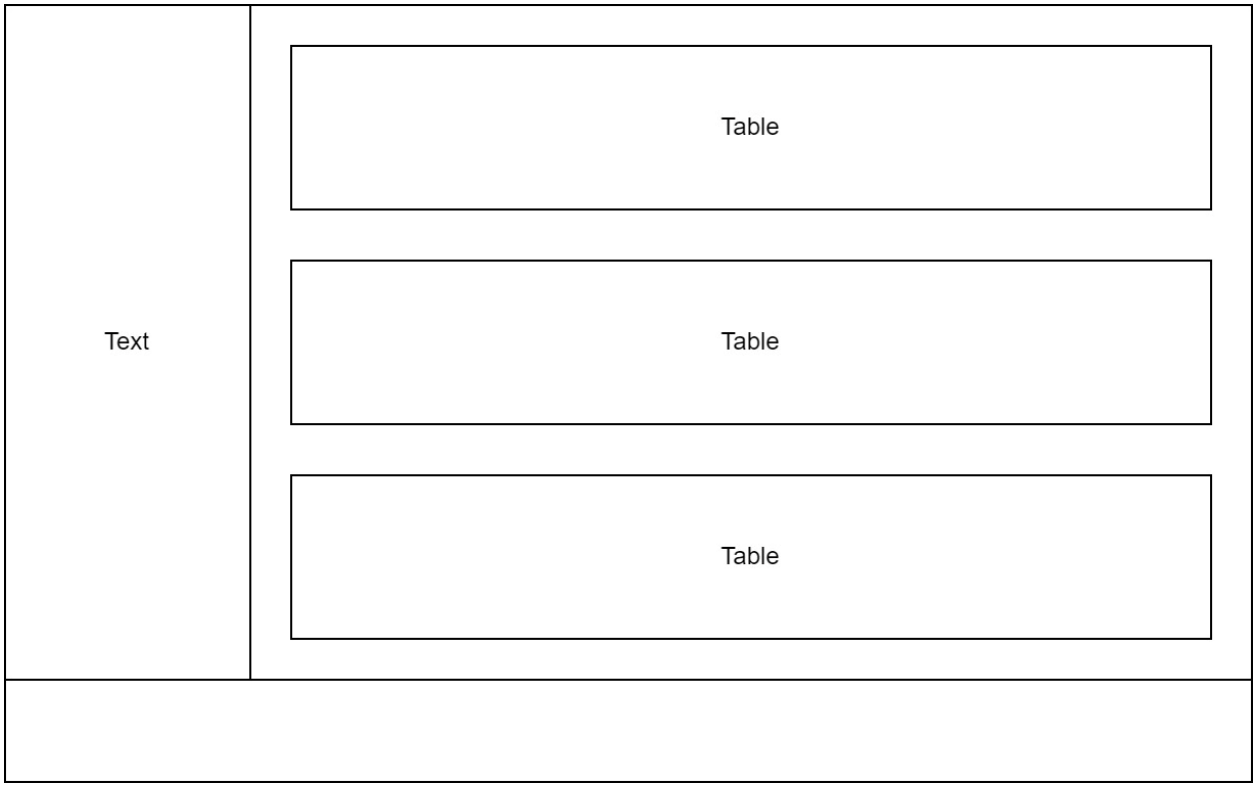

Рисунок 2.5 – Ескіз форми “Характеристика тимчасових моделей”

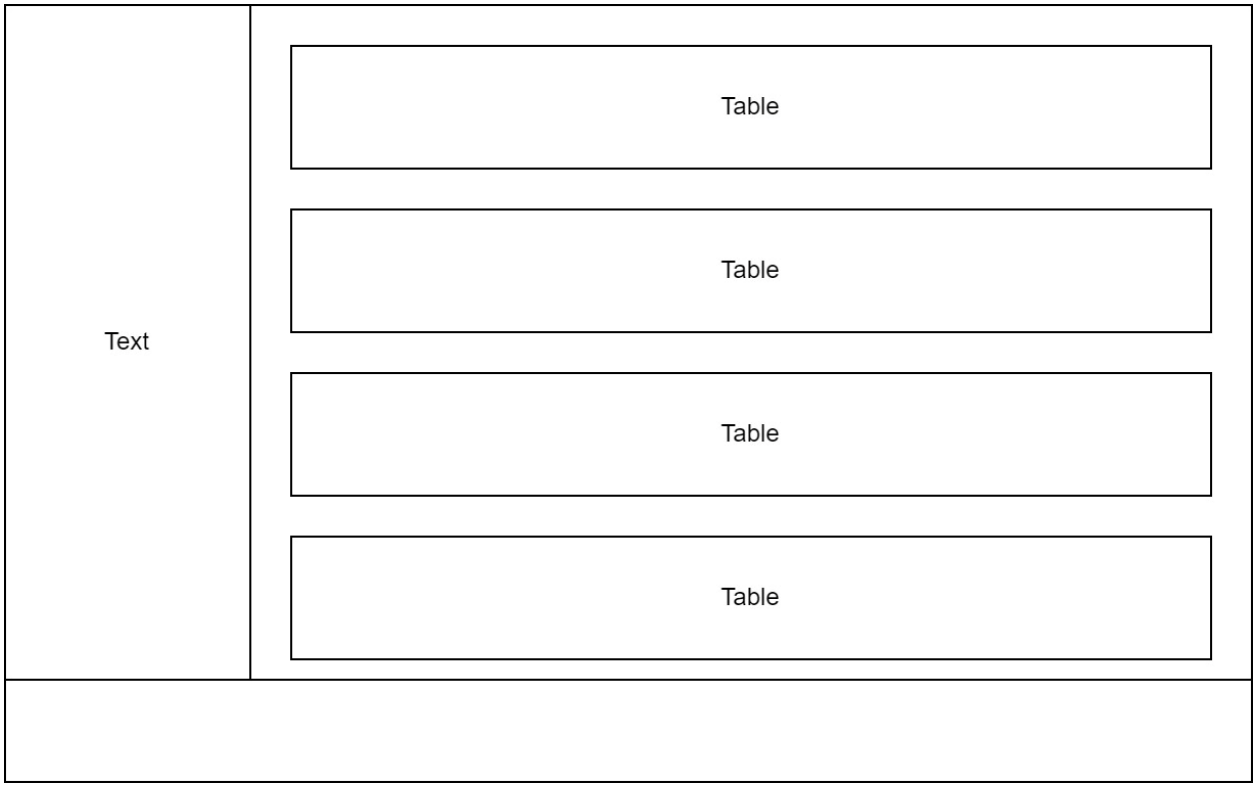

Рисунок 2.6 – Ескіз форми “Часи відновлення обладнання”

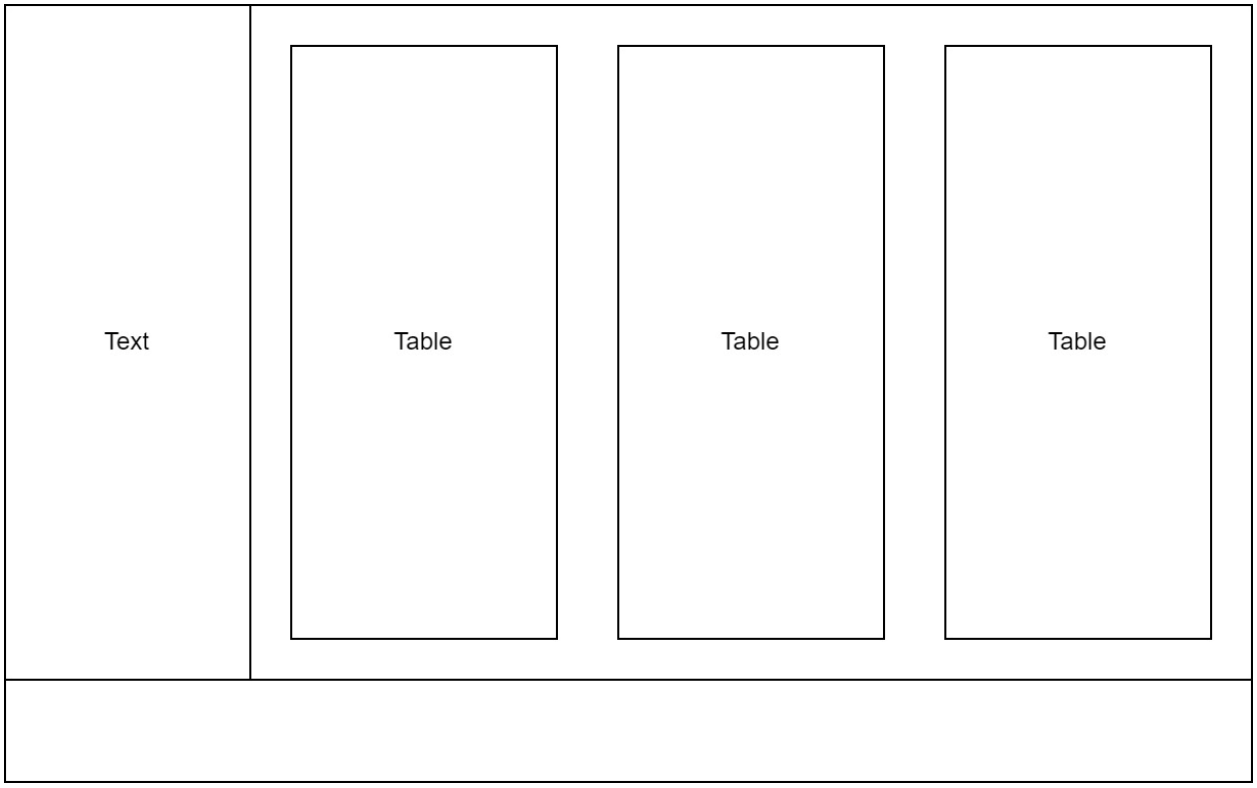

Рисунок 2.7 – Ескіз форми “Контейнер”

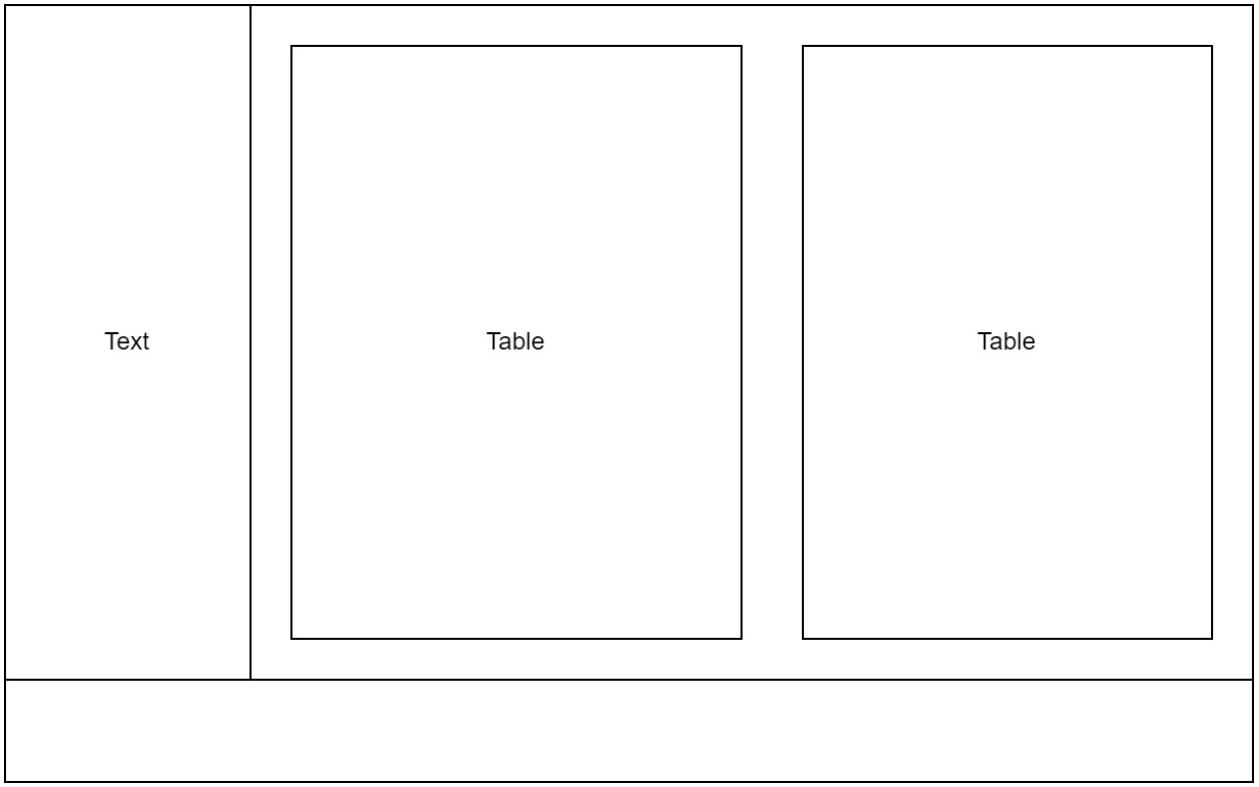

Рисунок 2.8 – Ескіз форми “Програма ремонту”

Після розробки всіх ескізів я приступив до розробки інтерфейсу. За основу вибору кольорової палітри та візуального розміщення елементів брав натхнення з дизайну початкового вікна IntelliJ IDEA, де відбуватися створення проектів та їх налаштування. Зробив список вибору вкладок так само як і в початковому вікні IntelliJ IDEA, це можна побачити на рисунку 2.9.

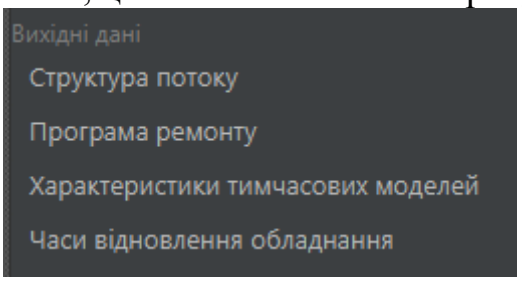

Рисунок 2.9 – Бокова панель вибору вкладок

Після почав розмішати всі необхідні таблиці у вкладках “Структура потоку”, “Програма ремонту”, “Характеристика тимчасових моделей”, “Часи відновлення обладнання” та інші елементи для вводу параметрів для моделювання, які можна побачити на рисунках(2.10-2.16).

