Одеський національний політехнічний університет

Інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Пояснювальна записка

до Розрахунково-графічної роботи

з дисципліни «Проектний практикум»

на тему «Редактор діаграм UML із асоціативною пам’яттю»

Виконав: ст. гр. АС-123

Малярозов П.О.

Перевірив:

доц. Блажко О. А.

Одеса – 2016 року

ЗМІСТ

[**1.** **ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ** 3](#_Toc449653097)

[**1.1** **Визначення бізнес-вимог** 3](#_Toc449653098)

[**1.2** **Функціональні вимоги** 4](#_Toc449653099)

[**1.4 Планування розробки** 8](#_Toc449653100)

[**2.** **ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ** 17](#_Toc449653101)

[**2.1** **Концептуальне проектування** 17](#_Toc449653102)

[**2.2 Логічне проектування** 17](#_Toc449653103)

[**2.2.1 Опис структур даних** 18](#_Toc449653104)

[**2.2.2 Алгоритми** 20](#_Toc449653105)

[**2.2.3 Інтерфейс користувача** 21](#_Toc449653106)

[**3.** **КОНСТУЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ** 24](#_Toc449653107)

[**3.1** **Опис програмних технологій** 24](#_Toc449653108)

[**3.2** **Опис програмних бібліотек** 24](#_Toc449653109)

[**3.3** **Особливості створення програмних модулів с урахуванням мови програмування** 24](#_Toc449653110)

[**3.4** **Особливості створення структур даних** 24](#_Toc449653111)

[**3.5** **Модульне тестування** 25](#_Toc449653112)

[Нижче представлено схему алгоритмів для функції Convert to Graph(рисунок 3.1): 25](#_Toc449653113)

[**4.** **ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ** 29](#_Toc449653114)

[**4.1** **Функціональне тестування** 29](#_Toc449653115)

[**5.** **РОЗГОРТАННЯ ПРОГРАМНОГО ПРОДУКТУ** 31](#_Toc449653116)

[**5.1 Інструкція з інсталяції** 31](#_Toc449653117)

1. **ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ**
   1. **Визначення бізнес-вимог**

Предметна область: комплексна система тестування програмного забезпечення.

Суб’єкти автоматизації: модулі комплексної системи тестування програмного забезпечення.

Представлення деталізованого об’єктів та суб’єктів у графічному вигляді(рисунок 1.1):

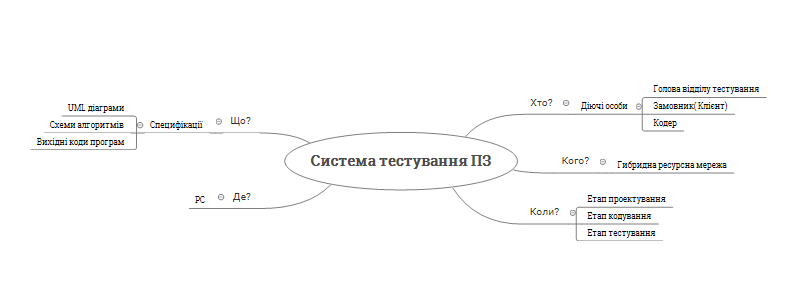


Рисунок 1.1 – ментальна карта програмного продукту

Визначення проблем та суб’єктів автоматизації при взаємодії з об’єктом:

При проведенні процесу тестування програмного забезпечення діючі продукти орієнтовані на програмний код, а наша система комплексного тестування програмного забезпечення дозволяє проводити тестування вже на стадіях проектування програмного продукту. Діючі аналоги не мають потрібного функціоналу, такого як переведення специфікацій вимог у документ формату XML для створення графу специфікацій вимог.

Проблемний аналіз існуючих програмних продуктів:

Таблиця 1.1 – Аналіз існуючих програмних продуктів

|  |  |  |  |
| --- | --- | --- | --- |
| Параметри | this.UML | StarUML | Visio |
| Створення діаграмм UML | + | + | + |
| Збереження файлу у файл формату XML | + | - | - |
| Конвертація діаграм UML у граф специфікацій вимог | + | - | - |
| Платний продукт | - | +/-(умовно безкоштовний) | - |
| Зручність інтерфейсу користувача | + | + | + |
| Популярність | - | + | + |

+ Так

* Ні

Визначення цілей , яких повинен досягати з урахуванням критеріїв, які показують економічний або інший ефект:

Метою являється створення системи - редактору діаграм UML(модулю системи комплексного тестування ПЗ), яка допоможе скоротити час на опис вимог. Система повинна створювати граф специфікацій вимог для подальшого використання іншими модулями.

Визначення унікальної назви програмного продукту, який включає описання об’єкту, досяжності цілей або вирішення проблем:

Назва програмного продукту – this.UML.

Нижче представлено логотип програмного продукту(рисунок 1.2):

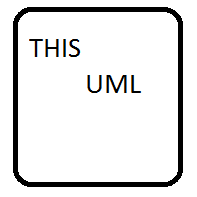


Рисунок 1.2 – логотип програмного продукту

Логотип відображає у собі назву продукту на білому фоні. Квадратний щит з округленими кутами обрамлений чорним кольором.

* 1. **Функціональні вимоги**

Опис діаграми прецедентів:

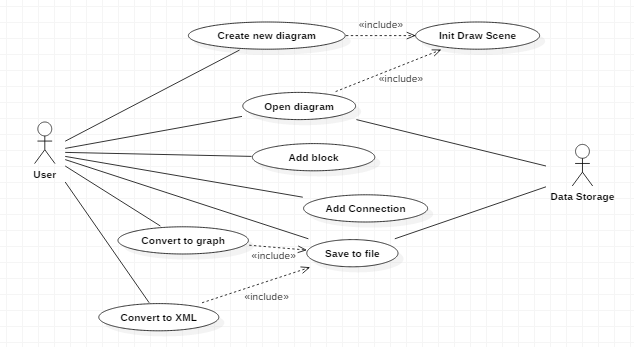


Рисунок 1.3 – Діаграма варіантів використання

Нижче представлено опис прецедентів:

**1.2.1 Прецедент: Create new diagram**

Передумова: Користувач запустив редактор

Сценарій:

1. Користувач вибирає опцію зі створення нової діаграми

2. Система відображає діалогове вікно з вибором назви діаграми та її розмірами.

3. Користувач вносить всі дані.

4. Система відображає діалогове вікно з вибором місця збереження файлу.

5. Користувач вибирає потрібне місце.

6. Система підтверджує успішне створення діаграми.

7. Запуск ВВ Init Draw Scene.

**1.2.2 Прецедент: Open Diagram**

Передумова: Користувач запустив редактор

1. Користувач вибирає опцію з відкриття існуючої діаграми.

