Міністерство освіти і науки України

**Національний університет «Запорізька політехніка»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Комп’ютерні системи і мережі\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(найменування кафедри)

**КУРСОВИЙ** **ПРОЕКТ**

**(РОБОТА)**

з \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Комп'ютерних Систем\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва дисципліни)

на тему:\_\_ Система для роботи з зображеннями на мові програмування Java\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_III\_ курсу КНТ- групи

Спеціальності 123 комп’ютерна інженерія

освітня програма (спеціалізація) \_\_\_\_Комп’ютерні системи та мережі \_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_Тута.\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_Точилін С.Д.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_ к.ф.-м.н., доцент\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали

2022 рік

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра\_\_\_\_\_\_\_Комп’ютерні системи і мережі\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дисципліна \_\_\_\_**«**Комп'ютені системи»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність \_\_123Комп’ютерна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курс 3 Група КНТ- Семестр 6

**ЗАВДАННЯ**

**на курсовий проект (роботу) студентові**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тута\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи): Система для роботи з зображеннями на мові\_\_\_\_ програмування Java\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Термін здачі студентом закінченого проекту (роботи):\_\_\_1.05.2021\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проекту (роботи): розробити додаток, за допомогою якого забезпечити обробку зображень, доступ до інформації організувати за допомогою бібліотеки swing яка створює графічний інтерфейс\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. ЗМІСТ розрахунково-пояснювальної записки (перелік питань, що їх належить розробити): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Вступна частина (титул, завдання, реферат, зміст, перелік умовних скороч.) \_\_\_\_\_\_\_\_\_\_\_\_

\_\_Основна частина (вступ, спеціальні розділи)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_Заключна частина (висновки, перелік посилань)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання:\_20.03.2022\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - 1. **КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів курсового  проекту (роботи) | Строк виконання етапів курсового проекту ( роботи ) | Примітка |
| 1 | Отримання завдання на курсову роботу | 20.03.2022 |  |
| 2 | Дослідження літератури та матеріалів на задану тему | 5.04.2022 |  |
| 3 | Виконання пояснювальної записки | 10.04.2022 |  |
|  | Вступна частина | 20.04.2022 |  |
|  | Основна частина | 22.04.2022 |  |
|  | Заключна частина | 30.04.2022 |  |
| 4 | Представлення закінченої роботи на перевірку | 01.05.2022 |  |
| 5 | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_Тута\_\_­­­­\_\_\_\_\_\_\_\_\_\_\_\_

( підпис ) (прізвище та ініціали)

Керівник проекту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ Точилін С.Д.\_\_\_\_\_\_\_

( підпис ) (прізвище та ініціали)

«01» Травня 2022 р.

**РЕФЕРАТ**

ПЗ: 45 с., 7 рис., 7 лістингів, 1 табл., 9 джерел, 1 додаток

JAVA, Swing, додаток, image, клас, JFrame, JPanel, Button, BufferedImage, функція, пакет

Об’єкт розробки- система для роботи з зображеннями на мові програмування Java.

Мета проекту- навчитися працювати з бібліотекою Swing та застосовувати на практиці.

В пояснювальній записці розроблений додаток написаний на мові JAVA який повністю відповідає поставленому завданню.

Розроблений додаток містить поле для вибору файлу над яким буде проведено маніпуляції, кнопки за допомогою яких можна буде зберегти отримані зображення та мініатюри того, як зображення буде змінено. При його створенні була задіяна мова програмування JAVA, клас BufferedImage що поставляється в пакеті java.awt.image та графічна бібліотека Swing.

**ЗМІСТ**

[**Вступ** 6](#_Toc106115635)

[1 Мова програмування JAVA 7](#_Toc106115636)

[1.1 Історія мови 7](#_Toc106115637)

[1.2 Структура мови 10](#_Toc106115638)

[1.3 Пакети 17](#_Toc106115639)

[2 Пакети java.awt та javax.swing 20](#_Toc106115640)

[2.1 Інформація про пакет java.awt 20](#_Toc106115641)

[2.2 Інформація про пакет javax.swing 23](#_Toc106115642)

[3 Практична частина 26](#_Toc106115643)

[3.1 Постановка задачі 26](#_Toc106115644)

[3.2 Складання технічного завдання на розробку додатку. 26](#_Toc106115645)

[3.3 Опис функцій на методів що використовувались при розробці 27](#_Toc106115646)

[3.4 Огляд можливостей розробленого додатку 34](#_Toc106115647)

[Висновок 37](#_Toc106115648)

[Перелік джерел посилань 38](#_Toc106115649)

[Додаток А 39](#_Toc106115650)

# **Вступ**

Ми живемо у вік цифровізації, щосекунди в світі робляться мільйони фотографій за допомогою телефонів або фотоапаратів, різні художники малюють цифрові рисунки. Інколи трапляються не казуси з цими зображеннями, як наприклад, потрібно відзеркатили, повернути на 90 градусів, через те, що телефон при фотографуванні не змінив орієнтацію та ми отримали перевернуте зображення, або нам потрібно інвертувати чи зробити фото чорно білим, шоб воно виглядало краще. Для таких задач можна використовувати спеціальні графічні програми за допомогою яких виконується така обробка але вони можуть бути як платними так і використовувати багато ресурсів комп’ютера.

Щоб вирішити проблеми з ресурсами, можна створити свій спрощенний додаток, який не буде мати всього того функціоналу, який навантажує комп’ютер, а буде вирішувати лише ті певні задачі які ми можемо самі за програмувати та кастомізувати.