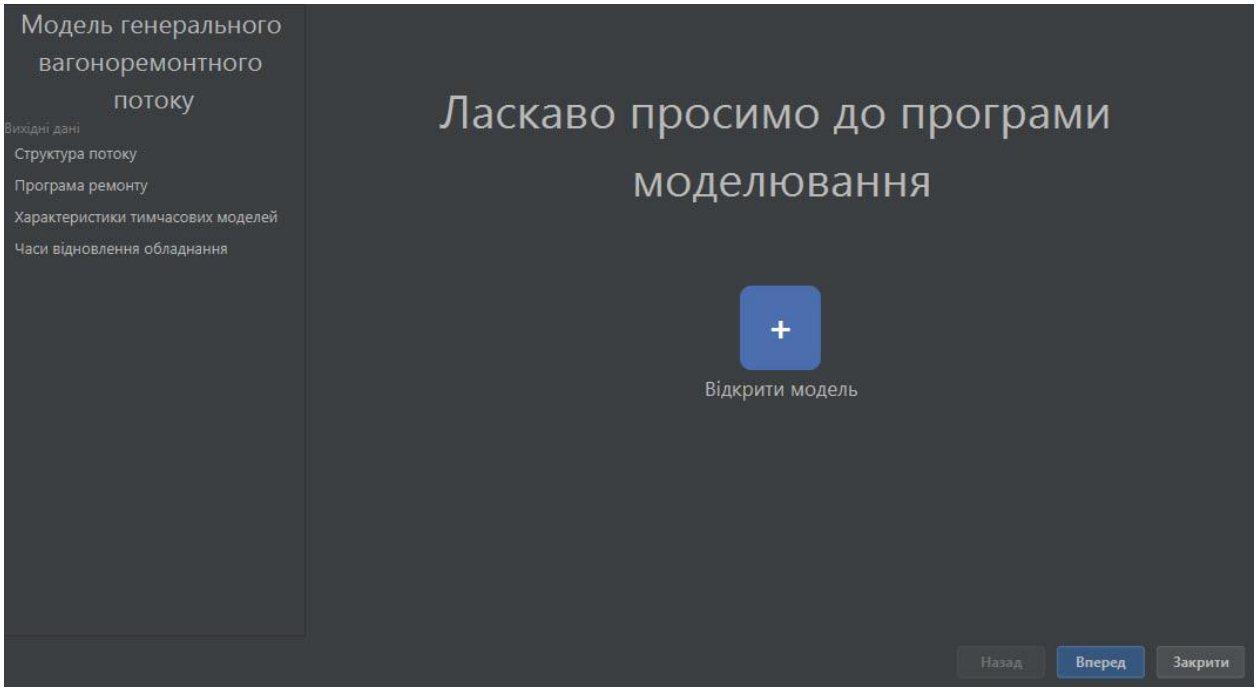

Рисунок 2.10 – Початкова сторінка

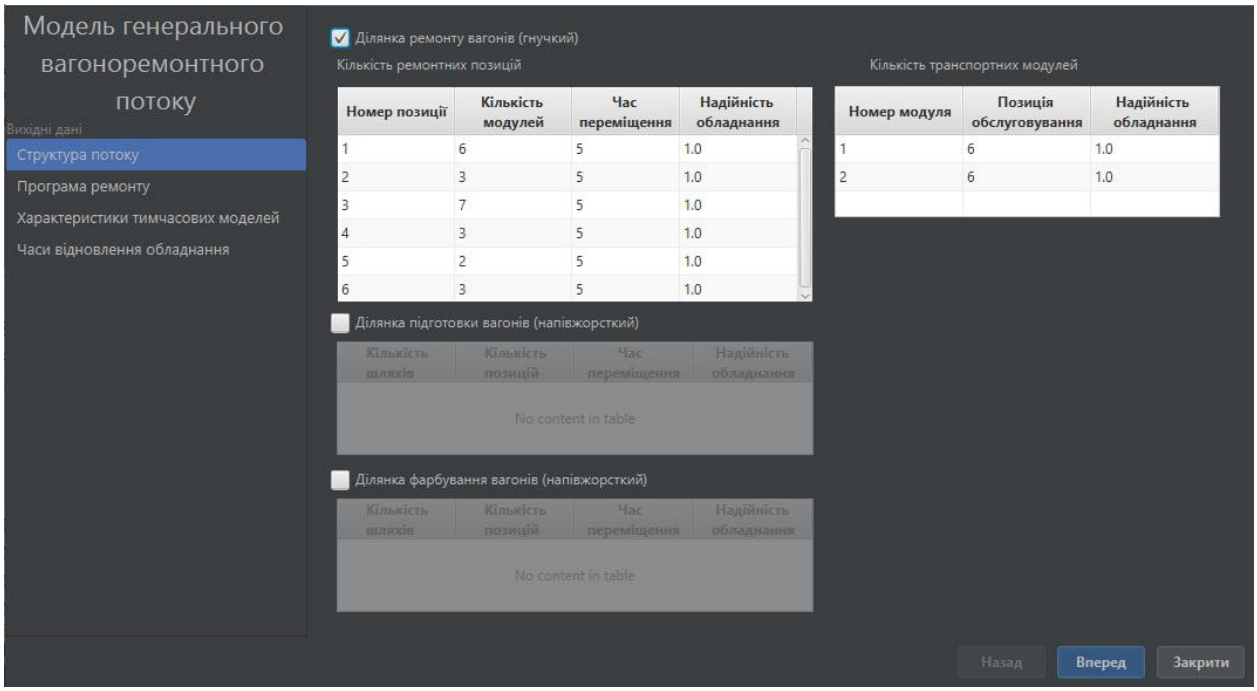

Рисунок 2.11 – Сторінка “Структура потоку”

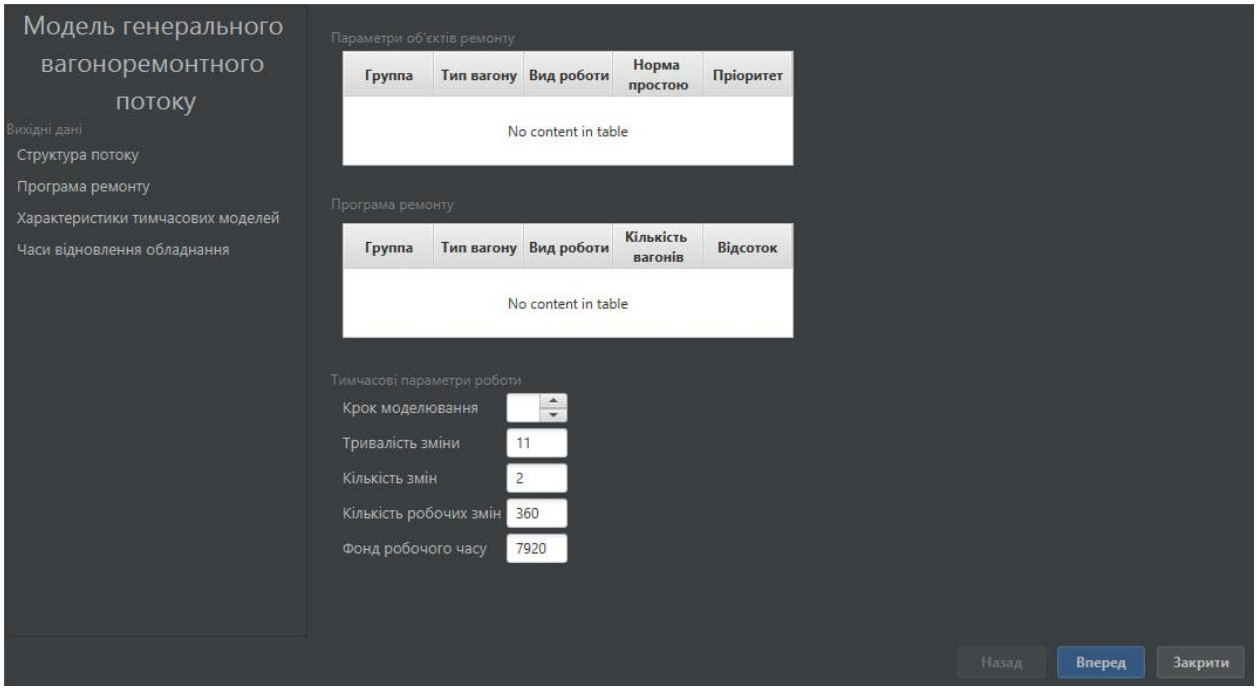

Рисунок 2.12 – Сторінка “Програма ремонту”

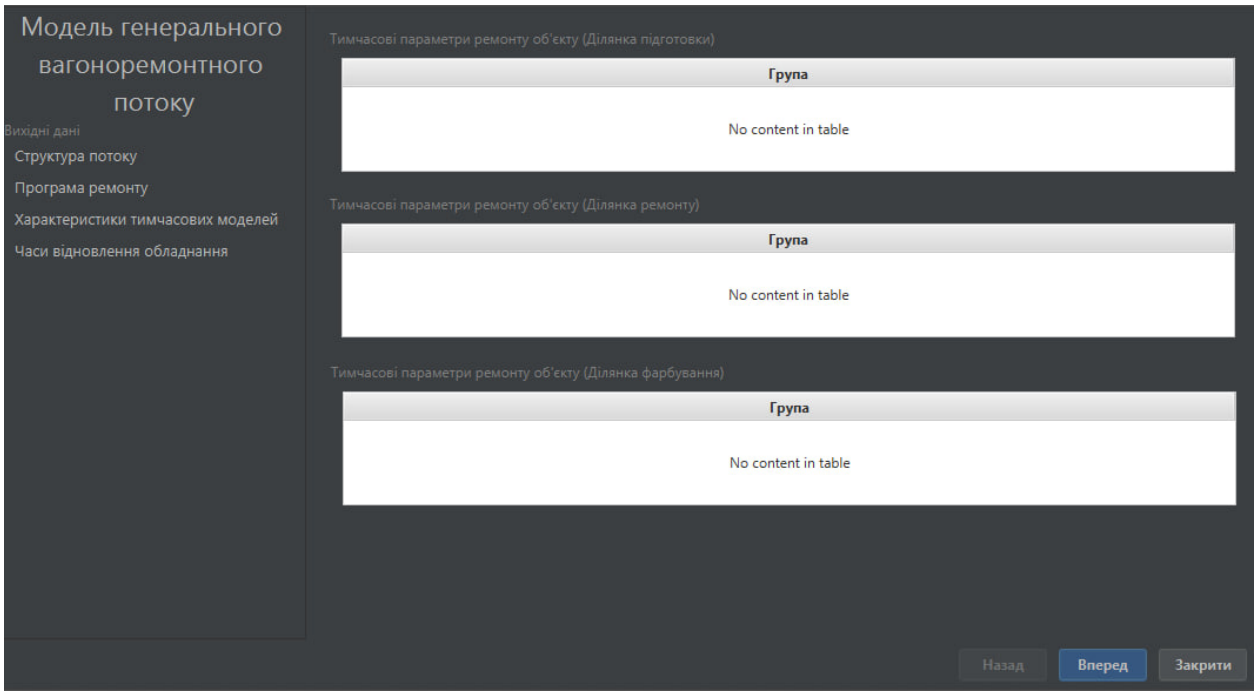

Рисунок 2.13 – Сторінка “Характеристика тимчасових моделей”

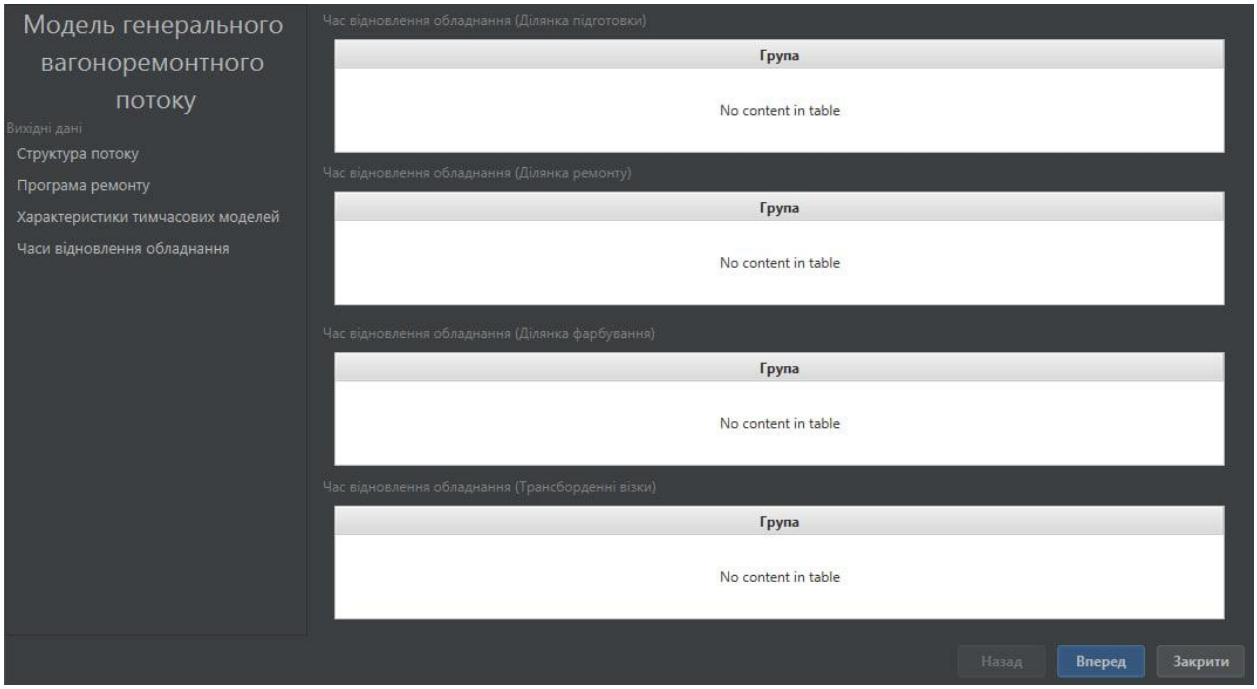

Рисунок 2.14 – Сторінка “Часи відновлення обладнання”