2. Система відображає діалогове вікно з вибору місця розташування файлу.

3. Користувач вибирає потрібний файл.

4. Система підтверджує успішне завантаження файлу.

5. Запуск ВВ Init Draw Scene.

**1.2.3 Прецедент: Init Draw Scene**

Передумова: Система завантажила файл з діаграмою

1. Система прочитує зміст файлу (тут надо расширение, что файл поврежден)

2. Система відображає графічну сцену редагування та панель інструментів.

3. Система розміщує компоненти, які були прочитані з файлу, на графічну сцену.

**1.2.4 Прецедент: Add Block**

Передумова: Файл з діаграмою відкрито

1. Користувач на панелі інструментів вибирає потрібний вид блоку.

2. Користувач на графічній сцені вибирає місце розташування компоненту.

3. Система відображає блок на графічній сцені.

4. Система відображає активне поле редагування всередині блоку.

5. Користувач заповнює вміст блоку.

6. Система фіксує зміст блоку на графічній сцені.

**1.2.5 Прецедент: Add Connection**

Передумова: Файл з діаграмою відкрито

1. Користувач на панелі інструментів вибирає компонент зв'язку.

2. Користувач на графічній сцені вибирає місце розташування початку стрілки.

3. Система відображає на графічній сцені стрілку, кінець якої закріплений у вибраному користувачем місці.

3а. На місці сцени розташований блок.

3а.1 Система закріплює початок стрілки до блоку та відображає це на графічній сцені.

4. Користувач вибирає місце розтащування кінця стрілки.

5. Система закріплює цінець стрілки на графічній сцені.

5а. На місці сцени розташований блок.

5а.1 Система закріплює цінець стрілки до блоку та відображає це на графічній сцені.

**1.2.6 Прецедент: Save to file**

Передумова: Файл з діаграмою відкрито

1. Користувач вибирає опцію зі збереження файлу.

2. Система відображає діалогове вікно з вибору назви та місця розташування файлу.

3. Користувач заповнює необхідні дані.

4. Система відображає повідомлення про успішне збереження файлу.

**1.2.7 Прецедент: Convert to XML**

Передумова: Файл з діаграмою відкрито

1. Користувач вибирає опцію збереження діаграми у форматі XML.

2. Система виконує переробку діаграми та відображає результат в окремому вікні.

3. Користувач вибирає опцію збереження файлу.

4. Запуск ВВ Save file.

**1.3 Нефункціональні вимоги**

Специфікація вимог користувача( нефункціональних вимог):

Таблиця 1.2 – Нефункціональні вимоги

| Набір характеристик (за ISO 9126) | Властивості програмного забезпечення |
| --- | --- |
| Функціональність | * Здатність до взаємодії (interoperability): Програмний продукт повинен взаємодіяти з іншими модулями системи для обміну даними. * Захищеність (security): Дані, які надає система конкретному модулю, недоступні іншим модулям. |
| Ефективність | * Часові характеристики (time behaviour): Збереження даних повинно бути < 15 секунд; Додавання нового елементу на область редагування < 5 секунд * Використання ресурсів (resource utilisation): Програма не повинна використовувати більше 500 МБ. |
| Надійність | * Здатність до відновлення працездатності при відмовах (recoverability): Програма повинна мати можливість автоматично зберігати дані та у разі збоїв запускати останній збережений варіант діаграми. |
| Переносимість (мобільність) | * Адаптованість (adaptability): Платформою для написання програми буде обрано мову програмування Java, програма буде підтримувати усі актуальні платформи. |

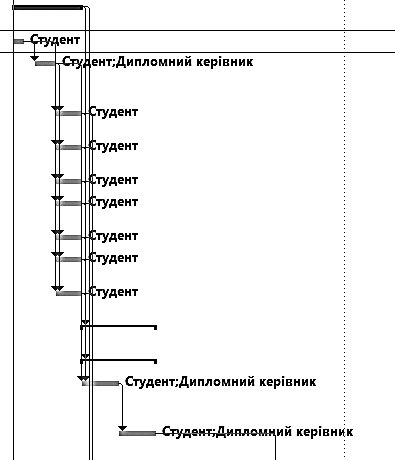
## **1.4 Планування розробки**

Побудування базового шляху проекту виконується на основі цілей проекту. Нижче представлено таблицю базового шляху проекту (таблиця 5), яка враховує робочі години за виключенням вихідних. На модифікацію архітектури ПЗ відведено 9 днів, аби виправити помилки вибору архітектурних стилів та вибрати правильні архітектурні тактики за допомогою методу оцінювання архітектури ATAM. Модифікація архітектури для варіантів використання виконується за 3 дні. На аналіз вимог видається 2 дні, вимоги чітко задані дипломним керівником та потребували незначного уточнення. На організаційні процеси виділено 10 днів. На навчання виділено 10 днів, так як в процес навчання входять розбір технологій, які допомагають пришвидшити процес сам процес розробки, вивчення досвіду інших при вирішенні проблем, пов’язаних с процесом кодування. На процес кодування виділено 25 днів. На документування відведено 3 дні, за цей час йде процес опису роботи системи та визначення технічних вимог до програми. На забезпечення якості виділено 6 днів, так як потрібно перевірити, чи задовольняє система вимогам кінцевого користувача.