Таким додатком можна користуватись як за допомогою консолі так і за допомогою графічного додатку. Якщо працювати за першим варіантом, то такий додаток буде працювати швидше, та швидше розроблюватись, але ним буде складно керувати, якщо створити за другим варіантом, а саме з графічним відображенням, то ми отримаємо зручний користувацький інтерфейс в якому можна зробити навігацію для знаходження оброблювального зображення, кнопок для їх збереження та мініатюр з результатом обробки.

Для створення такого додатку було обрано мову програмування Java та пакети java.awt для роботи з зображеннями та javax.swing для створення графічної оболонки.

Для виконання курсового проекту були поставлені наступні завдання: ознайомитися з пакетами java.awt та javax.swing, розібратися з роботою і їх поведінкою.

## 1 Мова програмування JAVA

### 1.1 Історія мови

Мова програмування - це формальна знакова система, призначена для запису комп'ютерних програм. Мова програмування визначає набір лексичних, синтаксичних і семантичних правил, що визначають зовнішній вигляд програми і дії, які виконає виконавець (зазвичай -ЕВМ) під її управлінням.

Java - це об'єктно-орієнтована мова програмування, розроблений компанією Sun Microsystems (в подальшому придбаної компанією Oracle) .Пріложенія Java зазвичай транслюються в спеціальній байт-код, тому вони можуть працювати на будь-якій віртуальній Java-машині незалежно від комп’ютерної архітектури. Дата офіційного випуску- 23 травня 1995 року.

Спочатку мова називалася Oak ( «Дуб»), розроблялася Джеймсом Гослінгом для програмування побутових електронних пристроїв. Через те, що мова з такою назвою вже існувала, Oak був перейменований в Java [4].

Існує 3 основні версії чому так назвали мову:

Версія № 1: Найцікавіша. Програмісти п'ють багато кави. Поки створювався нову мову програмування, було випито гігалітри кави, і тому вирішили назвати мову в честь сорту кави Java.

Версія № 2: Назва дана на честь марки кави Java, який був названий на честь острова Ява (на англ. - Java). Ця версія перегукується з версією № 1.

Версія № 3: Оскільки нова мова розроблялася для програмування побутових електронних пристроїв, то іноді пов'язують це з ілюзією на каво-машину як приклад побутового пристрою.

Програми на Java транслюються в байт-код, що виконується віртуальною java-машиною (JVM) - програмою, обробній байтовий код і передавальної інструкції обладнанню як інтерпретатор, але з тією відмінністю, що байтовий код, на відміну від тексту, обробляється значно швидше. Гідність подібного способу виконання програм - в повній незалежності байт-коду від ОС і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, який підтримує віртуальну машину. Додатки можна запустити на багатьох платформах. Одного разу написаний додаток не доведеться модифікувати під інші платформи: воно буде працювати без будь-яких змін на різних операційних системах і апаратних архітектурах.

Цикл розробки програмних засобів з використанням Java значно скорочується в силу того, що Java - інтерпретована мова. Процес компіляції-збірки-завантаження - тепер програму треба тільки відкомпілювати і відразу запускати. Java контролює звернення до пам'яті. Незважаючи на те, що ця мова - інтерпретована, його код оптимізується до фази виконання. Підтримка системи багатопоточності дозволяє створювати паралельно виконувані взаємодіючі легковагі процеси.

Важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

Часто до недоліків концепції віртуальної машини відносять зниження продуктивності. Ряд удосконалень кілька збільшив швидкість виконання програм на Java:

-застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді,

-обширне використання переносних орієнтованого коду (native-код) в стандартних бібліотеках,

-апаратні кошти, що забезпечують прискорену обробку байт-коду (наприклад, технологія Jazelle, підтримувана деякими процесорами архітектури ARM).

За даними сайту shootout.alioth.debian.org, для семи різних завдань час виконання на Java становить в середньому в півтора-два рази більше, ніж для C / C ++, в деяких випадках Java швидше, а в окремих випадках в 7 разів повільніше. З іншого боку, для більшості з них споживання пам'яті Java-машиною було в 10-30 разів більше, ніж програмою на C / C ++. Також примітно дослідження, проведене компанією Google, згідно з яким відзначається істотно нижча продуктивність і більше споживання пам'яті в тестових прикладах на Java в порівнянні з аналогічними програмами на C ++.

За даними компанії Oracle, програми на Java запускаються на 3 млрд девайсів. Це маркетингове повідомлення складно перевірити. Проте Java широко використовується і входить в число найбільш затребуваних мов, це не викликає сумніву.

Наприклад, переважна більшість великих компаній так чи інакше використовують Java. Дуже багато серверних додатків для корпорацій написані на цій мові. Наприклад, мова йде про програми для фінансових організацій, які забезпечують проведення транзакцій, фіксацію торгових операцій.

На Java написано багато веб-додатків. Популярні фреймворки, в тому числі Spring, Stuts, JSP, використовуються для створення різних додатків в інтернеті: від ecommerce-проектів до великих порталів, від освітніх платформ до урядових ресурсів. Популярна комп'ютерна гра Minecraft написана на Java.

Мобільна розробка - ще одна область використання Java. Цією мовою пишуть програми для пристроїв, що працюють під управлінням ОС Android.

На Java створюють клієнтські програми. Простий і близький розробникам приклад: IDE NetBeans написано на «Джаві».

Також Java застосовується для роботи з Big Data, розробки програм для наукових цілей, наприклад, обробки природних мов, програмування приладів - від побутових девайсів до промислових установок.

Тобто на Java можна писати різні типи додатків: веб, мобільний і десктопний софт, ігри і так далі. Традиційно у цієї мови сильні позиції в промисловому програмуванні, в сегменті великих компаній.

### 1.2 Структура мови

Для того щоб зрозуміти структуру Java, потрібно спочатку зрозуміти що таке об'єктно-орієнтоване програмування (ООП).