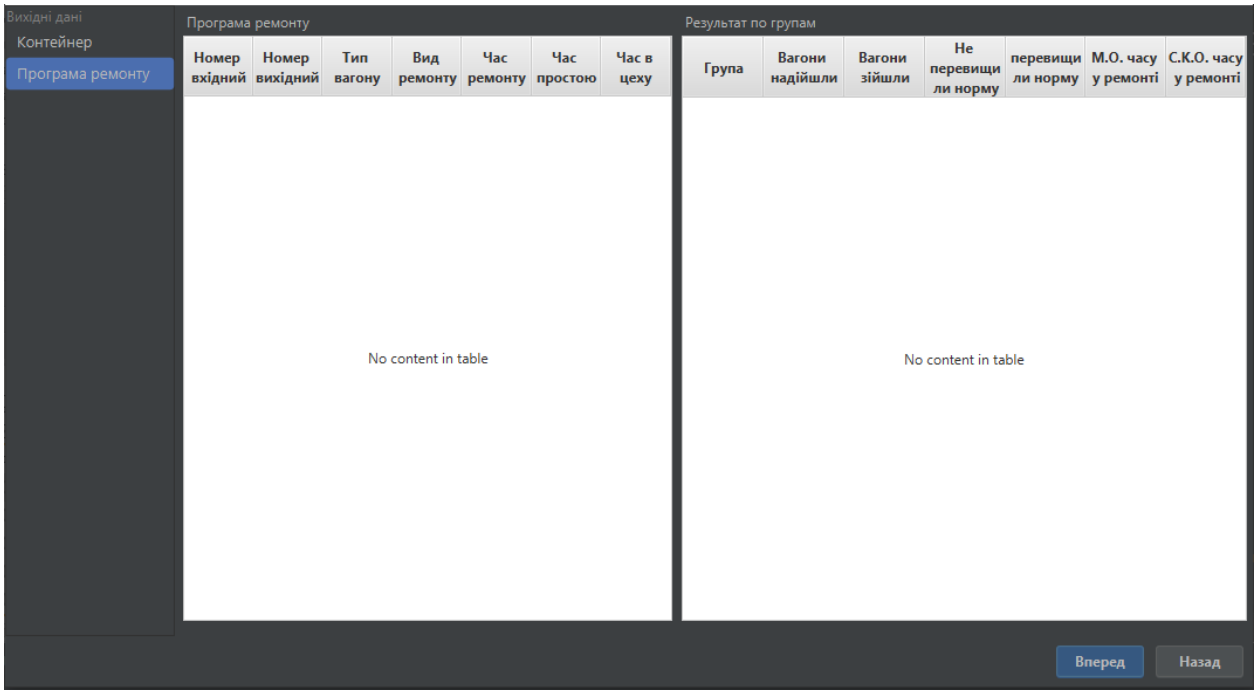

Рисунок 2.15 – Сторінка результату “Програма ремонту”

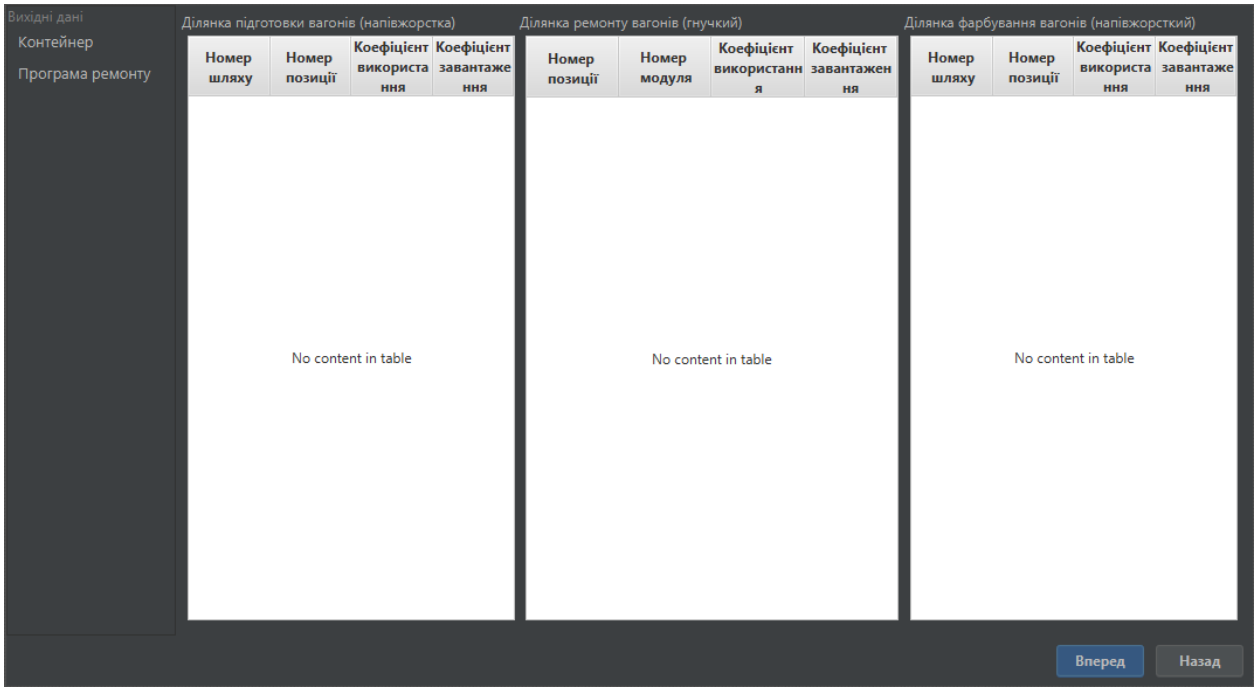

Рисунок 2.16 – Сторінка результату “Контейнер”

2.2.3 Проектування динаміки системи

На рис. 2.17 зображена діаграма послідовностей роботи програми.

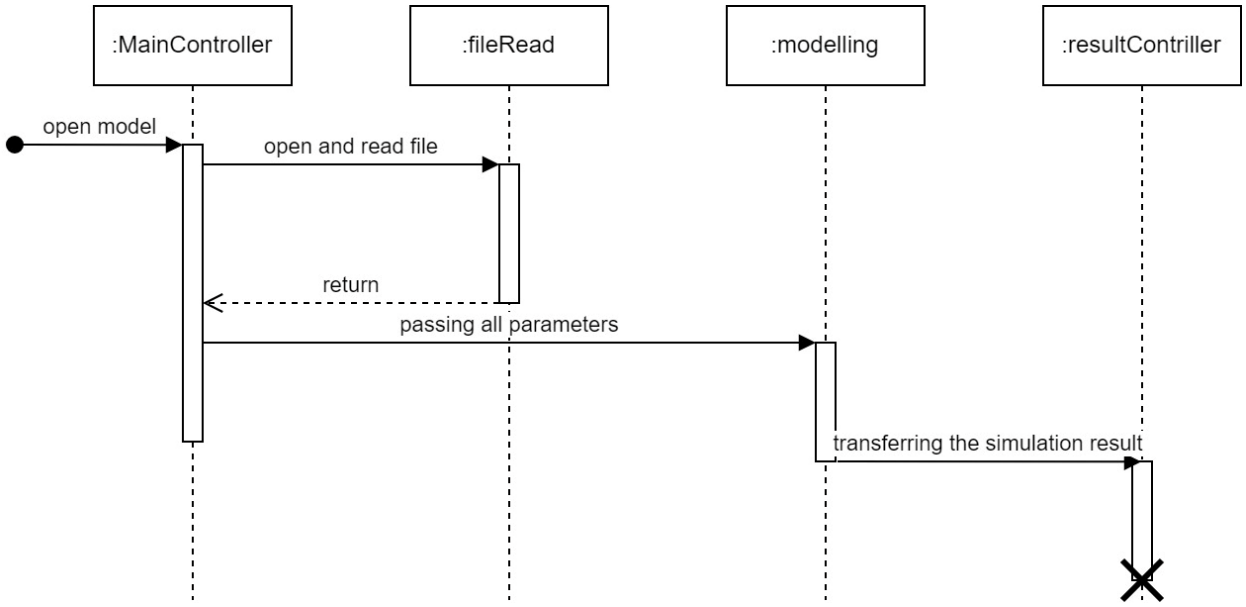

Рисунок 2.17 – Діаграма послідовностей роботи програми

Висновок до розділу 2

Під час зовнішнього проектування було встановлено основну функціональність програмного забезпечення, а саме, його здатність приймати конфігураційні значення на вхід та повертати результати моделювання.

У межах внутрішнього проектування була розроблена архітектура системи, визначено основні сутності проекту та встановлено способи їх взаємодії з інтерфейсом користувача та між собою.

Розділ 3. Розробка програми

3.1 Вибір мови проектування

Для реалізації розроблюваного програмного продукту було вибрано мову програмування Java. Оскільки основна мета полягає в моделюванні процесу, стабільність та швидкість виконання є найважливішими факторами, і саме в цих аспектах Java має перевагу. Крім того, вибір цієї мови програмування є важливим з урахуванням мого попереднього досвіду використання Java під час вивчання цієї мови в університеті та самостійно, що дає мені вже певний досвід у розв'язанні схожих завдань.

Для реалізації інтерфейсу програми була використана платформа JavaFX. Ця платформа, що базується на Java, призначена для створення програм з багатим графічним інтерфейсом. Вона може використовуватись як для розробки настільних програм, що запускаються безпосередньо на операційній системі, так і для інтернет-застосунків (RIA), що працюють у браузерах, а також для програм на мобільних пристроях. JavaFX була створена з метою заміни попередньої бібліотеки Swing, що використовувалася раніше. Вона конкурує з такими системами, як Microsoft Silverlight, Adobe Flash та іншими аналогічними рішеннями.

Висновок до розділу 3

Перш за все, Java була обрана через її переваги щодо стабільності та швидкості виконання. Оскільки основна мета полягає в моделюванні процесу, ці аспекти є найважливішими, і Java відома своєю ефективністю в цих сферах.

Крім того, вибір Java був зроблений з урахуванням попереднього досвіду використання цієї мови програмування. Це дає йому перевагу, оскільки я вже знайомий з мовою та може використовувати свій попередній досвід для ефективного розв'язання поточних завдань.

Нарешті, для реалізації інтерфейсу програми була використана платформа JavaFX, яка є розширенням Java для створення програм з графічним інтерфейсом. JavaFX забезпечує можливість розробки настільних програм. Вибір JavaFX базується на її багатих можливостях для створення привабливого графічного інтерфейсу.

Отже, вибір мови програмування Java та платформи JavaFX для реалізації розроблюваного програмного продукту зумовлений їхніми перевагами щодо стабільності, швидкості виконання, попереднього досвіду та можливостей для створення багатофункціонального графічного інтерфейсу.

Розділ 4. Тестування та налагодження

Для перевірки відповідності програми вимогам, вказаним у специфікації, та виявлення дефектів рекомендується проводити тестування за допомогою стратегії чорної скриньки, зокрема методу припущення про помилку. Під час використання методу припущення про помилку, тестувальник складає список можливих помилок та ситуацій, в яких ці помилки можуть виникнути. Цей метод ґрунтується в основному на власному

досвіді та інтуїції тестувальника програми. Для виявлення та локалізації знайдених помилок будуть використані методи індукції та дедукції.

Для налагодження програми будуть використовуватися такі засоби середовища розробки:

- Покрокове виконання програми: дозволяє виконувати програму по одному кроці за раз для детального аналізу й виявлення помилок.
- Точки зупину: дозволяють призупинити виконання програми на певному етапі для перевірки стану системи і аналізу даних.
- Перегляд значень змінних: надає можливість переглядати значення змінних в різних точках виконання програми для виявлення проблем та налагодження.

Ці засоби допоможуть спростити процес налагодження програми та знайти й виправити потенційні помилки.

Сформовано наступні припущення про помилку:

и може користувач відкрити наступну вкладку, якщо на вкладці “Структура потоку” всі ділянки не активні?

и може користувач відкрити наступну вкладку, якщо знаходиться на вкладках “Програма ремонту”, “Характеристики тимчасових моделей” та “Часи відновлення обладнання” та на них не введені всі потрібні дані?

Опис помилок	Опис ситуації	Способи усунення	Дії, що були застосовані для усунення
1	2	3	4
Чи може користувач відкрити наступну вкладку, якщо на вкладці “Структура потоку” всі ділянки не активні?	Користувач намагається відкрити наступну вкладку, якщо всі ділянки не активні.	Діактування кнопки “Вперед”.	Приписання логіки яка не дозволяє натискати кнопку “Вперед” поки одна або більше ділянок.
Чи може користувач відкрити наступну вкладку, якщо знаходиться на вкладках “Програма ремонту”, “Характеристики тимчасових моделей” та “Часи відновлення обладнання” та на них не введені всі потрібні	Користувач намагається відкрити наступну вкладку, якщо всі потрібна інформація не введена для моделювання.	Діактування кнопки “Вперед”.	Приписання всіх потрібні параметри для моделювання .

дані?			

Висновок до розділу 4

В даному розділі було обрано методи тестування та відлагодження додатку. Складено протокол налагодження програмної системи, що містить помилки та спосіб їх виправлення.

Для ефективного налагодження програми було обрано використовувати покрокове виконання програми, точки зупину і перегляд значень змінних. Ці засоби допоможуть аналізувати стан системи, виявляти помилки і здійснювати налагодження.

Висновки

Дипломна робота має на меті розробку програмного забезпечення для моделювання процесу ремонту на гнучких потокових виробництвах. Основний фокус роботи полягає в балансуванні модулів на позиціях і оптимізації ефективності роботи кожного модуля. Розроблений програмний продукт має практичне застосування у навчанні студентів процесу ремонту на гнучких потокових лініях.

Розроблений програмний продукт дозволяє додавати та видаляти позиції, вводити кількість модулів для кожної позиції, встановлювати час виконання робіт та нормальний допуск часу. Це дозволяє викладачам ілюструвати алгоритм гнучкого ремонту вагонів перед студентами, сприяючи їх кращому розумінню процесів на гнучких потокових виробництвах.

Під час розробки даного проекта був використаний архітектурний шаблон «Модель-Вигляд-Контролер», завдяки цьому, у майбутньому, буде легше удосконалювати програму, внаслідок того, що всі класи для моделювання вже прописані і осталось прописати потрібні алгоритмі для зміни моделі (наприклад, з жорстким чи напівжорстким зчепленням), достатньо прописати алгоритм без додаткових класів.

Також завдяки курсу по комп’ютерній графіці, виконання цього проекта було легше, бо вже було вивчено, як подібні проекти необхідно розроблювати, які сутності необхідні та як їх пов’язувати.

Отже, розроблений програмний продукт має значний потенціал у покращенні навчального процесу та сприяє кращому розумінню студентами процесів роботи на гнучких потокових виробництвах.

Додатки

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського державного університету науки і технологій

Анатолій РАДКЕВИЧ

18.02.22

ПРОГРАМА ДЛЯ МОДЕЛЮВАННЯ ТА РОЗРАХУНКУ ГНУЧКИХ ПОТОКОВИХ ТЕХНОЛОГІЙ РЕМОНТУ РУХОМОГО СКЛАДУ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

1116130.01298-01-ЛЗ

Представники

підприємства-розробника

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

18.02.22

Керівник розробки

Вадим ГОРЯЧКІН

18.02.22

Виконавець

Георгій СІНЬКОВ

18.02.22

Норм-контролер

Світлана ВОЛКОВА

18.02.22

2022

ЗАТВЕРДЖЕНО

1116130.01298-01-ЛЗ

ПРОГРАМА ДЛЯ МОДЕЛЮВАННЯ ТА РОЗРАХУНКУ ГНУЧКИХ ПОТОКОВИХ ТЕХНОЛОГІЙ РЕМОНТУ РУХОМОГО СКЛАДУ

Технічне завдання

1116130.01298-01-ЛЗ

Листів 16

2022

ЗМІСТ

Вступ

Метою даної роботи є розробка програмного забезпечення, яке здійснює моделювання процесу ремонту на гнучких потокових виробництвах. Основна увага приділяється виведенню модулів на позиціях, вагонів різних типів, які надходять до вільних модулів для проходження процесу ремонту, та транспортуванню цих вагонів між модулями.