Таблиця 1.3 – Базовий шлях проекту

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Назва задачі | Тривалість | Початок | Кінець | Попередник | Назви ресурсів | СДР |
| Модифікація архітетутри ПЗ | 9 днів | Чт 29.10.15 | Вт 10.11.15 |  | Студент | DW1 |
| Аналіз вимог | 2 днів | Чт 29.10.15 | Пт 30.10.15 |  | Студент | DW1.AA |
| Реконструкція програмної архітектури | 4 днів | Пн 02.11.15 | Чт 05.11.15 | 2 | Студент;Дипломний керівник | DW1.AB |
| ВВ: Create new Diagramm | 3 днів | Пт 06.11.15 | Вт 10.11.15 | 3;2 | Студент | DW1.AC |
| ВВ: Open Diagramm | 3 днів | Пт 06.11.15 | Вт 10.11.15 | 3;2 | Студент | DW1.AD |
| ВВ: Add Block | 3 днів | Пт 06.11.15 | Вт 10.11.15 | 3;2 | Студент | DW1.AE |
| ВВ: Add Connection | 3 днів | Пт 06.11.15 | Вт 10.11.15 | 3;2 | Студент | DW1.AG |
| ВВ: Save to File | 3 днів | Пт 06.11.15 | Вт 10.11.15 | 3;2 | Студент | DW1.AH |
| ВВ: Convert to XML | 3 днів | Пт 06.11.15 | Вт 10.11.15 | 3;2 | Студент | DW1.AI |
| ВВ: Init Draw Scene | 3 днів | Пт 06.11.15 | Вт 10.11.15 | 2;3 | Студент | DW1.AJ |
| Організаційні процеси | 10 днів | Ср 11.11.15 | Вт 24.11.15 |  | Студент | DW3 |
| Навчання | 10 днів | Ср 11.11.15 | Вт 24.11.15 | 3 | Студент | DW3.AC |
| Вивчення існуючої реалізації | 5 днів | Ср 11.11.15 | Вт 17.11.15 | 3;6;7;9;4;10;5;8 | Студент;Дипломний керівник | DW3.AC.1 |
| Вивчення архітектури MVC | 5 днів | Ср 18.11.15 | Вт 24.11.15 | 13 | Студент;Дипломний керівник | DW3.AC.2 |
| Розробка | 25 днів | Пт 13.11.15 | Чт 17.12.15 |  |  | DW5 |
| ВВ: Create new Diagramm | 4 днів | Пт 13.11.15 | Ср 18.11.15 | 4 | Студент | DW5.AA |
| ВВ: Open Diagramm | 4 днів | Пт 13.11.15 | Ср 18.11.15 | 5 | Студент | DW5.AK |
| ВВ: Add Block | 4 днів | Пт 13.11.15 | Ср 18.11.15 | 6 | Студент | DW5.AJ |
| ВВ: Add Connection | 4 днів | Пт 13.11.15 | Ср 18.11.15 | 7 | Студент | DW5.AI |
| ВВ: Save to File | 4 днів | Пт 13.11.15 | Ср 18.11.15 | 8 | Студент | DW5.AH |
| ВВ: Convert to XML | 4 днів | Пт 13.11.15 | Ср 18.11.15 | 9 | Студент | DW5.AG |
| ВВ: Init Draw Scene | 4 днів | Пт 13.11.15 | Ср 18.11.15 | 10 | Студент | DW5.AF |
| Допоміжні процеси | 9 днів | Ср 25.11.15 | Пн 07.12.15 | 14 |  | DW6 |
| Документування | 3 днів | Ср 25.11.15 | Пт 27.11.15 | 14;16;17;18;19;20;21;22 | Студент | DW6.AA |
| Забезпечення якості | 6 днів | Пн 30.11.15 | Пн 07.12.15 | 24;16;17;18;19;20;21;22 | Студент | DW6.AB |

Нижче на рисунках 1 та 2 представлені базовий шлях проекту та критичний шлях проекту(критичний шлях виділено жирним).



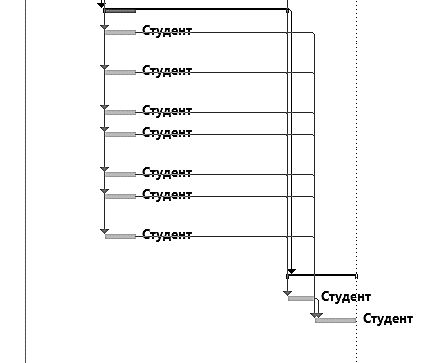


Рисунок 1.4 – Базовий шлях проекту

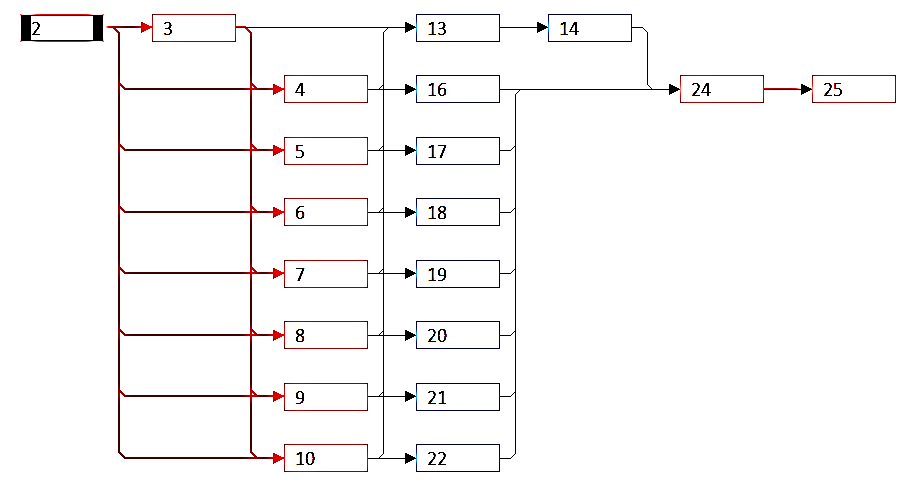


Рисунок 1.5 – Критичний шлях проекту

Таблиця 1.4 - Функціональний аналіз існуючих продуктів:

|  |  |  |  |
| --- | --- | --- | --- |
| Параметри | this.UML | StarUML | Visio |
| Створення діаграмм UML | + | + | + |
| Збереження файлу у файл формату XML | + | - | - |
| Конвертація діаграм UML у граф специфікацій вимог | + | - | - |
| Платний продукт | - | +/-(умовно безкоштовний) | - |
| Зручність інтерфейсу користувача | + | + | + |
| Популярність | - | + | + |
| Вивід документу на друкування | + | + | + |
| Можливість запуску на різних платформах | + | + | +/-(немає в наявності реалізації для Linux) |

+ Так

* Ні
  1. MoSCoW принцип розташування пріорітетів:

M: Збереження файлу у файл формату XML

S: Конвертація діаграм UML у граф специфікацій вимог

C: Можливість запуску на різних платформах

W: Платний продукт

Дана програма дозволяє створювати наступні види діаграм:

Діаграми варіантів використання(рисунок 1.6):

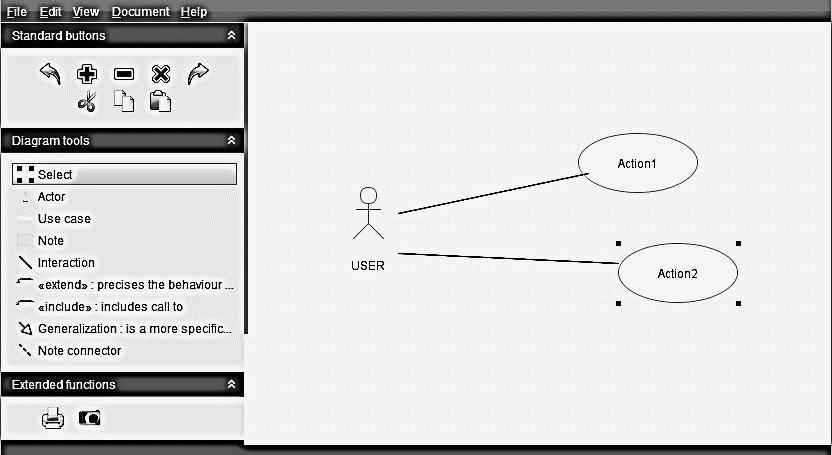


Рисунок 1.6 – Діаграма варіантів використання

Діаграми класів(рисунок 1.7):