Об'єктно-орієнтоване програмування (скорочено ООП) - це парадигма розробки програмних систем, в якій додатки складаються з об'єктів,, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію спадкування. Об'єкти - це сутності, у яких є властивості і поведінку. Зазвичай об'єкти є екземплярами якого-небудь класу. Поведінка об'єкта визначається за допомогою методів - спеціальних блоків коду, які можна викликати з різних частин програми. Використовуючи ці властивості і методи, можна значно прискорити розробку, зробити код більш читабельним. До того ж самому програмісту простіше складати код, якщо він думає з допомогою об'єктів.

Ідеологічно ООП - підхід до програмування як до моделювання інформаційних об'єктів, вирішальний на новому рівні основне завдання структурного програмування: структурування інформації з точки зору керованості, що істотно поліпшує керованість самим процесом моделювання, що, в свою чергу, особливо важливо при реалізації великих проектів.

Керованість для ієрархічних систем передбачає мінімізацію надмірності даних (аналогічну нормалізації) і їх цілісність, тому створене зручно керованим - буде і зручно розумітися. Таким чином, через тактичну задачу керованості вирішується стратегічне завдання - транслювати розуміння завдання програмістом в найбільш зручну для подальшого використання форму.

Основні принципи структурування в разі ООП пов'язані з різними аспектами базового розуміння предметної завдання, яке потрібно для оптимального управління відповідною моделлю:

-абстракція - це використання тільки тих характеристик об'єкта, які з достатньою точністю представляють його в даній системі. Основна ідея полягає в тому, щоб представити об'єкт мінімальним набором полів і методів і при цьому з достатньою точністю для розв'язуваної задачі.

-інкапсуляція - властивість мови програмування, що дозволяє користувачеві не замислюватися про складність реалізації використовуваного програмного компонента (що у нього всередині), а взаємодіяти з ним за допомогою наданого інтерфейсу (публічних методів і членів), а також об'єднати і захистити життєво важливі для компонента дані. При цьому користувачеві надається тільки специфікація (інтерфейс) об'єкта.

-успадкування - дозволяє описати новий клас на основі вже існуючого (батьківського), при цьому властивості і функціональність батьківського класу запозичуються новим класом. Це дозволяє звертатися з об'єктами класу-спадкоємця точно так же, як з об'єктами базового класу.

-поліморфізм - можливість об'єктів з однаковою специфікацією мати різну реалізацію. Коротко зміст поліморфізму можна виразити фразою: «Один інтерфейс, безліч реалізацій».

При написанні програм використовуються класи і методи. Головна їхня відмінність - в тому, що клас складається з інтерфейсу і реалізації.

Об'єкт - деяка сутність володіє певним станом і поведінкою, що має певні властивості (атрибути) і операції над ними (методи). Як правило, при розгляді об'єктів виділяється те, що об'єкти належать одному або декількох класах, які визначають поведінку (є моделлю) об'єкта. Об'єкти класів, як і дані примітивних типів, можуть передаватися в методи. Однак в даному випадку є одна особливість - при передачі об'єктів в якості значення передається копія посилання на область в пам'яті, де розташований цей об'єкт.

Клас - являє собою шаблон для створення об'єктів, що забезпечує початкові значення станів: ініціалізація полів-змінних і реалізація поведінки функцій або методів. Інші абстрактні типи даних - інтерфейси характеризуються якимись своїми, іншими особливостями. Клас є ключовим поняттям в ООП. На лістингу 1.1 приведений приклад створення класу мовою java, який містить в собі поля і змінні.

Лістинг 1.1 – Приклад створення класу

public class StudentUnivers {

String firstName;

String lastName;

int age;

String nameGroup;

String city;

int course;

void coutStudentInfo() {

System.out.printf("Name: %s \nLast Name: %s \nAge: %d\nGroup: %s\n City: %s\n Course: %d", firstName, lastName, age, nameGroup, city, course);}}

Крім звичайних методів класи можуть визначати спеціальні методи - конструктори. Конструктори викликаються при створенні нового об'єкта даного класу. Конструктори виконують ініціалізацію об'єкта. Якщо в класі не визначено жодного конструктора, то для цього класу автоматично створюється конструктор без параметрів. У лістингу 1.2 приведений приклад класу з наявним конструктором.

Лістинг 1.2 - Приклад класу з наявним конструктором

public class StudentUnivers {

String firstName;

String lastName;

int age;

String nameGroup;

String city;

int course;

public StudentUnivers (String firstName, String lastName, int age, String nameGroup, String city, int course) {

this.firstName = firstName;

this.lastName = lastName;

this.age = age;

this.nameGroup = nameGroup;

this.city = city;

this.course = course; }

void coutStudentInfo() {

System.out.printf("Name: %s \nLast Name: %s \nAge: %d\nGroup: %s\n City: %s\n Course: %d", firstName, lastName, age, nameGroup, city, course);

}}

public static void main(String[] args) {

StudentUnivers Ivan = new StudentUnivers ("Іван","Головко",19,"РТ-218","Николаев",3); }

Класи можуть успадковуватися один від одного. Клас-нащадок отримує всі поля і методи класу-батька, але може доповнювати їх власними або перевизначати вже наявні. Більшість мов програмування підтримує тільки одиничне успадкування (клас може мати тільки один клас-батько), лише в деяких допускається множинне спадкування - породження класу від двох або більше класів-батьків. Множинне успадкування створює цілий ряд проблем, як логічних, так і чисто реалізаційних, тому в повному обсязі його підтримка не поширена. На лістингу 1.3 приведений приклад наслідування класу.