Кожен вагон потребує різного часу для виконання робіт на кожному модулі. Наприклад, просте миття чистого вагона займе менше часу, ніж зварювання складної деталі на старому вагоні. Однак необхідно уникати ситуації, коли вагони, для яких робота вже завершена, простоюють на модулі, затримуючи загальний потік робіт.

Отже, кожен модуль на кожній позиції повинен працювати з максимальною ефективністю, уникати непотрібних простоїв. Це означає, що на позиціях, де робота займає багато часу, повинно бути більше модулів, а на позиціях з швидким проходженням процесу ремонту - менше модулів, щоб уникнути незайнятості простору в будівлі та недооптимізованого використання працівників та обладнання.

Практичне застосування розробленого програмного забезпечення полягає у навчанні студентів процесу ремонту на гнучких потокових лініях та в поясненні важливості балансування кількості модулів на позиціях.

1 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 08.12.21 №77ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи - “РОЗРОБКА ПРОГРАМИ ДЛЯ МОДЕЛЮВАННЯ ТА РОЗРАХУНКУ ГНУЧКИХ ПОТОКОВИХ ТЕХНОЛОГІЙ РЕМОНТУ РУХОМОГО СКЛАДУ”. Керівник - доцент Горячкін В. М.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – Розроблюваний програмний продукт має на меті моделювання процесу ремонту рухомого складу на гнучких потокових виробництвах з використанням введених даних. Він надає можливість додавати та видаляти позиції, вводити кількість модулів для кожної позиції, встановлювати середній час виконання робіт на цих позиціях та нормальний допуск часу від середнього значення.

Експлуатаційне призначення – Завдяки цьому програмному продукту, викладачі отримають можливість ілюструвати роботу алгоритму гнучкого ремонту вагонів наочним способом перед студентами та абітурієнтами. Це сприятиме кращому розумінню процесів на гнучких потокових виробництвах з боку студентів. Викладачі зможуть зрозуміло пояснити, що кількість модулів на позиції повинна змінюватися відповідно до часу, необхідного для виконання робіт на них.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Повинна бути можливість задавати:

- кількість модулів на позиції;
- середній час необхідний для виконання робіт на позиції;
- нормальний відступ від середнього часу;
- кількість вагонів, що повинні пройти через усі позиції;
- швидкість відображення анімації.

Також необхідна можливість додавати та видаляти позиції, а також спосіб зупинити/почати програвання музики.

Програмний продукт повинен давати змогу вводити данні для моделюванню, а саме таблицю кількість ремонтних позицій, кількість транспортних моделей, кількість груп ремонту, обсягу ремонту, крок моделювання, тривалість зміни, кількість зміни, кількість робочих змін, фонд робочого часу, С.К.О., М.О.

3.2 Опис зовнішнього інформаційного середовища

Всі дані для вводу отримуються із файлів(BogiesRecoveryData.txt, ObjData.txt, PosData.txt, ProgData.txt, RepRecoveryData.txt, TimeData.txt, TransData.txt, WorkTimeData.txt).

Користувач може перемикати всі вкладки програми за допомогою навігаційних кнопок або у списку назв цих вкладок.

Результат моделювання виводиться вигляді таблиць у окремому вікні.

Для функціонування ПЗ необхідно мати: встановлену операційну систему Windows 7 чи вище та встановлена JVM(Java Virtual Machine) на компютері користувача.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача.

Вся документація програмного додатку повинна задовольняти вимоги до програмної документації.

.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиці 1. – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.22 -18.02.22
Робочий проект	Програмування та відлагодження програми.	19.02.22 - 20.05.22
	Тестування програми	20.05.22 - 27.05.22
	Розробка, узгодження і затвердження програмної документації.	27.05.22 - 12.06.22

6 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Горячкін В. М.

Прийом здійснюється комісією у складі:

- Горячкін В. М. (керівник підрозділу);
- Горячкін В. М. (керівник розробки).
- Волкову С.А. (нормоконтролера)
- Мямліна С.В. (консультанта)

7 БІБЛІОГРАФІЧНИЙ СПИСОК

вченко, Ю.М. Основи стандартизації програмних систем: методичні вказівки до дипломного проектування та лабораторних робіт/уклад.: Ю.М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

ямлін, В. В. Развитие научных основ создания гибких поточных технологий ремонта подвижного состава [Текст]: диссертация / В. В. Мямлін. – Дніпропетровськ: Вид-во ЧФ «Стандарт-Сервис», 2015. – 380с.

араш Ю. С. Рациональные пути развития технической базы для деповского ремонта грузовых вагонов [Електронний ресурс] : автореф. thesis / Бараш Юрий Савельевич ; Днепропетровский институт инженеров железнодорожного транспорта. – [Б. м.], 1982. – Режим доступу: http://eadnurt.diit.edu.ua/jspui/handle/123456789/9255

ямлін, В. В. Теоретические основы создания гибких поточных производств для ремонта подвижного состава [Текст]: монографія / В. В. Мямлін. –Дніпропетровськ: Вид-во ЧФ «Стандарт-Сервис», 2014. – 380с.

естування програмного забезпечення [Текст]: навчальний посібник / Авраменко А.С., Авраменко В.С. Косенюк Г.В. – Черкаси : ЧНУ імені Богдана Хмельницького, 2017. – 284 с.

кість програмного забезпечення та тестування [Текст]: методичні вказівки до лабораторних робіт / уклад.: В. І. Шинкаренко, О. С. Куроп’ятник, Г. В. Забула, Д. О. Пєтін, Є. В. Лукін, Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во ПФ «Стандарт-Сервіс», 2018. – 50 с

Додаток Б

Текст програми

TCDF.java

```
// Клас фукнції розподілу
public class TCDF {
   private double[] FX; // Аргумент
   private double[] FF; // Функція розподілу
   private double[] FD; // Щільність розподілу
   private double FMean; // Середнє значення
   private double FVariance; // Дисперсія
   private int FN; // Довжина функції розподілу
   private Random FRnd; // Генератор рівномірно розподілених випадкових чисел
   private double FRepairTime; // Час обслуговування

   private void CalcCDF(double[] Data) {
      int NEqial; // Кількість послідовно розташованих однакових даних
      Stack<Double> CDF = new Stack<>();  // Функція розподілу
      Stack<Double> PDF = new Stack<>();  // Щільність ймовірності
      Stack<Double> CDA = new Stack<>();  // Аргумент функції розподілу

      // Визначити середнє та дисперсію
      FMean = 0;
      for (int n = 0; n < Data.length; n++) {
         FMean += Data[n];
      }
      FMean /= Data.length;
      for (int n = 0; n < Data.length; n++) {
         FVariance += (Data[n] - FMean) * (Data[n] - FMean);
      }
      FVariance /= Data.length - 1;

      // Побудувати функцію розподілу
      for (int n = 0; n < Data.length; n++) {
         // Посчитать кол-во одинаковых чисел
         NEqial = 1;
         while (n + NEqial < Data.length && Data[n] == Data[n + NEqial]) {
            NEqial++;
         }

         // Порахувати таке значення функції розподілу
         if (CDF.empty()) {
            CDF.push((double) NEqial / Data.length);
         } else {
            CDF.push(CDF.peek() + (double) NEqial / Data.length);
         }

         PDF.push((double) NEqial / Data.length);

         CDA.push(Data[n]);

         n += NEqial - 1; // Перейти до наступного числа
      }

      // Переписати результат до масивів
      FN = CDF.size();
```

```
      FX = new double[FN];
      FF = new double[FN];
      FD = new double[FN];
      for (int n = 0; n < FN; n++) {
         FF[FN - 1 - n] = CDF.pop();
         FD[FN - 1 - n] = PDF.pop();
         FX[FN - 1 - n] = CDA.pop();
      }
   }

   public TCDF(String FileName) {
      double[] InData = new
double[1000];
      double[] Data;
      int k = 0;
      FRepairTime = 0;
      try {
         java.io.BufferedReader br =
new java.io.BufferedReader(new
java.io.FileReader(FileName));
         while (true) {
            try {
               InData[k] =
Double.parseDouble(br.readLine());
               k += 1;
            } catch (Exception ex) {
               Data =
Arrays.copyOfRange(InData, 0, k);
               Arrays.sort(Data); //
Сортування даних щодо зростання
               CalcCDF(Data); //
Розрахунок функції розподілу
               FRnd = new Random();
               break;
            }
         }
         br.close();
      } catch (Exception ex) {

javax.swing.JOptionPane.showMessa
geDialog(null, "ВременнЫе
параметры обслуживания не
найдены. Моделирование
прервано.", "Ошибка",
javax.swing.JOptionPane.WARNING
_MESSAGE);
      }
   }

   public TCDF(double[] Data) {
      FRepairTime = 0;
      Arrays.sort(Data); // Сортування
даних щодо зростання
      CalcCDF(Data); // Розрахунок
функції розподілу
      FRnd = new Random();
   }

   public double getTime(double
Time) {
      double UniVal;
      double EDFVal;
      FRepairTime = 0;
      if (FRnd != null) {
         UniVal = FRnd.nextDouble();
// Отримати рівномірно розподілене
випадкове число
         int n = 0; // Отримати
випадкове число за емпіричною
функцією розподілу
         while (FF[n] < UniVal) {
            n += 1;
         }
         if (n == 0) {
            EDFVal = FX[0];
         } else {
            EDFVal = FX[n - 1] +
(FX[n] - FX[n - 1]) / (FF[n] - FF[n - 1])
* (UniVal - FF[n - 1]);
         }
         FRepairTime = EDFVal * 60;
// Переклад за секунди
         FRepairTime =
Math.round(FRepairTime / 60) * 60; //
Округлити до хвилини
         FRepairTime += Time;
      }
      return FRepairTime;
```

```
    }

    public int getN() {
        return FN;
    }

    public double[] getX() {
        return FX;
    }

    public double[] getF() {
        return FF;
    }

    public double[] getP() {
        return FD;
    }

    public double getM() {
        return FMean;
    }

    public double getS() {
        return Math.sqrt(FVariance);
    }

    public double getTime() {
        return FRepairTime;
    }

}
```

TCommonResults.java

```
public class TCommonResults {
    public int EnteredWagons;
    public int RepairedWagons;
    public int TotalWorkingTime;
    public int NLessTime;
    public int NMoreTime;
    public List<TRepairObject> RepairedObject;
    public List<TRepairedObjects> CategoryObjects;

    public double MeanRepairTime;
    public double DeviationRepairTime;
    public double MeanCycleLength;
    public double DeviationCycleLength;

    // Конструктор
    public TCommonResults() {
        EnteredWagons = 0;
        RepairedWagons = 0;
        TotalWorkingTime = 0;
        NLessTime = 0;
        NMoreTime = 0;

        RepairedObject = new ArrayList<TRepairObject>();
        CategoryObjects = new ArrayList<TRepairedObjects>();

        MeanRepairTime = 0;
        DeviationRepairTime = 0;
        MeanCycleLength = 0;
        DeviationCycleLength = 0;
    }

    public void clear() {
        EnteredWagons = 0;
        RepairedWagons = 0;
        TotalWorkingTime = 0;
        NLessTime = 0;
        NMoreTime = 0;

        RepairedObject.clear();
        CategoryObjects.clear();

        MeanRepairTime = 0;
        DeviationRepairTime = 0;
        MeanCycleLength = 0;
        DeviationCycleLength = 0;
    }
}
```

TConveyer.java

```
public class TConveyer {
```

```
    private TRepairWays FPrepareArea; // Участок подготовки
    private TRepairArea FRepairArea; // Участок ремонта
    private TRepairWays FPaintingArea; // Участок окраски

    // Конструктор
    public TConveyer(int PrepWayCount, int PrepPosCount, double PrepMTime, double[] PrepFailure, double[] PrepRecoveryMean, double[] PrepRecoveryDeviation,
                     int PaintWayCount, int PaintPosCount, double PaintMTime, double[] PaintFailure, double[] PaintRecoveryMean, double[] PaintRecoveryDeviation,
                     int RepPosCount, int TrModCount) {
        FPrepareArea = new TRepairWays(PrepWayCount, PrepPosCount, PrepMTime, PrepFailure, PrepRecoveryMean, PrepRecoveryDeviation);
        FRepairArea = new TRepairArea(RepPosCount, TrModCount);
        FPaintingArea = new TRepairWays(PaintWayCount, PaintPosCount, PaintMTime, PaintFailure, PaintRecoveryMean, PaintRecoveryDeviation);
    }

    // Установить параметры ремонтных позиций
    public void SetPositionParameters(int[] ModCount, double[] MovTimes, double[] Failure, double[] RecoverMean, double[] RecoverDeviation) {
        FRepairArea.setPositionParameters(ModCount, MovTimes, Failure, RecoverMean, RecoverDeviation);
    }

    // Установить параметры транспортных модулей
    public void SetTrModuleParameters(String[] ServZone, double[] Failure, double[] RecoverMean, double[] RecoverDeviation) {
        FRepairArea.setTrModuleParameters(ServZone, Failure, RecoverMean, RecoverDeviation);
    }

    public boolean IsFree() {
        // Проверка свободности конвейера
        boolean Result = false;
        if (FPrepareArea.IsFree() && FRepairArea.isFree() && FPaintingArea.IsFree()) {
            Result = true;
        }
        return Result;
    }

    public TRepairWays getPrepareArea() {
        return FPrepareArea;
    }

    public TRepairWays getPaintingArea() {
        return FPaintingArea;
    }

    public TRepairArea getRepairArea() {
        return FRepairArea;
```

```
  }
}
```