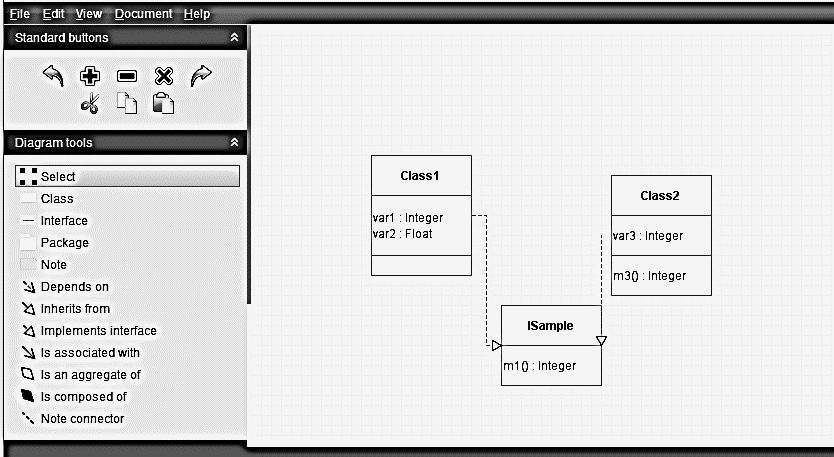


Рисунок 1.7 – Діаграма класів

Діаграми послідовностей(рисунок 1.8):

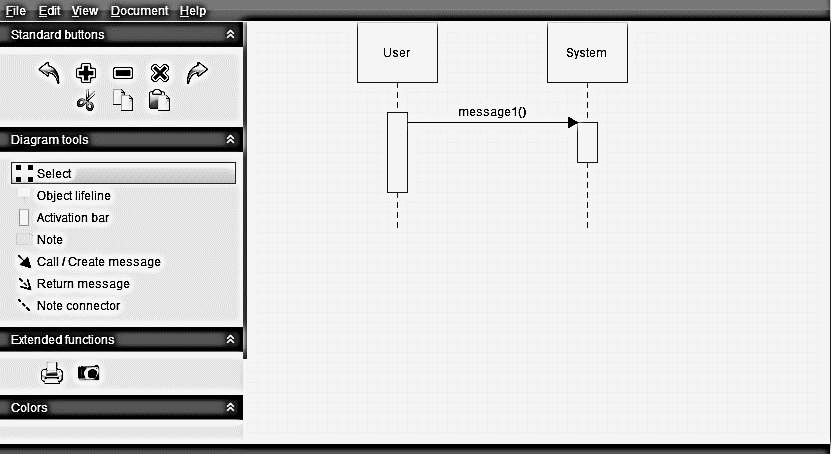


Рисунок 1.8 – Діаграма послідовностей

Діаграма кооперацій(рисунок 1.9):

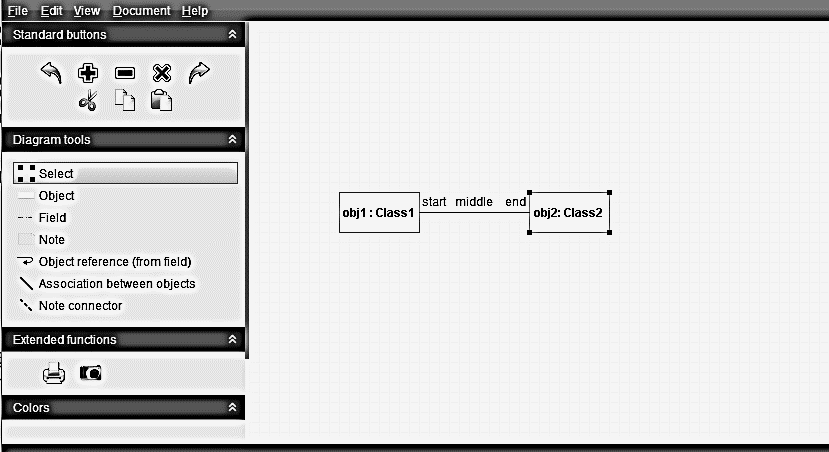


Рисунок 1.9 – Діаграма кооперацій

Атрибути якості:

Таблиця 1.5 – Атрибути якості

| Набір характеристик (за ISO 9126) | Властивості програмного забезпечення |
| --- | --- |
| Функціональність | * Здатність до взаємодії (interoperability): Програмний продукт повинен взаємодіяти з іншими модулями системи для обміну даними. * Захищеність (security): Дані, які надає система конкретному модулю, недоступні іншим модулям. |
| Ефективність | * Часові характеристики (time behaviour): Збереження даних повинно бути < 15 секунд; Додавання нового елементу на область редагування < 5 секунд * Використання ресурсів (resource utilisation): Програма не повинна використовувати більше 500 МБ. |
| Надійність | * Здатність до відновлення працездатності при відмовах (recoverability): Програма повинна мати можливість автоматично зберігати дані та у разі збоїв запускати останній збережений варіант діаграми. |
| Переносимість (мобільність) | * Адаптованість (adaptability): Платформою для написання програми буде обрано мову програмування Java, програма буде підтримувати усі актуальні платформи. |

Архітектура програмної системи:

На рисунку 1.10 зображена діаграма класів системи:

За основу був взятий архітектурний шаблон MVC. У процесі розробки архітектури була виявлена необхідність додати клас, який відповідає за збереження діаграми у файл формату XML.