Лістинг 1.3 – Приклад наслідування класів.

class Human {

String firstName;

String lastName;

int age;

Human(String firstName, String lastName, int age){

this.firstName = firstName;

this.lastName = lastName;

this.age = age; }

void coutHumanInfo(){

System.out.printf("Name: %s \nLast Name: %s \nAge: %d\n", firstName, lastName, age);

}}

class StudentUnivers extends Human{

String nameGroup;

String city;

int course;

StudentUnivers (String firstName, String lastName, int age, String nameGroup, String city, int course) {

super(firstName,lastName,age);

this.nameGroup = nameGroup;

this.city = city;

this.course = course;

}

void coutStudentInfo(){

System.out.printf("Name: %s \nLast Name: %s \nAge: %d\nGroup: %s\n City: %s\n Course: %d", firstName, lastName, age, nameGroup, city, course);

}

}

public static void main(String[] args) {

StudentUnivers Ivan = new StudentUnivers ("Іван","Головко",19,"РТ-218","Николаев",3);

}

Інтерфейс - це клас без полів і без реалізації, що включає тільки заголовки методів. Якщо якийсь клас успадковує (або, як кажуть, реалізує) інтерфейс, він повинен реалізувати всі вхідні в нього методи. Використання інтерфейсів надає відносно дешеву альтернативу множинного спадкоємства. Взаємодія об'єктів в абсолютній більшості випадків забезпечується викликом ними методів один одного.

Лістинг 1.4 – Приклад роботи інтерфейсу

public class Test {

public static void main(String[] args) {

B tempObj = new A("Клас А ");

tempObj.printHello();

tempObj.printBye();

tempObj = new B("Клас Б ");

tempObj.printHello();

tempObj.printBye();

}}

interface Interface {

void printHello();

void printBye();}

class A implements Interface {

String name;

A(String name) {

this.name = name; }

public void printHello() {

System.out.printf("%s каже привіт \n", name);

}

public void printBye() {

System.out.printf("%s каже бувай \n", name);

}

}

class B implements Interface {

private String name;

B(String name) {

this.name = name;

}

public void printHello() {

System.out.printf("З вами привітався %s \n", name);

}

public void printBye() {

System.out.printf("З вами попрощався %s \n", name);

}

}

Модифікатори доступу – мають всі члени класу: поля і методи. Модифікатори доступу дозволяють задати допустиму область видимості для членів класу, тобто контекст, в якому можна використовувати цю змінну або метод. Щоб використовувати модифікатор в Java, потрібно включити його ключове слово в визначенні класу, методу або змінної. Модифікатор повинен бути попереду решти операторів.

В Java використовуються наступні модифікатори доступу:

- public : публічний, загальнодоступний клас або член класу. Поля і методи, оголошені з модифікатором public, видно з інших класів поточного пакета і з зовнішніх пакетів.

-private : закритий клас або член класу, протилежність модифікатору public. До змінної, методу або класу, позначеного модифікатором private , можна звертатися тільки з того ж класу, де він оголошений. Для всіх інших класів позначений метод або змінна - невидимі. Це найвища ступінь закритості - тільки свій клас. Такі методи не успадковуються і не перевизначаються. Доступ до них з класу-спадкоємця також неможливий.

-protected : такий клас або член класу доступний в межах всіх класів, що знаходяться в тому ж пакеті, що і наш або в межах всіх класів-спадкоємців нашого класу. В основному даний модифікатор доступу використовують при наслідуванні класів.

-Модифікатор за замовчуванням. Якщо змінна або метод не позначені ніяким модифікатором, то вважається, що вони позначені «модифікатором за замовчуванням». Змінні і методи з таким модифікатором видно всіх класах пакету, в якому вони оголошені, і тільки їм. Цей модифікатор ще називають « package » або « package private », натякаючи, що доступ до змінних і методів відкритий для всього пакету, в якому знаходиться їх клас

Більш детально розібратися з специфікаторами доступу допоможе таблиця 1.1 на якій зображено те який мають доступ змінні методи та класи у різних ситуаціях.

Таблиця 1.1 – Специфікатори доступу

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип видимості | Ключове слово | Доступ | | | |
| Свій клас | Свій пакет | Клас - спадкоємець | Інші класи |
| Закритий | private | є | немає | немає | немає |
| Пакет | (Немає модифікатора) | є | є | немає | немає |
| Захищений | protected | є | є | є | немає |
| Відкритий | public | є | є | є | є |

Модифікатори класу, методу, змінної і потоку, які використовуються не для доступу. Java надає ряд модифікаторів не для доступу, а для реалізації багатьох інших функціональних можливостей:

- модифікатор static застосовується для створення методів і змінних класу. Ключове слово static використовується для створення змінних, які будуть існувати незалежно від будь-яких екземплярів, створених для класу. Тільки одна копія змінної static в Java існує незалежно від кількості екземпляру класу. Статичні змінні також відомі як змінні класу. В Java локальні змінні не можуть бути оголошені статичними. Ключове слово static при використанні для створення методів, які будуть існувати незалежно від будь-яких екземплярів, створених для класу. В Java статичні методи або методи static не використовують будь-які змінні екземпляра будь-якого об'єкта класу, вони визначені. Методи static приймають всі дані з параметрів і щось з цих параметрів обчислюється без посилання на змінні. Змінні і методи класу можуть бути доступні з використанням імені класу, за яким слідує крапка і ім'я змінної або методу.