TGaussTime.java

```
// Клас нормального розподілу
public class TGaussTime {
  private double FMean; // Середній час обслуговування
  private double FDeviation; // Середнє квадратичне відхилення часу обслуговування
  private double FRepairTime; // Поточний час обслуговування

  // Отримати випадкове число
  private double GetRandVal() {
    double RVal = 0;

    for (int k = 1; k <= 12; k++) {
      RVal = RVal + Math.random();
    }
    RVal = RVal - 6;
    RVal = RVal - 41 * (Math.pow(RVal, 5) - 10 * Math.pow(RVal, 3) + 15 * RVal) / 1935360;

    return RVal;
  }

  public TGaussTime() {
    FMean = 0;
    FDeviation = 0;
    FRepairTime = 0;
  }

  // Установка значений параметров
  public void setValues(double M, double D) {
    FMean = M * 60;
    FDeviation = D * 60; // Переклад за секунди
  }

  // Розрахунок моменту закінчення обслуговування
  public double getTime(double Time) {
    double RVal = GetRandVal();

    // Визначити тривалість ремонту
    FRepairTime = FMean + FDeviation * RVal;
    if (FRepairTime < 0) FRepairTime = 0;
    FRepairTime = Math.round(FRepairTime / 60) * 60; // Округлити до хвилини

    FRepairTime += Time; // Додати поточний час (вийде момент закінчення обслуговування)

    return FRepairTime;
  }

  // Середнє
  public double getMean() {
    return FMean;
  }

  public void setMean(double value) {
    FMean = value;
  }

  // С.К.О.
  public double getDeviation() {
    return FDeviation;
  }

  public void setDeviation(double value) {
    FDeviation = value;
  }
```

```
   // Поточний час обслуговування
   public double getTime() {
      return FRepairTime;
   }
}
```

TModule.java

```
// Клас модуля
public class TModule {
   protected double FFailure; // Імовірність відмови модуля
   protected TTimeFeatures FRecoverTime; // Тимчасові параметри відновлення працездатності модуля
   protected TRepairObject FRepairObject; // Вагон, що знаходиться зараз у модулі
   protected boolean FOccupied; // Прапор зайнятості модуля
   protected boolean FWagMoving; // Прапор руху вагона до модуля
   protected boolean FOperable; // Прапор працездатності модуля
   protected double FOccupationTime; // Момент переходу модуля у зайнятий стан
   protected double FTotalOccupiedTime; // Загальний час зайнятості модуля

   // Конструктор
   public TModule(double Failure, double RecoverMean, double RecoverDeviation) {
      FFailure = 1 - Failure;
      FRecoverTime = new TTimeFeatures();

FRecoverTime.setValues(RecoverMean, RecoverDeviation);
   }

   // Скидання параметрів
   public void Reset() {
      FOccupied = false;
      FOperable = true;
      FWagMoving = false;
      FTotalOccupiedTime = 0;
      FOccupationTime = 0;
   }

   // Вагон, що знаходиться зараз у модулі
   public TRepairObject getRepairObject() {
      return FRepairObject;
   }

   public void setRepairObject(TRepairObject value) {
      FRepairObject = value;
      FOccupied = true;
   }

   // Прапор зайнятості
   public boolean isOccupied() {
      return FOccupied;
   }

   public void setOccupied(boolean value) {
      FOccupied = value;
      if (!value) { // value == false
         // Модуль звільнився -> Прибрати вагон
         FRepairObject = null;
      }
   }

   // Прапор працездатності модуля
   public boolean isOperable() {
      return FOperable;
   }

   public void setOperable(boolean value) {
      FOperable = value;
```

```
    }

    // Прапор руху вагона до модуля
    public boolean isWagMoving() {
        return FWagMoving;
    }

    public void setWagMoving(boolean value) {
        FWagMoving = value;
    }

    // Ймовірність несправності
    public double getFailure() {
        return FFailure;
    }

    // Час відновлення працездатності модуля
    public TTimeFeatures getRecoverTime() {
        return FRecoverTime;
    }

    // Момент переходу модуля у зайнятий стан
    public double getOccupationTime() {
        return FOccupationTime;
    }

    public void setOccupationTime(double value) {
        FOccupationTime = value;
    }

    // Загальний час зайнятості
    public double getTotalOccupiedTime() {
        return FTotalOccupiedTime;
    }
}
```

TRandomGenerator.java

```
// Генератор випадкових чисел
public class TRandomGenerator {
    private Random FRndX;
    private Random FRndY;
    private double FMean;
    private double FDeviation;
    private TRandomValues FAuxRndVal; // Випадкові величини діапазону 0-1
    private TRandomValues FRndVal; // Випадкові величини із середнім Mean та п.к.о. Deviation
    private boolean FCNor;
    private boolean FCRel;
    private boolean FCExp;

    // Конструктор
    public TRandomGenerator(double mean, double deviation, boolean nor, boolean rel, boolean exp) {
        FAuxRndVal = new TRandomValues();
        FAuxRndVal.UniX = 0;
        FAuxRndVal.UniY = 0;
        FAuxRndVal.NorX = 0;
        FAuxRndVal.NorY = 0;
        FAuxRndVal.RelX = 0;
        FAuxRndVal.ExpX = 0;

        FRndVal = new TRandomValues();
        FRndVal.UniX = 0;
        FRndVal.UniY = 0;
        FRndVal.NorX = 0;
        FRndVal.NorY = 0;
        FRndVal.RelX = 0;
        FRndVal.ExpX = 0;

        FMean = mean;
        FDeviation = deviation;

        FCNor = nor;
        FCRel = rel;
        FCExp = exp;
```

```
        FRndX = new Random();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        FRndY = new Random();
    }

    // Отримати випадкове число
    public TRandomValues Values() {
        double RndValX;
        double RndValY;

        RndValX                    =
FRndX.nextDouble();
        RndValY                    =
FRndY.nextDouble();

        // Рівномірне
        FAuxRndVal.UniX = RndValX;
        FAuxRndVal.UniY = RndValY;
        FRndVal.UniX = FMean +
FAuxRndVal.UniX * FDeviation;

        // Нормальне
        if (FCNor) {
            FAuxRndVal.NorX            =
Math.cos(2.0 * Math.PI * RndValX) *
Math.sqrt(-2.0 * Math.log(RndValY));
            FAuxRndVal.NorY            =
Math.sin(2.0 * Math.PI * RndValX) *
Math.sqrt(-2.0 * Math.log(RndValY));
            if (FAuxRndVal.NorX < 0) {
                FAuxRndVal.NorX = 0;
            }
            if (FAuxRndVal.NorY < 0) {
                FAuxRndVal.NorY = 0;
            }
            FRndVal.NorX = FMean +
FAuxRndVal.NorX * FDeviation;
        }

        // Релея
        if (FCRel) {
            FAuxRndVal.RelX            =
Math.sqrt(FAuxRndVal.NorX          *
FAuxRndVal.NorX                    +
FAuxRndVal.NorY                    *
FAuxRndVal.NorY);
            if (FAuxRndVal.RelX < 0) {
                FAuxRndVal.RelX = 0;
            }
            FRndVal.RelX = FMean +
FAuxRndVal.RelX * FDeviation;
        }

        // Експонентне
        if (FCExp) {
            FAuxRndVal.ExpX = -
Math.log(1.0 - FAuxRndVal.UniX) /
FDeviation;
            if (FAuxRndVal.RelX < 0) {
                FAuxRndVal.RelX = 0;
            }
            FRndVal.ExpX = FMean +
FAuxRndVal.ExpX * FDeviation;
        }

        return FRndVal;
    }
}
```

TRepairArea.java

```
public class TRepairArea {
    private int FPositionsCount; //
Количество ремонтных позиций
    private int FTrModulesCount; //
Количество транспортных модулей
    private TRepairPosition[]
FRepairPositions; // Массив
ремонтных позиций
    private double[] FMovingTimes; //
Массив времен перемещений
вагона между позициями
    private TTransportModule[]
FTransportModules; // Массив
```

```
транспортных модулей

  // Конструктор
  public TRepairArea(int PosCount, int TrModCount) {
    // Установить размеры массивов ремонтных позиций
    FPositionsCount = PosCount;
    FRepairPositions = new TRepairPosition[FPositionsCount];
    FMovingTimes = new double[FPositionsCount];

    // Установить размеры массива транспортных модулей
    FTrModulesCount = TrModCount;
    FTransportModules = new TTransportModule[FTrModulesCount];
  }

  public void setPositionParameters(int[] ModCount, double[] MovTimes, double[] Failure, double[] RecoverMean, double[] RecoverDeviation) {
    // Создать ремонтные позиции
    for (int n = 0; n < FPositionsCount; n++) {
      FRepairPositions[n] = new TRepairPosition(ModCount[n], Failure[n], RecoverMean[n], RecoverDeviation[n]);
      FMovingTimes[n] = MovTimes[n] * 60; // Перевод в секунды
    }
  }

  public void setTrModuleParameters(String[] ServZone, double[] Failure, double[] RecoverMean, double[] RecoverDeviation) {
    // Создать транспортные модули
    for (int n = 0; n < FTrModulesCount; n++) {
      FTransportModules[n] = new TTransportModule(ServZone[n], Failure[n], RecoverMean[n], RecoverDeviation[n]);
    }
  }

  public boolean isFree() {
    // Проверка свободности всех позиций и транспортных модулей
    boolean result = true;

    for (TRepairPosition position : FRepairPositions) {
      if (!position.IsFree()) {
        result = false;
        break;
      }
    }
    if (result) {
      // Позиции свободны -> Проверить транспортные модули
      for (TTransportModule transportModule : FTransportModules) {
        if (transportModule.isOccupied()) {
          result = false;
          break;
        }
      }
    }
    return result;
  }

  public int getPositionsCount() {
    return FPositionsCount;
  }
```

```
    public int getTrModuleCount() {
        return FTrModulesCount;
    }

    public TRepairPosition
getRepairPosition(int n) {
        return FRepairPositions[n];
    }

    public TRepairPosition
getLastPosition() {
        return
FRepairPositions[FPositionsCount -
1];
    }

    public double getMovingTime(int
n) {
        return FMovingTimes[n];
    }

    public TTransportModule
getTransportModule(int n) {
        return FTransportModules[n];
    }
}
```

TRepairedObjects.java

```
public class TRepairedObjects { //
Структура для відремонтованих
вагонів
    public int NLessTime;
    public int NMoreTime;
    public double MeanRepairTime;
    public double
DeviationRepairTime;

    public List<TRepairObject>
Wagons;

    // Конструктор
    public TRepairedObjects() {
        NLessTime = 0;
        NMoreTime = 0;
        MeanRepairTime = 0;
        DeviationRepairTime = 0;
        Wagons = new
ArrayList<TRepairObject>();
    }

    public void clear() {
        NLessTime = 0;
        NMoreTime = 0;
        MeanRepairTime = 0;
        DeviationRepairTime = 0;
        Wagons.clear();
    }
}
```

TRepairModule.java

```
// Ремонтний модуль
public class TRepairModule extends
TModule {
    private boolean FWorking; //
Прапор роботи модуля
    private double FStartWorkingTime;
// Час початку роботи модуля
    private double
FTotalWorkingTime; // Загальний
час роботи модуля

    // Конструктор
    public TRepairModule(double
Failure, double RecoverMean, double
RecoverDeviation) {
        super(Failure, RecoverMean,
RecoverDeviation);
        Reset();
    }

    // Скидання параметрів
    @Override
    public void Reset() {
        super.Reset();
        FWorking = false;
        FTotalWorkingTime = 0;
    }
```

```
   // Прапор роботи
   public boolean getWorking() {
      return FWorking;
   }
   public void setWorking(boolean
value) {
      FWorking = value;
   }

   // Завершення роботи модуля
   public boolean getWorkFinished() {
      boolean Finished = false;

      if (FWorking == false &&
FStartWorkingTime >=
getOccupationTime()) {
         Finished = true;
      }
      return Finished;
   }

   // Час початку роботи модуля
   public double
getStartWorkingTime() {
      return FStartWorkingTime;
   }
   public void
setStartWorkingTime(double value) {
      FStartWorkingTime = value;
   }

   // Загальний час роботи
   public double
getTotalWorkingTime() {
      return FTotalWorkingTime;
   }
   public void
setTotalWorkingTime(double Time) {
      FTotalWorkingTime = Time;
   }
}
```

TRepairObject.java

```
public class TRepairObject {
   private String FType;
   private String FMode;
   private int FCategory;
   private double FRTime;
   private boolean FPriority;
   private int FEnterNumber;
   private int FExitNumber;
   private boolean FRepairing;
   private boolean FMoving;
   private TTimeFeatures[]
FPrepareTimes;
   private TTimeFeatures[]
FRepairTimes;
   private TTimeFeatures[]
FPaintingTimes;
   private double FStartRepairTime;
   private double FTotalRepairTime;
   private double
FStartConveyerTime;
   private double
FTotalConveyerTime;
   private double FStopMovingTime;

   // Constructors
   public TRepairObject() {
      FTotalRepairTime = 0;
      FTotalConveyerTime = 0;
   }

   public TRepairObject(String type,
String mode, int PrepPosCount, int
RepPosCount, int PaintPosCount) {
      FType = type;
      FMode = mode;
      FTotalRepairTime = 0;
      FTotalConveyerTime = 0;
      FMoving = false;
      FRepairing = false;

      // Set the size of the preparation
time features array
      FPrepareTimes = new
TTimeFeatures[PrepPosCount];
```

```
      // Create repair time features
      for (int n = 0; n < PrepPosCount;
n++) {
        FPrepareTimes[n] = new
TTimeFeatures();
      }

      // Set the size of the repair time
features array
      FRepairTimes = new
TTimeFeatures[RepPosCount];
      // Create repair time features
      for (int n = 0; n < RepPosCount;
n++) {
        FRepairTimes[n] = new
TTimeFeatures();
      }

      // Set the size of the painting time
features array
      FPaintingTimes = new
TTimeFeatures[PaintPosCount];
      // Create repair time features
      for (int n = 0; n < PaintPosCount;
n++) {
        FPaintingTimes[n] = new
TTimeFeatures();
      }
    }

    public TRepairObject(int
PrepPosCount, int RepPosCount, int
PaintPosCount) {
      FTotalRepairTime = 0;
      FTotalConveyerTime = 0;

      // Set the size of the preparation
time features array
      FPrepareTimes = new
TTimeFeatures[PrepPosCount];
      // Create repair time features
      for (int n = 0; n < PrepPosCount;
n++) {
        FPrepareTimes[n] = new
TTimeFeatures();
      }

      // Set the size of the repair time
features array
      FRepairTimes = new
TTimeFeatures[RepPosCount];
      // Create repair time features
      for (int n = 0; n < RepPosCount;
n++) {
        FRepairTimes[n] = new
TTimeFeatures();
      }

      // Set the size of the painting time
features array
      FPaintingTimes = new
TTimeFeatures[PaintPosCount];
      // Create repair time features
      for (int n = 0; n < PaintPosCount;
n++) {
        FPaintingTimes[n] = new
TTimeFeatures();
      }
    }


    public String getMode() {
      return FMode;
    }

    public void setMode(String m) {
      FMode = m;
    }


    public int getCategory() {
      return FCategory;
    }

    public void setCategory(int value) {
      FCategory = value;
    }
```

```
    public double getRTime() {
        return FRTime;
    }

    public void setRTime(double value)
{
        FRTime = value;
    }


    public boolean getPriority() {
        return FPriority;
    }

    public void setPriority(boolean p) {
        FPriority = p;
    }


    public                TTimeFeatures
getPrepareTimes(int n) {
        return FPrepareTimes[n];
    }

    public                void
setPrepareTimes(TTimeFeatures[]
value) {
        FPrepareTimes = value;
    }


    public                TTimeFeatures
getRepairTimes(int n) {
        return FRepairTimes[n];
    }

    public                void
setRepairTimes(TTimeFeatures[]
value) {
        FRepairTimes = value;
    }
    public                TTimeFeatures
getPaintingTimes(int n) {
        return FPaintingTimes[n];
    }

    public                void
setPaintingTimes(TTimeFeatures[]
value) {
        FPaintingTimes = value;
    }


    public boolean isRepairing() {
        return FRepairing;
    }


    public boolean isMoving() {
        return FMoving;
    }

    public    void    setMoving(boolean
value) {
        FMoving = value;
    }


    public                double
getStopMovingTime() {
        return FStopMovingTime;
    }

    public                void
setStopMovingTime(double value) {
        FStopMovingTime = value;
    }


    public double getStartRepairTime()
{
        return FStartRepairTime;
    }

    public                void
```

```
setStartRepairTime(double value) {
      FStartRepairTime = value;
   }


   public double getTotalRepairTime()
{
      return FTotalRepairTime;
   }

   public                          void
setTotalRepairTime(double time) {
      FTotalRepairTime = time;
   }


   public                          double
getStartConveyerTime() {
      return FStartConveyerTime;
   }

   public                          void
setStartConveyerTime(double value) {
      FStartConveyerTime = value;
   }

   public                          double
getTotalConveyerTime() {
      return FTotalConveyerTime;
   }

   public                          void
setTotalConveyerTime(double   value)
{
      FTotalConveyerTime = value;
   }


   public int getEnterNumber() {
      return FEnterNumber;
   }

   public   void   setEnterNumber(int
value) {
      FEnterNumber = value;
   }

   public int getExitNumber() {
      return FExitNumber;
   }

   public   void   setExitNumber(int
value) {
      FExitNumber = value;
   }

   public String getFType() {
      return FType;
   }

   public void setFType(String FType)
{
      this.FType = FType;
   }
}
```