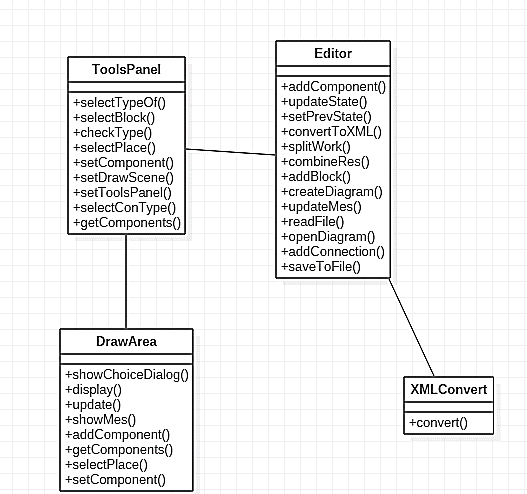


Рисунок 1.10 – Діаграма класів системи

Нижче описані зміни у порівнянні з базовим проектом:

* Додано метод splitWork() та combineRes() до класу Editor згідно застосуванню тактики Введення паралелізму.
* Додано компонент XMLConvert, який містить у собі метод convert(), згідно застосуванню тактики Введення паралелізму.
* До класу Editor додано метод displayErMes() згідно застосування тактики Транзакція.
* До класу Editor додано метод updateState() згідно застосування тактики Транзакція.
* До класу Editor додано метод setPrevSrate() згідно застосування тактики Транзакція.
  1. Технології розробки:
* Операційні системи: Windows, Linux, Mac OS( але обов’язковою умовою являється наявність встановленої Java версії не нижче Java 1.7).
* Структури даних: файли формату XML для діаграм, бінарні файли для збереження діаграм та для переведення діаграм у файл формату XML.
* Інструменти для розробки: IntelliJ IDEA Community Edition 15.0.2 для самого програмного продукту, Sublime Text 3 для створення DTD опису файлу XML, git та github для системи контролю версій.

На рисунку 1.11 зображена діаграма пакетів програмних модулів системи:

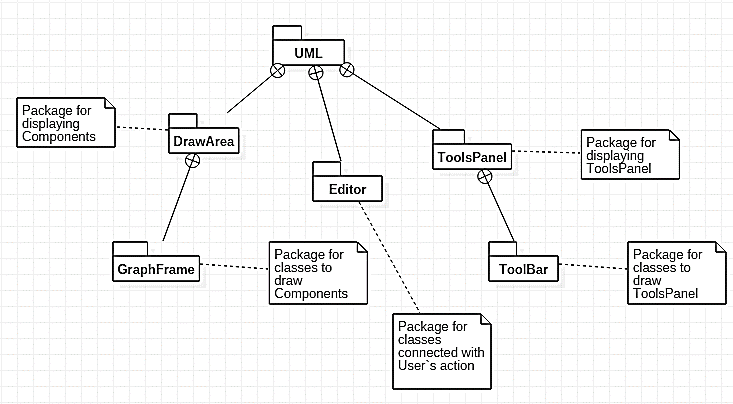


Рисунок 1.11 – Діаграма пакетів програмних модулів системи

1. **ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ**
   1. **Концептуальне проектування**

Нижче представлено концептуальні класи, створені на етапі проектування програмної системи(Рис. 2.1).

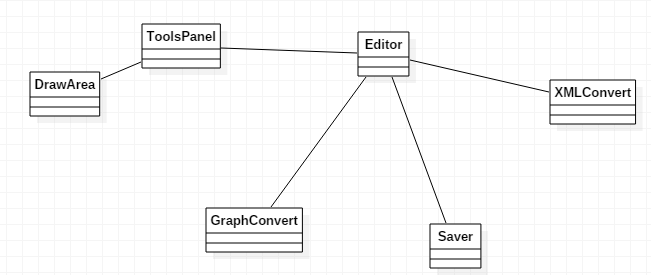


Рисунок 2.1 – діаграма програмних класів.

На етапі проектування діаграми концептуальних класів потрібно вибрати опорні іменники у якості потенційних класів. Було вибрано: ToolsPanel, Editor, XMLConvert, GraphConvert, Saver та DrawArea.

## **2.2 Логічне проектування**

### **2.2.1 Опис структур даних**

На рисунку 2.2 представлена діаграма програмних класів, яка була після проектування програмних класів та застосування тактик з проектування програмного забезпечення.

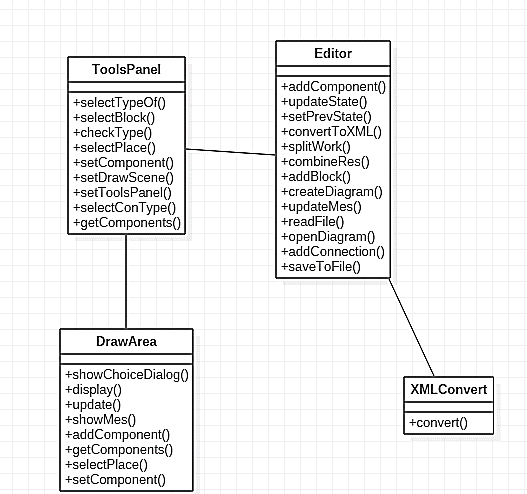


Рисунок 2.2 – діаграма програмних класів

- описание структур данных (оперативных и постоянных в виде XML-файлов или

баз данных).

У програмній системі взаємодія з іншими модулями здійснюється за допомогою файлів формату XML. Нижче приведено опис структури файлу формату XML.

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<!DOCTYPE file [

<!ELEMENT file (class)+>

<!ELEMENT class (field\*, method\*)>

<!ATTLIST class nameC CDATA "">

<!ELEMENT method (input)\*>

<!ATTLIST method nameM CDATA "">

<!ATTLIST method typeM CDATA "">

<!ATTLIST method mod CDATA "">

<!ELEMENT field (#PCDATA)>

<!ATTLIST field nameF CDATA "">

<!ATTLIST field typeF CDATA "">

<!ATTLIST field mod CDATA "">

<!ELEMENT input (#PCDATA)>

<!ATTLIST input nameI CDATA "">

<!ATTLIST input typeI CDATA "">

]>

<file>

<class nameC="Class1">

<field nameF="value1" typeF="int" mod="private">1</field>

<field nameF="value2" typeF="double" mod="private">1.0</field>

<field nameF="value3" typeF="string" mod="private">Str</field>

<method nameM="func1" typeM="void" mod="public">

<input nameI="par1" typeI="int">null</input>

</method>

<method nameM="func2" typeM="int" mod="public">

<input nameI="par2" typeI="int">null</input>

</method>

</class>

<class nameC="Class2">

<field nameF="value1" typeF="int" mod="private">2</field>

<field nameF="value2" typeF="double" mod="private">2.0</field>

<field nameF="value3" typeF="string" mod="private">Str2</field>

<method nameM="func1" typeM="void" mod="public">

<input nameI="par1" typeI="int">null</input>

</method>

<method nameM="func2" typeM="int" mod="public">

<input nameI="par2" typeI="int">null</input>

</method>

</class>

<class nameC="ClassB"></class>

</file>

Нижче представлено опис тегів:

<file> - кореневий тег, означає сам документ.

<class> - означає дані, які наявні у класі, має атрибут nameC, який означає ім’я класу.

<field> - означає дані, які відповідають за поля класу. Має атрибути nameF, який відповідає за ім’я поля, typeF, який відповідає за тип поля, mod, який відповідає за модифікатор доступу поля.

<method> - означає дані, які відповідають за методи класу. Має атрибути nameM, який відповідає за ім’я методу, typeM, який відповідає за тип даних, який повертається методом, mod, який відповідає за модифікатор доступу методу.