- модифікатор final використовується для завершення реалізації класів, методів і змінних. Зміна final може бути ініціалізована тільки один раз. Посилальна змінна, оголошена як final, ніколи не може бути призначена для позначення іншого об'єкта. Однак дані всередині об'єкта можуть бути змінені. Таким чином, стан об'єкта може бути змінено, але не посилання. При створенні змінних в Java модифікатор final часто використовується з static, щоб зробити константою змінну класу. Метод final не може бути перевизначений будь-яким підкласом. Як згадувалося раніше, в Java модифікатор final захищає метод від змін в підкласі. Головним наміром зробити метод final буде те, що зміст методу не повинно бути змінено на стороні.

- модифікатор abstract необхідний для створення абстрактних класів і методів. Клас з таким модифікатором не може створити екземпляр. Якщо клас оголошений як abstract, то єдина мета для нього бути розширеним за допомогою наслідування. Клас не може бути одночасно abstract і final, так як клас final не може бути розширеним. Якщо клас містить абстрактні методи, то він повинен бути оголошений як abstract. В іншому випадку буде згенеровано помилка компіляції. Клас abstract може містити як абстрактні методи, а також і звичайні. Метод abstract є методом, оголошеним з будь реалізацією. Тіло методу (реалізація) забезпечується підкласом. Будь-клас, який розширює абстрактний клас повинен реалізувати всі абстрактні методи суперкласу, якщо підклас не є абстрактним класом.

### 1.3 Пакети

Пакет Java - є спеціальним бібліотечним модулем, який містить групу класів, об'єднаних в одному просторі імен. Їх використовуються для запобігання конфліктів з назвами, для контролю доступу, для полегшення пошуку / знаходження і використання класів, інтерфейсів. Приклад правильного звернення до класу: «ім'я пакета» + «ім'я класу».

Щоб вказати, що клас належить певному пакету, треба використовувати директиву package, після якої вказується ім'я пакета: «package назва\_пакета;»

Як правило, назви пакетів відповідають фізичній структурі проекту, тобто організації каталогів, в яких знаходяться файли з вихідним кодом. А шлях до файлів всередині проекту відповідає назві пакета цих файлів.

Класи необов'язково визначати в пакети. Якщо для класу пакет не визначений, то вважається, що даний клас знаходиться в пакеті за замовчуванням, який не має імені.

Імпорт пакетів - Якщо нам треба використовувати класи з інших пакетів, то нам треба підключити ці пакети і класи. Наприклад використовуємо клас Scanner який знаходиться в пакеті java.util для отримання даний вводяться в консоль. Що б імпортувати пакет ми прописуємо: «import java.util.Scanner;», тобто ми вказуємо повний шлях до файлу в пакеті, і створюємо об'єкт «Scanner in = new Scanner (System.in);» за допомогою якого зможемо викликати функцію nextInt (), яка забирає введені значення з консолі. Імпортувати можна необмежену кількість пакетів в програмований проект.

Основні пакети які поставляються в java:

java.lang - базова функціональність мови та основні типи

java.util - включає велику кількість допоміжних класів, широко використовуваних в інших вбудованих пакетах Java. Ці класи розташовані в пакеті java.util і використовуються для роботи з набором об'єктів, взаємодії з системними функціями низького рівня, для роботи з математичними функціями, генерації випадкових чисел і маніпуляцій з датою і часом

java.io - містить майже кожен клас, який може знадобитися для здійснення введення і виведення в Java. Всі дані потоки представлені потоком введення і адресою виведення. Потік в пакеті java.io здійснює підтримку різних даних, таких як примітиви, об'єкти, локалізовані символи і т.п;

java.nio - новий фреймворк для прикладних програмних інтерфейсів мови Java, призначених для реалізації високопродуктивних операцій введення-виведення;

java.net - операції з мережами, сокетами, DNS-запитів підтримує сімейства протоколів TCP, UDP і т.д. У складі пакету java.net є безліч класів, пов'язаних з мережею;

java.security - генерація ключів, шифрування і дешифрування;

java.sql - Java Database Connectivity (JDBC) для доступу до баз даних;

javax.swing - ієрархія пакетів для переносних GUI компонентів;

java.awt - ієрархія основних пакетів для рідних компонентів GUI надає технологію AWT для створення графічного (віконного) інтерфейсу користувача – GUI;

Для того що б створити свій власний пакет, потрыбно вибрати йому ім'я і включити оператор package разом з цим ім'ям поверх кожного вихідного файлу, який містить класи, інтерфейси, перерахування та типи анотацій, які ви хочете включити в пакет.

Оператор package повинен бути першим рядком у вихідному файлі. Може бути тільки один запит package в кожному вихідному файлі, і він застосовується до всіх типів в цьому файлі. Якщо оператор package не використано, тоді класи, інтерфейси, перерахування та типи анотацій будуть поміщені в поточний пакет за замовчуванням.

Пакети повинні розташовуватися у відповідних директоріях, тобто файл пакета (з ім'ям ІмяПакета) повинен бути збережений в папці ІмяПакета.

Вони можуть бути вкладеними (наприклад, java.util, де java- головний пакет, autil - вкладений). Відповідно ці два пакети повинні розташовуватися в каталозі java \ util.

## 2 Пакети java.awt та javax.swing

### 2.1 Інформація про пакет java.awt

Пакет package java.awt містить всі класи для створення інтерфейсів і для малювання графіки і зображень. Об'єкт інтерфейсу, такий як кнопка або смуга прокручування, в термінології AWT називається компонентом. всіх компонентів AWT, наведено у розділі Компонент.

Деякі компоненти викликають події, коли користувач взаємодіє з ними. Клас AWTEvent і його підкласи використовуються для представлення подій, які можуть викликати компоненти AWT. Опис моделі подій AWT див.

Контейнер-це компонент, який може містити компоненти та інші контейнери. Контейнер також може мати менеджер компонування, який керує візуальним розміщенням компонентів у контейнері. та Менеджер компонування.