TRepairProgram.java

```
public class TRepairProgram {
   private boolean FExecuted; // Флаг
выполненности программы ремонта
   private int FObjTypeCount; // Кол-
во типов объектов ремонта
   private   int[]   FObjectCount;   //
Массив количеств объектов
ремонта
   private double[] FObjProbability; //
Массив частостей появления
объектов различных типов в объеме
ремонта
   private int FCount; // Объем
ремонта (общее кол-во объектов
ремонта)
   private int FIndex; // Индекс вагона
в начале очереди (следующего
поступающего на конвейер)
   private TRepairObject[] FObjects; //
Массив объектов ремонта
   private Random Rnd; // Генератор
```

случайных номеров объектов

```
  public TRepairProgram() {
  }

  public TRepairProgram(int OTCount, int[] OTCounts, int PrepPosCount, int RepPosCount, int PaintPosCount) {
      FCount = 0;
      FIndex = 0;
      FExecuted = false;
      FObjTypeCount = OTCount;
      FObjectCount = new int[FObjTypeCount];
      FObjProbability = new double[FObjTypeCount];
      // Установить количества объектов ремонта
      for (int n = 0; n < FObjTypeCount; n++) {
        FObjectCount[n] = OTCounts[n];
        FCount += FObjectCount[n];
      }
      for (int n = 0; n < FObjTypeCount; n++) {
        if (n == 0) {
          FObjProbability[n] = FObjectCount[n] / (double)FCount;
        } else {
          FObjProbability[n] = FObjProbability[n - 1] + FObjectCount[n] / (double)FCount;
        }
      }
      FObjects = new TRepairObject[FCount];
      for (int n = 0; n < FCount; n++) {
        FObjects[n] = new TRepairObject(PrepPosCount, RepPosCount, PaintPosCount);
      }
      Rnd = new Random(0);
  }

  public int GetObjectIndex() {
    int ObjIndex = -1;
    int Count = 0;
    int LastObjIndex = 0;
    double Probability;
    for (int n = 0; n < FObjTypeCount; n++) {
      if (FObjectCount[n] > 0) {
        LastObjIndex = n;
        Count += 1;
      }
    }
    if (Count > 1) {
      do {
        Probability = Rnd.nextDouble();
        for (int n = 0; n < FObjTypeCount; n++) {
          if (Probability < FObjProbability[n]) {
            ObjIndex = n;
            break;
          }
        }
      } while (FObjectCount[ObjIndex] <= 0);
      FObjectCount[ObjIndex] -= 1;
    } else {
      ObjIndex = LastObjIndex;
      FObjectCount[ObjIndex] -= 1;
    }
    return ObjIndex;
  }

  public TRepairObject getNextObject() {
    TRepairObject nextObject = null;
    if (FIndex < FCount) {
      nextObject = FObjects[FIndex];
      FIndex++;
    } else {
```

```
        int index = FIndex % FCount;
        TRepairObject newObj = new
TRepairObject();

newObj.setMode(FObjects[index].get
Mode());
        newObj.setPaintingTimes(new
TTimeFeatures[]{FObjects[index].get
PaintingTimes(index)});
        newObj.setPrepareTimes(new
TTimeFeatures[]{FObjects[index].get
PrepareTimes(index)});

newObj.setPriority(FObjects[index].g
etPriority());
        newObj.setRepairTimes(new
TTimeFeatures[]{FObjects[index].get
RepairTimes(index)});

newObj.setRTime(FObjects[index].ge
tRTime());

newObj.setFType(FObjects[index].ge
tFType());

newObj.setCategory(FObjects[index].
getCategory());
        nextObject = newObj;
        FIndex++;
      }
      return nextObject;
    }

    public boolean isExecuted() {
      return FExecuted;
    }

    public int getCount() {
      return FCount;
    }

    public                TRepairObject
getObjects(int n) {
      return FObjects[n];
    }

}
```

TRepairWay.java

```
// Ремонтний шлях
public class TRepairWay {
  private int FPositionsCount; //
Кількість ремонтних позицій
  private TRepairPosition[]
FRepairPositions; // Масив
ремонтних позицій
  private double FMoveTime; // Час
переміщення між позиціями

  // Конструктор
  public TRepairWay(int rpCount,
double mTime, double[] Failure,
double[] RecoverMean, double[]
RecoverDeviation) {
    FPositionsCount = rpCount;
    FMoveTime = mTime * 60; //
Перевод в секунды

    // Установить размеры массива
ремонтных позиций
    FRepairPositions = new
TRepairPosition[FPositionsCount];

    // Создать ремонтные позиции
    for (int n = 0; n <
FPositionsCount; n++) {
      FRepairPositions[n] = new
TRepairPosition(1, Failure[n],
RecoverMean[n],
RecoverDeviation[n]);
    }
  }

  public boolean IsFree() {
    // Проверка свободности всех
позиций
    boolean Result = true;
```

```
        for (TRepairPosition Position : FRepairPositions) {
            if (!Position.IsFree()) {
                Result = false;
                break;
            }
        }

        return Result;
    }

    public int getPositionsCount() {
        return FPositionsCount;
    }

    public TRepairModule getRepairPositions(int n) {
        return FRepairPositions[n].getRepairModules(n);
    }

    public TRepairModule getLastPosition() {
        return FRepairPositions[FPositionsCount - 1].getRepairModules();
    }

    public double getMoveTime() {
        return FMoveTime;
    }
}
```

TRepairWays.java

```
public class TRepairWays {
    private int FWayCount;
    private TRepairWay[] FWays;

    // Конструктор
    public TRepairWays(int wCount, int rpCount, double mTime, double[] failure, double[] RecoverMean, double[] RecoverDeviation) {
        FWayCount = wCount;

        // Установить размеры массива ремонтных путей
        FWays = new TRepairWay[FWayCount];

        // Создать ремонтные пути
        for (int k = 0; k < FWayCount; k++) {
            FWays[k] = new TRepairWay(rpCount, mTime, failure, RecoverMean, RecoverDeviation);
        }
    }

    public boolean IsFree() {
        // Проверка свободности всех путей
        boolean Result = true;

        for (TRepairWay Way : FWays) {
            if (!Way.IsFree()) {
                Result = false;
                break;
            }
        }

        return Result;
    }

    public int getWayCount() {
        return FWayCount;
    }

    public TRepairWay getWays(int n) {
        return FWays[n];
    }
}
```

TTimeFeatures.java

```
// Тимчасові параметри
```

```
обслуговування
public class TTimeFeatures {
   private double FRepairTime; // Момент закінчення обслуговування
   private TGaussTime FGaussTime; // Нормальний розподіл часу
   private TCDF FEmpiricTime; // Емпіричний розподіл часу

   // Конструктор
   public TTimeFeatures() {
      FRepairTime = 0;
      FGaussTime = null;
      FEmpiricTime = null;
   }

   // Встановлення значень параметрів для нормального розподілу часу
   public void setValues(double M, double D) {
      FGaussTime = new TGaussTime();
      FGaussTime.setMean(M * 60);
      FGaussTime.setDeviation(D * 60); // Переклад за секунди
   }

   // Встановлення значень параметрів для емпіричного розподілу часу
   public void setValues(String fileName) {
      FEmpiricTime = new TCDF(fileName);
   }

   // Розрахунок моменту закінчення обслуговування
   public double getTime(double time) {
      if (FGaussTime != null) {
         FRepairTime = FGaussTime.getTime(time); // Нормальний розподіл часу обслуговування
      } else if (FEmpiricTime != null) {
         FRepairTime = FEmpiricTime.getTime(time); // Експериментальний розподіл часу обслуговування
      }

      return FRepairTime;
   }

   // Поточний час обслуговування
   public double getTime() {
      if (FGaussTime != null) {
         FRepairTime = FGaussTime.getTime();
      } else if (FEmpiricTime != null) {
         FRepairTime = FEmpiricTime.getTime();
      }
      return FRepairTime;
   }
}
```

TTransportModule.java

```
// Транспортний модуль
public class TTransportModule extends TModule {
   private String FServZone; // Зона обслуговування
   private TRepairModule FTargetModule; // Модуль до якого їде транспортний модуль
   private int FNextPosition; // Номер наступної позиції
   private int FNextModule; // Номер вільного модуля наступної позиції
   private double FMovingTime; // Час переміщення вагона до вільного модуля наступної позиції

   // Конструктор
   public TTransportModule(String
```

```
ServZone, double Failure, double RecoverMean, double RecoverDeviation) {
        super(Failure, RecoverMean, RecoverDeviation);
        super.Reset();
        FServZone = ServZone;
        FTargetModule = null;
    }

    // Час переміщення до вільного модуля наступної позиції
    public double getMovingTime() {
        return FMovingTime;
    }

    public void setMovingTime(double value) {
        FMovingTime = value;
    }

    // Куди всюди вагон (номер наступної позиції)
    public int getNextPosition() {
        return FNextPosition;
    }

    public void setNextPosition(int value) {
        FNextPosition = value;
    }

    // Куди везти вагон (номер вільного модуля наступної позиції)
    public int getNextModule() {
        return FNextModule;
    }

    public void setNextModule(int value) {
        FNextModule = value;
    }

    // Куди везти вагон (Модуль призначення)
    public TRepairModule getTargetModule() {
        return FTargetModule;
    }

    public void setTargetModule(TRepairModule value) {
        FTargetModule = value;
    }
}
```

fileRead.java

```
public class fileRead {

    public String[] readObjDate(String nameFile) {
        try (BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(nameFile), StandardCharsets.UTF_8))) {
            String fileContent = "";
            String sub;
            String[] words = new String[0];
            while ((sub = br.readLine()) != null) {
                fileContent = String.format("%s%s\n", fileContent, sub);
                words = fileContent.split(" ");
            }
            return words;
        } catch (IOException e) {
            e.printStackTrace();
        }
        return new String[0];
    }

    public ArrayList<Integer> readPosData(String nameFile) {
        String fileContent = "";
        String[] words;
```

```
        String sub;
        int rows = 0;
        ArrayList<Integer> arrays = new
ArrayList<>();
        int[] array = null;

        File file = new File(nameFile);
        Scanner scanner = null;
        try {
            scanner = new Scanner(file);
            while (scanner.hasNextInt()) {
                rows++;

arrays.add(scanner.nextInt());
            }
            scanner.close();
            return arrays;
        } catch (FileNotFoundException
e) {
            throw new
RuntimeException(e);
        }
    }

    public ArrayList<Integer>
readProgData(String nameFile) {
        ArrayList<Integer> arrays = new
ArrayList<>();
        File file = new File(nameFile);
        Scanner scanner = null;
        try {
            scanner = new Scanner(file);
            while (scanner.hasNextInt()) {

arrays.add(scanner.nextInt());
            }
            scanner.close();
            return arrays;
        } catch (FileNotFoundException
e) {
            throw new
RuntimeException(e);
        }
    }

    public ArrayList<Integer>
readRepRecoveryData(String
nameFile) {
        ArrayList<Integer> arrays = new
ArrayList<>();
        File file = new File(nameFile);
        Scanner scanner = null;
        try {
            scanner = new Scanner(file);
            while (scanner.hasNextInt()) {

arrays.add(scanner.nextInt());
            }
            scanner.close();
            return arrays;
        } catch (FileNotFoundException
e) {
            throw new
RuntimeException(e);
        }
    }

    public ArrayList<Double>
readTimeData(String nameFile) {
        int rows = 0;
        ArrayList<Double> arrays = new
ArrayList<>();
        File file = new File(nameFile);
        Scanner scanner = null;
        try {
            scanner = new Scanner(file);
            while
(scanner.hasNextDouble()) {

arrays.add(scanner.nextDouble());
            }
            System.out.println();
            scanner.close();
            return arrays;
        } catch (FileNotFoundException
e) {
            throw new
RuntimeException(e);
```

```
    }
  }

  public ArrayList<Integer>
readTransData(String nameFile) {
    ArrayList<Integer> arrays = new
ArrayList<>();
    File file = new File(nameFile);
    Scanner scanner = null;
    try {
      scanner = new Scanner(file);
      while (scanner.hasNextInt()) {

arrays.add(scanner.nextInt());
      }
      scanner.close();
      return arrays;
    } catch (FileNotFoundException
e) {
      throw new
RuntimeException(e);
    }
  }

  public ArrayList<Integer>
readWorkTimeData(String nameFile)
{
    ArrayList<Integer> arrays = new
ArrayList<>();
    File file = new File(nameFile);
    Scanner scanner = null;
    try {
      scanner = new Scanner(file);
      while (scanner.hasNextInt()) {

arrays.add(scanner.nextInt());
      }
      scanner.close();
      return arrays;
    } catch (FileNotFoundException
e) {
      throw new
RuntimeException(e);
    }
  }

  public ArrayList<Double>
readBogiesRecoveryData(String
nameFile) {
    int rows = 0;
    ArrayList<Double> arrays = new
ArrayList<>();
    File file = new File(nameFile);
    Scanner scanner = null;
    try {
      scanner = new Scanner(file);
      while
(scanner.hasNextDouble()) {

arrays.add(scanner.nextDouble());
      }
      System.out.println();
      scanner.close();
      return arrays;
    } catch (FileNotFoundException
e) {
      throw new
RuntimeException(e);
    }
  }

  public static int linesInFile(String
fileName) {
    int lines = 0;
    try (BufferedReader reader = new
BufferedReader(new
FileReader(fileName))) {
      String line;
      while ((line =
reader.readLine()) != null)
        lines++;
      return lines;
    } catch (IOException e) {
      throw new
RuntimeException(e);
    }
  }
  public static int countInFile(String
```

```
fileName) {
        File file = new File(fileName);
        Scanner scanner = null;
        try {
            scanner = new Scanner(file);
            int words = 0;
            while  (scanner.hasNextLine())
{
                String[]        array        =
scanner.nextLine().split(" ");
                words       =       words       +
array.length;
            }
            return words;
        }  catch  (FileNotFoundException
e) {
            throw                        new
RuntimeException(e);
        }
    }
}
```