<input> - означає дані, які віповідають за вхідні параметри методу. Має атрибути nameI, який відповідає за ім’я вхідного параметру, typeI, який відповідає за тип вхідного параметру.

### **2.2.2 Алгоритми**

Нижче представлена схема алгоритму для функції Convert to Graph(Рис 2.3):



Рисунок 2.3 – схема алгоритму Convert to Graph

При оцінці алгоритмічної складності треба враховувати середній час доступу до даних у пам’яті, наявність циклів під час виконання операцій та час збереження даних. Нижче зображено схему алгоритму для функції Convert to XML(рисунок 2.4):



Рисунок 2.4 – схема алгоритму для функції Convert to XML

У наведених схемах алгоритмів більшість елементарних блоків виконують операції зчитування/запису даних. Воно відбувається за середній час O(n\*log(n)). Також на схемах наявні цикли, складність яких O(n)(середній випадой). У найгіршому разі складність програмної системи: O(n\*n). Середній випадок: O(n\*log(n)).

### **2.2.3 Інтерфейс користувача**

У початку роботи програми користувач бачить пусте вікно зі стрічкою меню. Нова діаграма створюється наступним чином: користувач натискає пункт меню File -> New -> Static View(діаграма класів або діаграма кооперацій/коммунікацій) або Dynamic View( діаграма використання або діаграма послідовностей). Далі користувач додає блоки діаграм на графічну сцену та поєднує їх лініями шляхом перетягування блоків на графічну сцену(метод drag`n`drop). Збереження діаграми відбувається таким чином: користувач натискає пункт меню File -> Save та задає ім’я файлу. Для конвертації у файл формату XML/граф необхідно вибрати пункт меню File -> Save as та вибрати ім’я та формат файлу( XML або граф).для відкриття вже створеної діаграми необхідно вибрати пункт меню File -> Open та вибрати файл с діаграмою.

Дана програма дозволяє створювати наступні види діаграм:

Діаграми варіантів використання(рисунок 2.5):

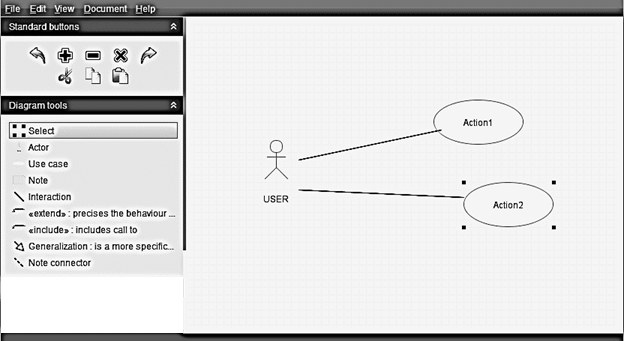


Рисунок 2.5 – Діаграма варіантів використання

Діаграми класів(рисунок 2.6):

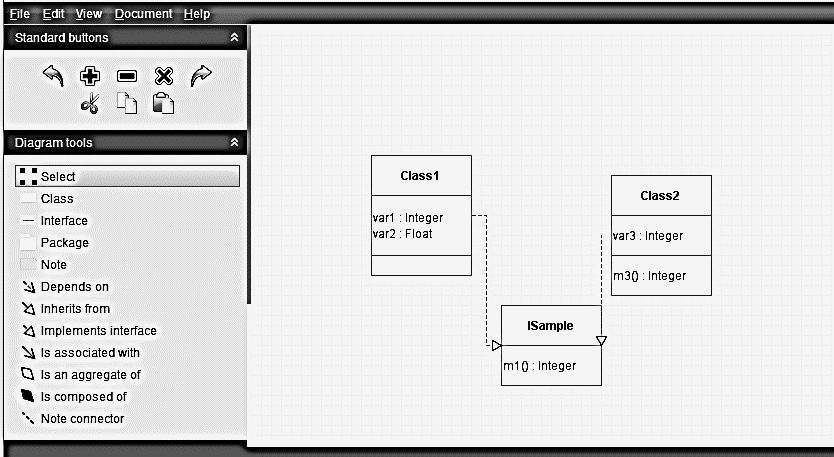


Рисунок 2.6 – Діаграма класів

Діаграми послідовностей(рисунок 2.7):

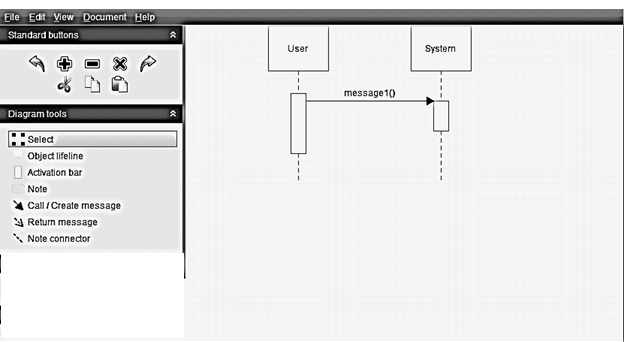


Рисунок 2.7 – Діаграма послідовностей

Діаграма кооперацій(рисунок 2.8):

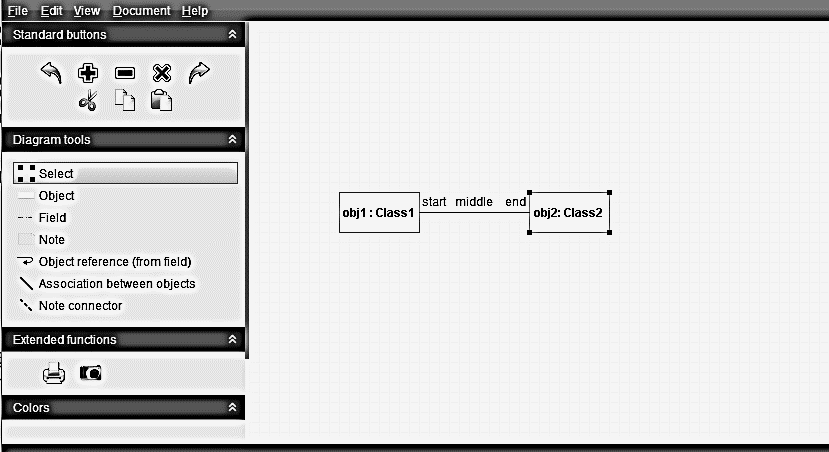


Рисунок 2.8 – Діаграма кооперацій

1. **КОНСТУЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ**

## **Опис програмних технологій**

Для створення програмного продукту були використані програмні технології: Eclipse(Java), IntelliJ IDEA Community Edition 2016.1(64) (Java), NetBeans IDE(Java).