Кожен об'єкт Component обмежений за своїм максимальним розміром та місцезнаходженням, оскільки значення зберігаються у вигляді цілого числа. Крім того, платформа може додатково обмежувати максимальний розмір та координати розташування. Точні максимальні значення залежать від платформи. Ці максимальні значення неможливо змінити ні в коді Java, ні в машинному коді. Ці обмеження також накладають обмеження на компонування компонентів. Якщо межі об'єкта Component перевищують межу платформи, неможливо правильно розташувати їх в об'єкті-контейнері. Кордони об'єкта визначаються координатою будь-якого об'єкта разом із його розміром відповідної осі.

Пакет java.awt.image надає класи для створення та зміни зображень. Зображення обробляються за допомогою потокової структури, яка включає в себе виробника зображень, додаткові фільтри зображень та споживача зображень. Ця структура дозволяє поступово відображати зображення під час його отримання та генерації. Більше того ,фреймворк дозволяє застосунку відкидати пам'ять, що використовується зображенням, і в будь-який момент генерувати його заново. Цей пакет надає ряд виробників, споживачів та фільтрів зображень, які ви можете налаштувати для потреб обробки зображень.

BufferedImage підклас описує Image з доступним буфером даних зображення. BufferedImage складається з ColorModel і Raster даних зображення. Кількість і типи смуг SampleModel в Raster , повинні відповідати кількості і типи , необхідну ColorModel представляти його колір і альфа - компоненти. Всі об'єкти BufferedImage мають координату верхнього лівого кута (0, 0). Будь-який Raster використовується для побудови BufferedImage, отже, повинні бути Minx = 0 і Miny = 0.

Цей клас покладається на методи вибірки та налаштування даних Raster, а також методи характеристики кольору ColorModel.

Пакет package java.awt.geom надає класи Java 2D для визначення та виконання операцій над об'єктами, пов'язаними з двовимірною геометрією. Деякі важливі особливості пакету включають:

• класи для маніпулювання геометрією, такі як AffineTransform та інтерфейс PathIterator, що реалізується всіма об'єктами Shape.

• класи, що реалізують інтерфейс Shape, такі як CubicCurve2D, Ellipse2D, Line2D, Rectangle2D та GeneralShape.

• клас Area, який надає механізми для операцій складання (об'єднання), віднімання, перетину та exclusiveOR над іншими об'єктами Shape.

Публічний клас AffineTransform Клас являє собою 2D-афінне перетворення, яке виконує лінійне відображення 2D-координат в інші 2D-координати, зберігаючи "прямолінійність" і "паралельність ліній. Аффинные перетворення можуть бути побудовані з використанням послідовностей перекладів, шкал, переворотів, поворотів і зрушень.

Таке перетворення координат може бути представлено матрицею 3 рядка на 3 стовпця з неявної останньої рядком [ 0 0 1 ]. Ця матриця перетворює вихідні координати (x,y)координати призначення(x',y'), вважаючи їх вектором стовпців і помноживши вектор координат на матрицю у відповідності з наступним процесом:

У деяких варіаціях rotateметодів в AffineTransform клас, аргумент подвійної точності визначає кут повороту в радіанах. Ці методи мають спеціальну обробку для поворотів приблизно на 90 градусів (включаючи кратні, такі як 180, 270 і 360 градусів), так що загальний випадок обертання квадранта обробляється більш ефективно. Ця спеціальна обробка може призвести до того, що кути, дуже близькі до кратним 90 градусам, будуть розглядатися так, як якщо б вони були точними кратними 90 градусам.

Для малих кратних 90 градусів діапазон кутів, розглянутих як поворот квадранта, становить приблизно 0,00000121 градуси в ширину. У цьому розділі пояснюється, чому потрібен такий особливий догляд і як він реалізується.

Оскільки 90 градусів представлені PI/2в радіанах, а ПІ - трансцендентне (і, отже, ірраціональне число, неможливо точно уявити кратне 90 градусам як точне значення подвійної точності, виміряне в радіанах. В результаті теоретично неможливо описати повороти квадранта (90, 180, 270 або 360 градусів) з допомогою цих значень.

Значення з плаваючою комою подвійної точності можуть бути дуже близькі до ненульовим значеннямPI/2, але ніколи не досить близькі, щоб синуса або косинуса були точно дорівнюють 0.0, 1.0 або -1.0. Math.sin() Math.cos() відповідно , ніколи не повертайте 0.0 для жодного іншого випадку Math.sin(0.0). Однак ці ж реалізації повертають рівне 1.0 і -1.0 для деякого діапазону чисел навколо кожного кратного 90 градусів, оскільки правильна відповідь настільки близький до 1.0 або -1.0, що значення подвійної точності не може представляти різницю так само точно, як для чисел, близьких до 0.0.

Кінцевим результатом цих проблем є те, що якщо методи Math.sin() and Math.cos() використовуються для безпосереднього генерування значень для модифікацій матриці під час цих операцій обертання на основі радиана, то результуюче перетворення ніколи не буде строго класифікуватися як обертання квадранта навіть для такого простого випадку, як rotate(Math.PI/2.0) з-за незначних варіацій матриці, викликаних значеннями, відмінними від 0.0, отриманими для синуса і косинуса. Якщо ці перетворення не класифікуються як обертання квадранта, то наступний код, який намагається оптимізувати подальші операції на основі типу перетворення, буде віднесений до його найбільш загальній реалізації.