numberOfRepairGroups.java

```
public class numberOfRepairGroups {
    private int group;
    private String wagonType;
    private String typeOfWork;
    private int normIsSimple;
    private boolean priority;

    public numberOfRepairGroups(int
group, String wagonType, String
typeOfWork,  int  normIsSimple,
boolean priority) {
        this.group = group;
        this.wagonType = wagonType;
        this.typeOfWork = typeOfWork;
        this.normIsSimple        =
normIsSimple;
        this.priority = priority;
    }

    public int getGroup() {
        return group;
    }

    public void setGroup(int group) {
        this.group = group;
    }

    public String getWagonType() {
        return wagonType;
    }

    public  void  setWagonType(String
wagonType) {
        this.wagonType = wagonType;
    }

    public String getTypeOfWork() {
        return typeOfWork;
    }

    public  void  setTypeOfWork(String
typeOfWork) {
        this.typeOfWork = typeOfWork;
    }

    public int getNormIsSimple() {
        return normIsSimple;
    }

    public  void  setNormIsSimple(int
normIsSimple) {
        this.normIsSimple        =
normIsSimple;
    }

    public boolean isPriority() {
        return priority;
    }

    public  void  setPriority(boolean
priority) {
        this.priority = priority;
    }
}
```

numberOfTransportModules.java

```
public                              class
```

```
numberOfTransportModules {
   private int moduleNumber;
   private int servicePosition;
   private double
confidenceOfPossession;

   public
numberOfTransportModules(int
moduleNumber, int servicePosition,
double confidenceOfPossession) {
      this.moduleNumber =
moduleNumber;
      this.servicePosition =
servicePosition;
      this.confidenceOfPossession =
confidenceOfPossession;
   }

   public int getModuleNumber() {
      return moduleNumber;
   }

   public void setModuleNumber(int
moduleNumber) {
      this.moduleNumber =
moduleNumber;
   }

   public int getServicePosition() {
      return servicePosition;
   }

   public void setServicePosition(int
servicePosition) {
      this.servicePosition =
servicePosition;
   }

   public double
getConfidenceOfPossession() {
      return confidenceOfPossession;
   }

   public void
setConfidenceOfPossession(int
confidenceOfPossession) {
      this.confidenceOfPossession =
confidenceOfPossession;
   }
}
```

repairPositions.java

```
public class repairPositions {
   private int positionNumber;
   private int numberOfModules;
   private int movingHour;
   private double
confidenceOfPossession;

   public repairPositions() {
      this.positionNumber = 0;
      this.numberOfModules = 0;
      this.movingHour = 0;
      this.confidenceOfPossession =
0.0;
   }

   public repairPositions(int
positionNumber, int
numberOfModules, int movingHour,
double confidenceOfPossession) {
      this.positionNumber =
positionNumber;
      this.numberOfModules =
numberOfModules;
      this.movingHour = movingHour;
      this.confidenceOfPossession =
confidenceOfPossession;
   }

   public int getPositionNumber() {
      return positionNumber;
   }

   public int getNumberOfModules() {
      return numberOfModules;
   }

   public int getMovingHour() {
```

```
        return movingHour;
    }

    public double getConfidenceOfPossession() {
        return confidenceOfPossession;
    }

    public void setPositionNumber(int positionNumber) {
        this.positionNumber = positionNumber;
    }

    public void setNumberOfModules(int numberOfModules) {
        this.numberOfModules = numberOfModules;
    }

    public void setMovingHour(int movingHour) {
        this.movingHour = movingHour;
    }

    public void setConfidenceOfPossession(int confidenceOfPossession) {
        this.confidenceOfPossession = confidenceOfPossession;
    }
}
```

repairProgram.java

```
public class repairProgram {
    private String entranceNumber;
    private String exitNumber;
    private String typeWagon;
    private String typeOfRepair;
    private String repairTime;
    private String downtime;
    private String timeInShop;

    public repairProgram(String entranceNumber, String exitNumber, String typeWagon, String typeOfRepair, String repairTime, String downtime, String timeInShop) {
        this.entranceNumber = entranceNumber;
        this.exitNumber = exitNumber;
        this.typeWagon = typeWagon;
        this.typeOfRepair = typeOfRepair;
        this.repairTime = repairTime;
        this.downtime = downtime;
        this.timeInShop = timeInShop;
    }

    public String getEntranceNumber() {
        return entranceNumber;
    }

    public void setEntranceNumber(String entranceNumber) {
        this.entranceNumber = entranceNumber;
    }

    public String getExitNumber() {
        return exitNumber;
    }

    public void setExitNumber(String exitNumber) {
        this.exitNumber = exitNumber;
    }

    public String getTypeWagon() {
        return typeWagon;
    }

    public void setTypeWagon(String typeWagon) {
        this.typeWagon = typeWagon;
    }
```

```
    public String getTypeOfRepair() {
        return typeOfRepair;
    }

    public void setTypeOfRepair(String
typeOfRepair) {
        this.typeOfRepair =
typeOfRepair;
    }

    public String getRepairTime() {
        return repairTime;
    }

    public void setRepairTime(String
repairTime) {
        this.repairTime = repairTime;
    }

    public String getDowntime() {
        return downtime;
    }

    public void setDowntime(String
downtime) {
        this.downtime = downtime;
    }

    public String getTimeInShop() {
        return timeInShop;
    }

    public void setTimeInShop(String
timeInShop) {
        this.timeInShop = timeInShop;
    }
}
```

volumeOfRepairsPerYear.java

```
public class volumeOfRepairsPerYear
{
    private int group;
    private String wagonType;
    private String typeOfWork;
    private int numberOfWagons;
    private double interest;

    public
volumeOfRepairsPerYear(int group,
String wagonType, String
typeOfWork, int numberOfWagons,
double interest) {
        this.group = group;
        this.wagonType = wagonType;
        this.typeOfWork = typeOfWork;
        this.numberOfWagons =
numberOfWagons;
        this.interest = interest;
    }

    public int getGroup() {
        return group;
    }

    public void setGroup(int group) {
        this.group = group;
    }

    public String getWagonType() {
        return wagonType;
    }

    public void setWagonType(String
wagonType) {
        this.wagonType = wagonType;
    }

    public String getTypeOfWork() {
        return typeOfWork;
    }

    public void setTypeOfWork(String
typeOfWork) {
        this.typeOfWork = typeOfWork;
    }

    public int getNumberOfWagons() {
        return numberOfWagons;
```

```
    }

    public void setNumberOfWagons(int numberOfWagons) {
        this.numberOfWagons = numberOfWagons;
    }

    public double getInterest() {
        return interest;
    }

    public void setInterest(double interest) {
        this.interest = interest;
    }
}
```

resultController.java

```
public class resultController {
    public AnchorPane anchorPane0;
    public AnchorPane anchorPane1;
    public Button btnFlowStructure;
    public Button btnRepairProgram;
    public Button btnForward;
    public ObservableList<repairProgram> listRepairProgram = FXCollections.observableArrayList();
    public  TableView<repairProgram> tableRepairProgram;
    public TableColumn<repairProgram, String> columnRepairProgram1;
    public TableColumn<repairProgram, String> columnRepairProgram2;
    public TableColumn<repairProgram, String> columnRepairProgram3;
    public TableColumn<repairProgram, String> columnRepairProgram4;
    public TableColumn<repairProgram, String> columnRepairProgram5;
    public TableColumn<repairProgram, String> columnRepairProgram6;
    public TableColumn<repairProgram, String> columnRepairProgram7;

    @FXML
    private void initialize() {
        init();
    }

    private void initTableRepairProgram(String[] words) {

        columnRepairProgram1.setCellValueFactory(new PropertyValueFactory<repairProgram, String>("entranceNumber"));

        columnRepairProgram2.setCellValueFactory(new PropertyValueFactory<repairProgram, String>("exitNumber"));

        columnRepairProgram3.setCellValueFactory(new PropertyValueFactory<repairProgram, String>("typeWagon"));

        columnRepairProgram4.setCellValueFactory(new PropertyValueFactory<repairProgram, String>("typeOfRepair"));

        columnRepairProgram5.setCellValueFactory(new PropertyValueFactory<repairProgram, String>("repairTime"));

        columnRepairProgram6.setCellValue
```

```
Factory(new
PropertyValueFactory<repairProgram
, String>("downtime"));

columnRepairProgram7.setCellValue
Factory(new
PropertyValueFactory<repairProgram
, String>("timeInShop"));

      for (int i = 0; i < words.length;
i+=7) {
         listRepairProgram.add(new
repairProgram(words[i], words[i+1],
words[i+2], words[i+3], words[i+4],
words[i+5], words[i+6]));
      }


tableRepairProgram.setItems(listRepa
irProgram);
   }

   private void init() {
      fileRead file = new fileRead();
      String[] words;
      words =
file.readObjDate("D:/IdeaProjects/dip
loma/Conveyers_Model/src/main/reso
urces/data/result_.txt");
      initTableRepairProgram(words);
   }

   private void
initDefaultVisiblePane() {
      anchorPane0.setVisible(false);
      anchorPane1.setVisible(false);
   }

   private void
initDefaultBackgroundButton() {
      btnFlowStructure.setStyle("-fx-
background-color: #3C3F41;");
      btnRepairProgram.setStyle("-fx-
background-color: #3C3F41;");
   }

   public void
actionRepairProgram(ActionEvent
actionEvent) {
      initDefaultVisiblePane();
      initDefaultBackgroundButton();
      btnRepairProgram.setStyle("-fx-
background-color: #4B6EAF;");
      anchorPane1.setVisible(true);
   }

   public void
actionFlowStructure(ActionEvent
actionEvent) {
      initDefaultVisiblePane();
      initDefaultBackgroundButton();
      btnFlowStructure.setStyle("-fx-
background-color: #4B6EAF;");
      anchorPane0.setVisible(true);
   }
}
```