* 1. **Опис програмних бібліотек**

java.io.\* - введення/виведення для дебагу.

java.util.\* - для використання структур данних.

javax.swing.\* - для малювання графічного інтерфейсу.

import org.w3c.dom.\* - для збереження діаграми у файл формату XML.

import javax.xml.parsers.\* - для збереження діаграми у файл формату XML.

* 1. **Особливості створення програмних модулів с урахуванням мови програмування**

В даному проекті кожен програмний модуль представлений окремим файлом у вигляді класу. Зв'язок між даними файлами (класами) забезпечена за допомогою import. Так як Java є повністю об'єктів-орієнтованої, то всюди притаманні класи.

* 1. **Особливості створення структур даних**

Основні структури даних являють собою з’єднання між блоками, інформацію у блоках, надписи на з’єднувальних стрілках та XML файл.

Основні структури даних для створення діаграм у програмному коді: ArrayList, LinkedList, HashMap, XML-файл.

* 1. **Модульне тестування**

Нижче представлено схему алгоритмів для функції Convert to Graph(рисунок 3.1):



Рисунок 3.1 – схема алгоритму Convert to Graph

Нижче представлено схему алгоритмів для функції Convert to XML(рисунок 3.2):



Рисунок 3.2 – схема алгоритму для функції Convert to XML

Функція 1: Convert to Graph

**Білий ящик**

|  |  |
| --- | --- |
| Вхідні параметри | Очікуваний результат |
| Шлях: U:\Users\yyy\Documents\git\ ProjectPractice | Помилка: такого шляху не існує |
| Шлях: ProjectPractice1 | Помилка: такого шляху не існує |
| Шлях: C:\Users\yyy\Documents\git\ ProjectPractice | Коректний путь перехід до пункту збереження у файл з графом |
| Ім’я файлу: onpu123.txt | Неправильний формат файлу |
| Ім’я файлу: политех.graph | Неправильний формат імені файлу(містить кирилицю) |
| Ім’я файлу: onpu123.graph | Збереження файлу за заданим шляхом |

Функція 2: Convert to XML

**Чорний ящик**

Моя система активно взаємодіє з користувачем, тому необхідно побудова класів еквівалентності для обліку всіх можливих помилок при введенні даних. Нижче наведена таблиця класів еквівалентності (таблиця 1).

Таблиця 1 - Таблиця класів еквівалентності

|  |  |  |
| --- | --- | --- |
| Вхідні події | Допустимі класи еквівалентності | Неприпустимі класи еквівалентності |
| Введення назви файлу | Латинські і кириличні символи, цифри, знаки тире і дефіс | Знак точка, знак "@", використання в назві знаків пунктуації |
| Введення даних в блок на діаграмі класів | Дані повинні бути представлені у вигляді пари: назва змінної / методу: Тип; У назвах і іменах типів можуть використовуватися тільки латинські символи; | Не допускається опускання роздільник ":", цей символ обов'язковий; назва або тип не повинно починатися з цифри |
| Введення даних в блок «актор» на діаграмі ВІ | Перший символ повинен бути великою літерою | Перший символ - мала літера/ Перший символ - не буква |
| Введення даних в блок на діаграмі | Допускаються латинські символи;  Перша буква - велика | Допускаються латинські символи;  Перша буква - мала |
| Введення повідомлення на стрілках | Допускаються тільки латинські символи; | Перша буква - рядкова Всі символи, які не входять до латинський алфавіт; Перша бука - заголовна |

1. **ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ**
   1. **Функціональне тестування**

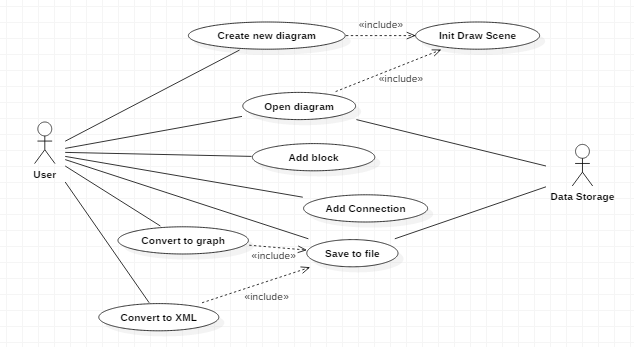


Рисунок 1 – діаграма варіантів використання

**Create New Diagramm**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Условие | Пользователь создал пустую диаграмму | T | T | T | T | F | F | F | F |
| Пользователь указал правильный путь | T | F | T | F | T | T | F | F |
| Пользователь указал правильное имя диаграммы | T | T | F | F | T | F | T | F |
| Действие | Сохранить диаграмму | + |  |  |  |  |  |  |  |
| Вывести сообщение об ошибке: неправильный путь |  | + |  | + | + |  | + | + |
| Вывести сообщение об ошибке: существующее имя |  |  | + |  |  | + |  |  |
| Отображение информации о сохранении диаграммы | + |  |  |  |  |  |  |  |
| Вывести сообщение: отсутствуют данные на диаграмме. Создайте диаграмму |  |  |  |  | + | + | + | + |

Выполним склейку таблицы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Условие | Пользователь создал пустую диаграмму | T | T | T | F | F |
| Пользователь указал правильный путь | T | F | Т | Х | F |
| Пользователь указал правильное имя диаграмму | T | Х | F | Х | T |
| Действие | Сохранить диаграмму | + |  |  |  |  |
| Вывести сообщение об ошибке: неправильный путь |  | + |  | + |  |
| Вывести сообщение об ошибке: существующее имя |  |  | + |  | + |
| Отображение информации о сохранении диаграммы | + |  |  |  |  |
| Вывести сообщение: отсутствуют данные на диаграмме. Создайте диаграмму |  |  |  | + | + |

**Add Block**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Условия | Пользователь добавил новый элемент | T | T | T | T | F | F | F | F |
| Пользователь выбрал элемент | T | T | F | F | T | F | T | F |
| Действие | Отображение элемента на сцене | + |  |  |  |  |  |  |  |
| Выдача сообщения: элемент не выбран |  |  | + | + |  |  |  |  |
| Выдача сообщения об ошибке: отсутствует элемент на диаграмме |  |  |  |  | + |  |  | + |
| Выбор элемента с перечня всех возможных |  |  | + |  |  | + |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Условие | Пользователь добавил новый элемент | T | Х | T | T | F | F |
| Пользователь выбрал элемент | T | T | F | F | Х | F |
| Действие | Отображение элемента на сцене | + |  |  |  |  |  |
| Выдача сообщения: элемент не выбран |  |  | + | + |  |  |
| Выдача сообщения об ошибке: отсутствует элемент на диаграмме |  |  |  |  | + |  |
| Выбор элемента с перечня всех возможных |  |  | + |  |  | + |