Оскільки повороти квадранта досить поширені, цей клас повинен обробляти ці випадки досить швидко, як при застосуванні поворотів до перетворення, так і при застосуванні результуючого перетворення координат. Щоб полегшити це оптимальне звернення, методи, які приймають кут повороту, що вимірюється в радіанах, намагаються виявити кути, які повинні бути поворотами квадранта, і розглядати їх як такі. Тому ці методи розглядають кут *тета* як поворот квадранта якщо або Math.sin(*theta*), або Math.cos(*theta*) повертає рівне 1.0 або -1.0. Як правило, це властивість справедливо для діапазону приблизно 0.0000000211 радианов (або 0.00000121 градуси) навколо невеликих кратних Math.PI/2.0.

### 2.2 Інформація про пакет javax.swing

Swing — це бібліотека для створення графічного інтерфейсу для програм Java. Swing було розроблено компанією Sun Microsystems. Він містить ряд графічних компонентів (Swing widgets), таких як кнопки, поля введення, таблиці і т.д.

Swing відноситься до бібліотеки класів JFC, яка є набором бібліотек для розробки графічних оболонок. До цих бібліотек відносяться Java 2D, Accessibility-API, Drag&Drop-API та AWT.

Компонент Swing має модель як окремий елемент, у той час як частини View і Controller об'єднані в елементах інтерфейсу користувача. Через це Swing має архітектуру зовнішнього вигляду, що підключається.

Легка вага – компоненти Swing не залежать від API власної операційної системи, тому що елементи керування Swing API відображаються з використанням чистого коду JAVA, а не викликів базової операційної системи.

Багаті елементи керування – Swing надає багатий набір розширених елементів керування, таких як Tree, TabbedPane, слайдер, палітра кольорів та елементи керування таблицями.

Широкі можливості налаштування – елементи керування Swing можна легко налаштувати, оскільки вигляд не залежить від внутрішнього уявлення.

Змінний вигляд – графічний інтерфейс на основі SWING Зовнішній вигляд програми може змінюватись під час виконання залежно від доступних значень.

Кожен інтерфейс користувача розглядає наступні три основні аспекти:

Елементи інтерфейсу користувача - це основні візуальні елементи, з якими користувач в кінцевому підсумку бачить і взаємодіє. GWT надає величезний список широко використовуваних та поширених елементів від базових до складних.

Макети – вони визначають, як елементи інтерфейсу користувача повинні бути організовані на екрані, і забезпечують остаточний зовнішній вигляд графічного інтерфейсу користувача.

Поведінка – це події, які відбуваються, коли користувач взаємодіє з елементами інтерфейсу користувача.

Для групування компонентів інтерфейсу використовуються контейнери (Container). Для створення основного контейнера програми найчастіше використовується контейнер JFrame (є ще JWindows і JApplet). Найпростіше успадковуватися від JFrame тим самим отримати доступ до багатьох методів, наприклад:

setBounds(x, y, w, h) - вказує координати верхньої лівої вершини вікна, а також його ширину та висоту.

setResizable(bool) – вказує, чи можна змінювати розмір вікна.

setTitle(str) – встановлює назву вікна.

setVisible(bool) - що відображає вікно.

setDefaultCloseOperation(operation) - вказує операцію, яка буде зроблена при закритті вікна.

Основні елементи керування:

JLabel – елемент для відображення фіксованого тексту;

JTextField – простий edit-box;

JButton – звичайна кнопка (button);

JCheckBox – елемент вибору (аналог checkbox);

JRadioButton - кнопка радіо

При відображенні елементів керування використовуються спеціальні менеджери – LayoutManager. У всіх LayoutManager'ів є методи додавання у видалення елементів.

FlowLayout – використовується для послідовного відображення елементів. Якщо елемент не міститься у певному рядку, він відображається у наступному.

GridLayout – відображення елементів у вигляді таблиці з однаковими розмірами осередків.

BorderLayout – використовується при відображенні не більше 5 елементів. Ці елементи розташовуються по краях кадру і в центрі: North, South, East, West, Center.

BoxLayout – відображає елементи у вигляді рядка або колонки.

GridBagLayout - дозволяє призначати місцезнаходження та розмір кожного віджету. Це найскладніший, але найефективніший вид відображення.

Варто ще звернути увагу на обробку подій. І тому використовуються звані Event Listeners.

getContentPane повертає контейнер верхнього рівня. ButtonGroup служить створення групи взаємозалежних радіо-кнопок.

Внутрішній клас ButtonActionListener продає інтерфейс ActionListener. Для цього необхідно надати імплементацію методу actionPerformed.

JOptionPane використовується для відображення діалогових вікон.

## 3 Практична частина

### 3.1 Постановка задачі

Потрібно створити додаток, який можна використовувати для обробки зображень на комп'ютерах з не високою ефективністю, також так як даний додаток створено за допомогою мови Java, то її можна буде рапустити на будь-якій платформі, яка підтримує віртуальну машину java. Також він повинен мати графічний інтерфейс за допомогою якого можна буде легко задати шлях до зображення та відображено кнопки за допомогою яких можна зберегти оброблені зображення та переглянути їх мініатюри.

Ця програма за допомогою спеціального поля, відкривати меню з зручною навігацією за допомогою якої можна буде обрати зображення яке буде оброблюватись.

Та повинна мати наступний функціонал для відображення мініатюри:

- обраного зображення;

- зображення поверненого на 180 градусів;

- зображення поверненого на 90 градусів;

- зображення у дзеркальному вігляді;

- зображення у негативі(інверсивному вигляді);

- зображення у чорно/білому вигляді.

Та повинна мати аналогічні кнопки для збереження обробленого фото.

### 3.2 Складання технічного завдання на розробку додатку.

Розроблювальний додаток повинен бути зручним в користуванні. На робочій поверхні екрану повинні бути розташовані такі елементи: поле для обирання шляху до зображення, кнопки збереження обробленого фото певним фільтром та відображення мініатюри з фотографіями.