MainController.java

```
public class MainController {

   public Button btnFlowStructure;
   public Button btnRepairProgram;
   public Button
btnCharacteristicsOfTemporaryModel
s;
   public Button
btnEquipmentRecoveryTimes;
   public Button btnCreateModel;
   public Spinner<Integer>
spinnerRepairPositions;
   public Spinner<Integer>
spinnerNumberOfTransportModules;
   public CheckBox
checkBoxAreaOfPaintingWagons;
   public CheckBox
checkBoxWagonPreparationSection;
   public CheckBox
checkBoxCarRepairArea;
   public Label
```

```
labelNumberOfTransportModules;
    public Label labelRepairPositions;
    public TableView
tableWagonPreparationSection;
    public TableView
tableAreaOfPaintingWagons;
    public AnchorPane anchorPane0;
    public AnchorPane anchorPane1;
    public AnchorPane anchorPane2;
    public AnchorPane anchorPane3;
    public AnchorPane anchorPane4;
    public Spinner<Integer>
spinnerModelingStep;
    public Button btnForward;
    public Button
btnAddRepairPositions;
    public Button
btnAddNumberOfTransportModules;
    public Button
btnDeleteRepairPositions;
    public Button
btnDeleteNumberOfTransportModule
s;
    public Button btnOpenModel;
    private
SpinnerValueFactory<Integer>
valueRepairPositions;
    private
SpinnerValueFactory<Integer>
valueNumberOfTransportModules;
    private
SpinnerValueFactory<Integer>
valueSpinnerModelingStep;
    private
SpinnerValueFactory<Integer>
valueSpinnerVolumeOfRepairsPerYe
ar;
    private
SpinnerValueFactory<Integer>
valueSpinnerNumberOfRepairGroups
;
    public
TableView<numberOfTransportModu
les>
tableNumberOfTransportModules;
    public
TableColumn<numberOfTransportM
odules, Integer> moduleNumber;
    public
TableColumn<numberOfTransportM
odules, Integer> servicePosition;
    public
TableColumn<numberOfTransportM
odules, Double>
confidenceOfPossession1;

    public TableView<repairPositions>
tableRepairPositions;
    public
TableColumn<repairPositions,
Integer> positionNumber;
    public
TableColumn<repairPositions,
Integer> numberOfModules;
    public
TableColumn<repairPositions,
Integer> movingHour;
    public
TableColumn<repairPositions,
Double> confidenceOfPossession;

    public
TableView<numberOfRepairGroups>
tableNumberOfRepairGroups;
    public
TableColumn<volumeOfRepairsPerY
ear, Integer> groupColumn;
    public
TableColumn<volumeOfRepairsPerY
ear, String> wagonTypeColumn;
    public
TableColumn<volumeOfRepairsPerY
ear, String> typeOfWorkColumn;
    public
TableColumn<numberOfRepairGroup
s, Integer> normIsSimpleColumn;
    public
TableColumn<numberOfRepairGroup
```

```
s, Boolean> priority;

    public
TableView<volumeOfRepairsPerYear
> tableVolumeOfRepairsPerYear;
    public
TableColumn<volumeOfRepairsPerY
ear, Integer> groupColumn1;
    public
TableColumn<volumeOfRepairsPerY
ear, String> wagonTypeColumn1;
    public
TableColumn<volumeOfRepairsPerY
ear, String> typeOfWorkColumn1;
    public
TableColumn<volumeOfRepairsPerY
ear,                    Integer>
numberOfWagonsColumn;
    public
TableColumn<volumeOfRepairsPerY
ear, Double> interestColumn;
    public        TableView<InputData>
tableDB;
    public        TableView<InputData>
tableTimeData;
    public        TableView<InputData>
tableRepRecoveryData;
    public  ObservableList<InputData>
listRepRecoveryData             =
FXCollections.observableArrayList();

    public
ObservableList<repairPositions>
listRepairPositions             =
FXCollections.observableArrayList();
    public
ObservableList<numberOfTransport
Modules>
listNumberOfTransportModules    =
FXCollections.observableArrayList();
    public
ObservableList<numberOfRepairGro
ups> listNumberOfRepairGroups =
FXCollections.observableArrayList();
    public
ObservableList<volumeOfRepairsPer
Year> listVolumeOfRepairsPerYear =
FXCollections.observableArrayList();

    private int pane = 1;
    private boolean checkButton =
false;
    @FXML
    private void initialize() {
        initCheckBox();
        initVariables();

spinnerRepairPositions.setValueFacto
ry(valueRepairPositions);

spinnerNumberOfTransportModules.s
etValueFactory(valueNumberOfTrans
portModules);

spinnerModelingStep.setValueFactory
(valueSpinnerModelingStep);
    }

    private void initVariables() {
        valueRepairPositions    =    new
SpinnerValueFactory.IntegerSpinner
ValueFactory(1, 10, 6);

valueNumberOfTransportModules    =
new
SpinnerValueFactory.IntegerSpinner
ValueFactory(1, 10, 2);
        valueSpinnerModelingStep    =
new
SpinnerValueFactory.IntegerSpinner
ValueFactory(1, 60, 60);

valueSpinnerVolumeOfRepairsPerYe
ar                =                new
SpinnerValueFactory.IntegerSpinner
ValueFactory(100, 10000, 5000);

valueSpinnerNumberOfRepairGroups
```

```
= new SpinnerValueFactory.IntegerSpinnerValueFactory(1, 100, 1);
    }

    public void initTableRepairPositions() {

positionNumber.setCellValueFactory(new PropertyValueFactory<repairPositions, Integer>("PositionNumber"));

numberOfModules.setCellValueFactory(new PropertyValueFactory<repairPositions, Integer>("numberOfModules"));

movingHour.setCellValueFactory(new PropertyValueFactory<repairPositions, Integer>("movingHour"));

confidenceOfPossession.setCellValueFactory(new PropertyValueFactory<repairPositions, Double>("confidenceOfPossession"));


tableRepairPositions.setItems(listRepairPositions);
    }
    public void initTableNumberOfTransportModules() {


moduleNumber.setCellValueFactory(new PropertyValueFactory<numberOfTransportModules, Integer>("moduleNumber"));

servicePosition.setCellValueFactory(new PropertyValueFactory<numberOfTransportModules, Integer>("servicePosition"));

confidenceOfPossession1.setCellValueFactory(new PropertyValueFactory<numberOfTransportModules, Double>("confidenceOfPossession"));


tableNumberOfTransportModules.setItems(listNumberOfTransportModules);
    }

    public void initTableNumberOfRepairGroups(String[] content) {


groupColumn.setCellValueFactory(new PropertyValueFactory<volumeOfRepairsPerYear, Integer>("group"));

wagonTypeColumn.setCellValueFactory(new PropertyValueFactory<volumeOfRepairsPerYear, String>("wagonType"));

typeOfWorkColumn.setCellValueFactory(new PropertyValueFactory<volumeOfRepairsPerYear, String>("typeOfWork"));

normIsSimpleColumn.setCellValueFactory(new PropertyValueFactory<numberOfRepairGroups, Integer>("normIsSimple"));
```

```
priority.setCellValueFactory(new
PropertyValueFactory<numberOfRep
airGroups, Boolean>("priority"));

      for (int i = 0; i < content.length;
i+=5) {

listNumberOfRepairGroups.add(new
numberOfRepairGroups(Integer.parse
Int(content[i]), content[i+1],
content[i+2],
Integer.parseInt(content[i+3]),
Boolean.parseBoolean(content[i+4])))
;
      }


tableNumberOfRepairGroups.setItem
s(listNumberOfRepairGroups);
   }

   public void
initTableVolumeOfRepairsPerYear(St
ring[] content) {


groupColumn1.setCellValueFactory(n
ew
PropertyValueFactory<volumeOfRep
airsPerYear, Integer>("group"));

wagonTypeColumn1.setCellValueFac
tory(new
PropertyValueFactory<volumeOfRep
airsPerYear, String>("wagonType"));

typeOfWorkColumn1.setCellValueFa
ctory(new
PropertyValueFactory<volumeOfRep
airsPerYear, String>("typeOfWork"));

numberOfWagonsColumn.setCellVal
ueFactory(new
PropertyValueFactory<volumeOfRep
airsPerYear,
Integer>("numberOfWagons"));

interestColumn.setCellValueFactory(
new
PropertyValueFactory<volumeOfRep
airsPerYear, Double>("interest"));

      for (int i = 0; i < content.length;
i+=5) {

listVolumeOfRepairsPerYear.add(ne
w
volumeOfRepairsPerYear(Integer.par
seInt(content[i]), content[i+1],
content[i+2],
Integer.parseInt(content[i+3]),
Double.parseDouble(content[i+4])));
      }


tableVolumeOfRepairsPerYear.setIte
ms(listVolumeOfRepairsPerYear);
   }

   private void initCheckBox() {

checkBoxCarRepairArea.setSelected(t
rue); //true

checkBoxWagonPreparationSection.s
etSelected(false);

checkBoxAreaOfPaintingWagons.set
Selected(false);

handleOptionWagonPreparationSecti
on();

handleOptionAreaOfPaintingWagons(
);
      handleOptionCarRepairArea();
   }
```

```
    public void handleOptionWagonPreparationSection() {
        if (checkBoxWagonPreparationSection.isSelected())
            tableWagonPreparationSection.setDisable(false);
        else
            tableWagonPreparationSection.setDisable(true);
    }
    public void handleOptionAreaOfPaintingWagons() {
        if (checkBoxAreaOfPaintingWagons.isSelected())
            tableAreaOfPaintingWagons.setDisable(false);
        else
            tableAreaOfPaintingWagons.setDisable(true);
    }
    public void handleOptionCarRepairArea() {
        if (checkBoxCarRepairArea.isSelected()) {
            labelRepairPositions.setDisable(false);
            labelNumberOfTransportModules.setDisable(false);
            spinnerRepairPositions.setDisable(false);
            spinnerNumberOfTransportModules.setDisable(false);
            tableRepairPositions.setDisable(false);
            tableNumberOfTransportModules.setDisable(false);
        } else {
            labelRepairPositions.setDisable(true);
            labelNumberOfTransportModules.setDisable(true);
            spinnerRepairPositions.setDisable(true);
            spinnerNumberOfTransportModules.setDisable(true);
            tableRepairPositions.setDisable(true);
            tableNumberOfTransportModules.setDisable(true);
        }
    }

    private void initDefaultBackgroundButton() {
        btnFlowStructure.setStyle("-fx-background-color: #3C3F41;");
        btnRepairProgram.setStyle("-fx-background-color: #3C3F41;");
        btnCharacteristicsOfTemporaryModels.setStyle("-fx-background-color: #3C3F41;");
        btnEquipmentRecoveryTimes.setStyle("-fx-background-color: #3C3F41;");
    }
    private void initDefaultVisiblePane() {
        anchorPane0.setVisible(false);
```

```
        anchorPane1.setVisible(false);
        anchorPane2.setVisible(false);
        anchorPane3.setVisible(false);
        anchorPane4.setVisible(false);
    }

    public void actionFlowStructure() {
        initDefaultBackgroundButton();
        initDefaultVisiblePane();
        btnFlowStructure.setStyle("-fx-
background-color: #4B6EAF;");
        anchorPane1.setVisible(true);
    }
    public void actionRepairProgram()
{
        initDefaultBackgroundButton();
        initDefaultVisiblePane();
        btnRepairProgram.setStyle("-fx-
background-color: #4B6EAF;");
        anchorPane2.setVisible(true);
    }
    public                          void
actionCharacteristicsOfTemporaryMo
dels() {
        initDefaultBackgroundButton();
        initDefaultVisiblePane();

btnCharacteristicsOfTemporaryModel
s.setStyle("-fx-background-color:
#4B6EAF;");
        anchorPane3.setVisible(true);
    }
    public                          void
actionEquipmentRecoveryTimes() {
        initDefaultBackgroundButton();
        initDefaultVisiblePane();

btnEquipmentRecoveryTimes.setStyle
("-fx-background-color: #4B6EAF;");
        //anchorPane4.toFront();
        anchorPane4.setVisible(true);
    }

    private void renameButton() {
btnForward.setText("Моделювання")
;
    }

    public                          void
actionForwardButton(ActionEvent
actionEvent) {

        switch (pane) {
            case 2: {
                actionRepairProgram();
                pane++;
                break;
            }
            case 3: {

actionCharacteristicsOfTemporaryMo
dels();
                pane++;
                break;
            }
            case 4: {

actionEquipmentRecoveryTimes();
                pane++;
                renameButton();
                checkButton = true;
                break;
            }
            case 5: {
                break;
            }
            default: {
                break;
            }
        }
        if (checkButton) {
            showAsDialog("result");
        }

    }
```

```
   public void handleButtonClick(ActionEvent actionEvent) {
      if (actionEvent.getSource() == btnAddRepairPositions)

showAsDialog("addRepairPositions");
      if (actionEvent.getSource() == btnAddNumberOfTransportModules)

showAsDialog("addNumberOfTransportModules");
   }

   private void showAsDialog(String fxml) {
      try {
         Parent parent = FXMLLoader.load(getClass().getResource( "/fxml/" + fxml + ".fxml")); // getClass().getResource( "/fxml/" + fxml + ".fxml")
         Stage stage = new Stage();
         Scene scene = new Scene(parent);
         stage.setScene(scene);
         stage.show();
      } catch (IOException e) {
         throw new RuntimeException(e);
      }
   }

   public void handleDeleteButtonClick(ActionEvent actionEvent) {
      if (actionEvent.getSource() == btnDeleteRepairPositions) {
         int id = tableRepairPositions.getSelectionModel().getSelectedIndex();

tableRepairPositions.getItems().remove(id);
         //System.out.println(id);
      }
      if (actionEvent.getSource() == btnDeleteNumberOfTransportModules) {
         int id = tableNumberOfTransportModules.getSelectionModel().getSelectedIndex();

tableNumberOfTransportModules.getItems().remove(id);
         //System.out.println(id);
      }
   }

   public void addTableInt(ArrayList<Integer> array, int a) {
      switch (a) {
         case 1: {
            for (int i = 0; i < array.size(); i+=4) {

listRepairPositions.add(new repairPositions((int) array.get(i), (int) array.get(i+1), (int) array.get(i+2), (double) array.get(i+3)));
            }
            initTableRepairPositions();
            break;
         }
         case 2: {
            for (int i = 0; i < array.size(); i+=3) {

listNumberOfTransportModules.add(new numberOfTransportModules((int) array.get(i), (int) array.get(i+1), (double) array.get(i+2)));
            }

initTableNumberOfTransportModules();
```

```
            break;
        }
    }
}

public void addTable(repairPositions repairPositions) {

listRepairPositions.add(repairPositions);

System.out.println(listRepairPositions.get(0));
}

private void addTableString(String[] content) {
    for (int i = 0; i <= content.length; i+=6) {

listNumberOfRepairGroups.add(new numberOfRepairGroups(Integer.parseInt(content[i]), content[i+1], content[i+2], Integer.parseInt(content[i+3]), Boolean.parseBoolean(content[i+4])));
    }
    for (int i = 0; i <= content.length; i++) {
        System.out.println("----------");
        System.out.println(content[i]);
    }
}

private int count(ArrayList<?> array) {
    int count = 0;
    for (var i: array) {
        count++;
    }
    return count;
}
private void createColn(ArrayList<Double> array, int colsInpDate, int a) {
    switch (a) {
        case 1: {
            for (int j = 0; j <= colsInpDate - 1; j++) {
                final int indexColumn = j;
                TableColumn<InputData, String> tableColumn = new TableColumn<>((j % 2 == 0) ? "M.O." : "C.K.O.");

tableColumn.setCellValueFactory(param -> new SimpleStringProperty(param.getValue().getItems().get(indexColumn)));

tableTimeData.getColumns().add(tableColumn);
            }

            break;
        }
        case 2: {
            for (int j = 0; j <= colsInpDate - 1; j++) {
                final int indexColumn = j;
                TableColumn<InputData, String> tableColumn = new TableColumn<>((j % 2 == 0) ? "M.O." : "C.K.O.");

tableColumn.setCellValueFactory(param -> new SimpleStringProperty(param.getValue().getItems().get(indexColumn)));

tableRepRecoveryData.getColumns().add(tableColumn);
            }
            break;
        }
```

```
        }
    }

    private void setDoubleFile(ArrayList<Double> array, int b) {
        String[] fields = new String[array.size()];
        int i = 0;
        for (var a: array) {
            fields[i] = String.valueOf(a);
            if (i == array.size()/b)

            i++;
            //System.out.println(a);
        }
    }

    public void actionOpenModel(ActionEvent actionEvent) {
        fileRead file = new fileRead();
        String[] words;
        words = file.readObjDate("D:/IdeaProjects/diploma/Conveyers_Model/src/main/resources/data/ObjData.txt");

        initTableNumberOfRepairGroups(words);
        words = file.readObjDate("D:/IdeaProjects/diploma/Conveyers_Model/src/main/resources/data/ProgData.txt");

        initTableVolumeOfRepairsPerYear(words);
        ArrayList<Integer> array = new ArrayList<>();
        array = file.readPosData("D:/IdeaProjects/diploma/Conveyers_Model/src/main/resources/data/PosData.txt"); // return ArrayList<Integer>
        addTableInt(array, 1);
        array.clear();
        ArrayList<Integer> array1;
        array1 = file.readTransData("D:/IdeaProjects/diploma/Conveyers_Model/src/main/resources/data/TransData.txt");
        addTableInt(array1, 2);
        String name = "D:/IdeaProjects/diploma/Conveyers_Model/src/main/resources/data/TimeData.txt";
        int countColumn = fileRead.countInFile("D:/IdeaProjects/diploma/Conveyers_Model/src/main/resources/data/TimeData.txt")/3;
        ArrayList<Double> array2;
        array2 = file.readBogiesRecoveryData("D:/IdeaProjects/diploma/Conveyers_Model/src/main/resources/data/TimeData.txt");

        System.out.println(countColumn);
        setDoubleFile(array2, 3);
        actionFlowStructure();
        pane++;
    }
}
```