**Open Diagramm**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Условие | Пользователь запросил файл из ФС | T | T | T | T | F | F | F | F |
| Пользователь выбрал файл из ФС | T | F | T | F | T | T | F | F |
| Сообщение об ошибке: выберите документ |  | + |  | + |  |  | + | + |
| Сообщение об ошибке: файл не существует |  |  |  |  | + | + | + | + |

**Convert to XML**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Условие | Пользователь сформировал данные для передачи | T | T | F | F |
| Пользователь передал данные в ФС | T | F | T | F |
| Действие | Система сохранила данные в ФС | + |  |  |  |
| Сообщение об ошибке: некорректно сформированы данные |  |  | + |  |
| Сообщение об ошибке: сформируйте данные для передачи |  |  |  | + |

1. **РОЗГОРТАННЯ ПРОГРАМНОГО ПРОДУКТУ**

## **Інструкція з інсталяції**

урахуванням розміщення програмного продукту на порталі github.com.

Кроки виконання супроводжувати прикладами екранних форм.

1. Завантажити файл UMLeditor.jar.
2. Двічі клацнувши по ньому лівою клавішею миші, запустити його.
3. Наступним етапом вже буде робота безпосередньо із самою програмою.
   1. **Інструкція користувача**

У початку роботи програми користувач бачить пусте вікно зі стрічкою меню. Нова діаграма створюється наступним чином: користувач натискає пункт меню File -> New -> Static View(діаграма класів або діаграма кооперацій/коммунікацій) або Dynamic View( діаграма використання або діаграма послідовностей). Далі користувач додає блоки діаграм на графічну сцену та поєднує їх лініями шляхом перетягування блоків на графічну сцену(метод drag`n`drop). Збереження діаграми відбувається таким чином: користувач натискає пункт меню File -> Save та задає ім’я файлу. Для конвертації у файл формату XML/граф необхідно вибрати пункт меню File -> Save as та вибрати ім’я та формат файлу( XML або граф).для відкриття вже створеної діаграми необхідно вибрати пункт меню File -> Open та вибрати файл с діаграмою.

Дана програма дозволяє створювати наступні види діаграм:

Діаграми варіантів використання(рисунок 5.1):

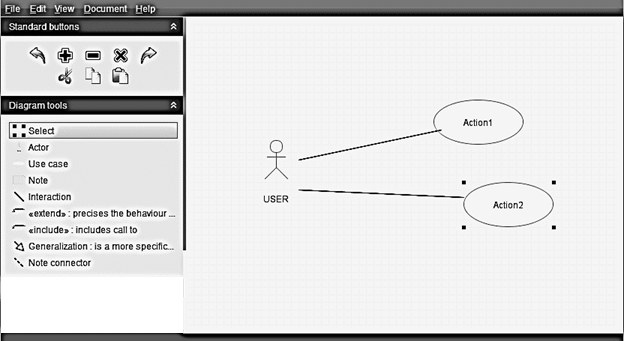


Рисунок 5.1 – Діаграма варіантів використання

Діаграми класів(рисунок 5.2):

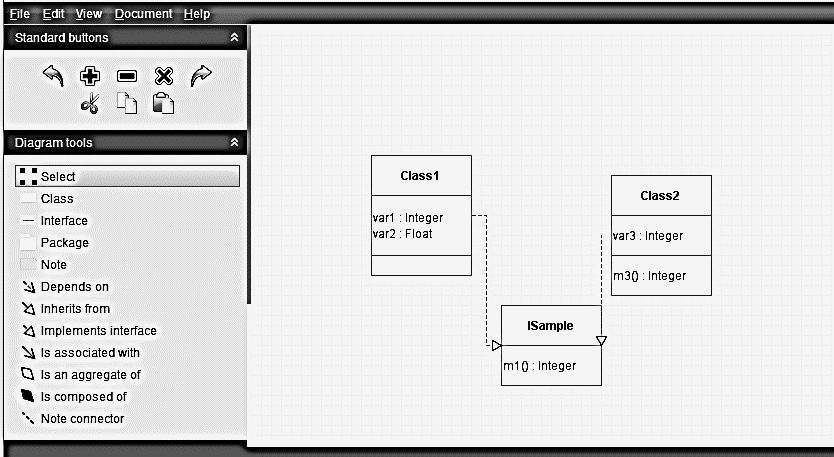


Рисунок 5.2 – Діаграма класів

Діаграми послідовностей(рисунок 5.3):

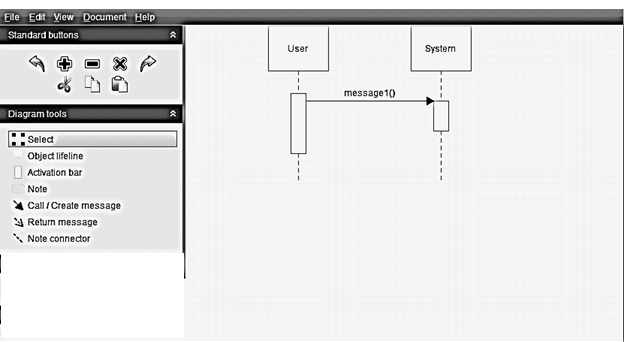


Рисунок 5.3 – Діаграма послідовностей

Діаграма кооперацій(рисунок 5.4):

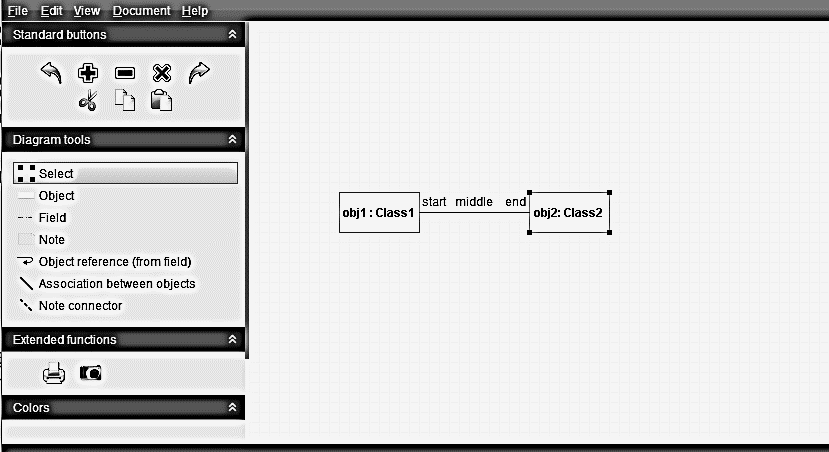


Рисунок 5.4 – Діаграма кооперацій