Додаток повинен відкриватись у вигляді повноцінної програми, мати можливість відкриватися на ввесь екран та масштабуватись, кольори повинні бути приємними для ока, та мати функціонал завершення програми або її згортання до нижньої панелі.

Вміст цього додатку повинен бути поділений на функціональні блоки:

- перший блок відповідає за введення шляху до зображення яке буде оброблюватись у ході роботи програми,

- другий блок для кнопок які будуть зберігати оброблене зображення, він буде типу сітки з розмірністю 3х2 та розміщувати в собі кнопки відповідно до розміщення зображень нижче

- третій буде відображати мініатюри з тією ж самою сіткою 3х2 та буде складатись с упорядкованих зображень з підписаним фільтром який було на них накладено.

### 3.3 Опис функцій на методів що використовувались при розробці

При розробці додатку для програмування було використано середовище розробки Intelij IDE, за його допомогою можна легко та швидко розроблювати программи з нуля, та зручно їх компілювати для перевірки їх працездатності, також є режим відладки за допомогою якого можна легко знайти помилку в коді, переглянувши вміст змінної.

Перед тим як починати писати логіку додатку, потрібно спочатку створити графічний інтерфейс, за допомогою якого буде виконуватись взаємодія людини з програмою.

Інтерейс було поділено на блоки для того щоб користувачу було зручно користуватися програмою. Повний код розробленої програми приведено в додатках А.1 - А.3.

Для кращого розуміння побудови інтерфейсу, розглянемо те, як створюється фундамент додатку а саме його вікно с полем для встановлення шлаху до зображення, в яке в подальшому буде вставлено інші блоки. У лістингу 3.1 зображено приклад створення вікна додатку. При запуску скомпільованої java програми ми отримаємо вікно заданого розміру з можливістю його масштабування згортання та закриття. Так як повою розробки є java то створювальний додаток буде виглядати однаково незалежно від операційної системи на якій його було запущено. На рисунку 3.1 зображено те як виглядає це вікно.

Кожна десктопна програма повинна мати вікно. У цьому вікні будуть розміщуватись всі необхідні компоненти графічного інтерфейсу користувача, за допомогою яких користувач буде працювати з додатком. Спробуємо зробити вікно нашої програми та розмістити там щось. Поки що наша програма буде простою і нехитрою. До сформування вікна докладання в Swing використовується клас JFrame. Він містить ряд методів та властивостей, які дозволяють налаштувати його належним чином.

При побудові інтерфейсів потрібні компоненти-контерйнери, які будуть містити інші компоненти інтерфейсу користувача. У Swing одним із таких компонентів-контейнерів є JPanel. За замовчуванням JPanel сама по собі нічого не малює за винятком фону. При роботі з контейнерами розробнику треба вирішити, як правило, дві основні проблеми. Перша – задати розташування дочірніх компонентів та друга – здійснити додавання компонентів на контейнер.

Розглянемо першу проблему – завдання розташування дочірніх компонентів. Swing має механізм Layout Manager'ів для завдання структури розташування дочірніх компонентів на контейнері. Крім того, Layout Manager визначає те, як будуть реагувати компоненти на зміну розмірів батьківського контейнера. Задається Layout Manager для контейнера за допомогою методу setLayout. Друга проблема – додавання компонента. Додавання компонента виконується за допомогою методу add. Як параметри цього методу передається компонент, що додається, а також і інші об'єкти, відповідальні за розташування компонента.

JLabel це мабуть найпростіший компонент, який є Java Swing. За допомогою JLabel можна показати текст із іконкою. Якщо потрібний компонент, щоб відобразити якесь повідомлення користувачеві або зробити для поля введення текстову мітку, або показати іконку - використовуємо JLabel. Текст, який показує JLabel не можна виділяти, тільки дивитися.

Для створення об'єкта JLabel, який потім розташовуватимемо у вікні, можна скористатися конструктором з рядковим параметром public JLabel(String text). Параметр — це текст, який буде відображатися в JLabel. Крім того, текст, який буде відображатися в JLabel, можна встановити за допомогою методу setText. Єдиним параметром методу є рядок тексту String, що відображається.

JLabel дозволяє настроювати шрифт, який буде використовуватись для відображення тексту. Встановлення шрифту відбувається за допомогою методу setFont класу JLabel.

Кожна програма, яка має графічний інтерфейс користувача, не може обходитися без кнопок. У Java Swing кнопку представлена класом JButton. У кнопки є різні методи її конфігурування — установка написи на JButton, установка іконки, вирівнювання тексту, установка розмірів тощо. Крім усього іншого, розробнику необхідно навісити на JButton слухача, який буде виконуватися як тільки користувач натисне кнопку. Як тільки користувач натискає кнопку, створюється ActionEvent подія, яка передається слухачам кнопки. Для того, щоб організувати слухача, Swing надає інтерфейс ActionListener, який необхідно реалізувати. Інтерфейс ActionListener вимагає лише реалізації одного методу - actionPerformed. Після того, як обробник створений, його потрібно додати до кнопки. Робиться це за допомогою методу addActionListener. Як параметр методу передається обробник.

При роботі з файлами з програми виникає потреба рано чи пізно використовувати діалог для вибору файлів. JFileChooser з бібліотеки Java Swing якраз є таким діалогом. Крім того, діалог JFileChooser дозволяє робити навігацію по файловій системі. JFileChooser тільки надає можливість вибору файлу або директорії, що більше нічого з ними не робить. все інше – завдання розробника. Працювати з ним просто та легко. За допомогою методу getSelectedFile ми отримуємо посилання на об'єкт File і продовжуємо з ним працювати.

Лістинг 3.1- Код що описує створення вікна додатку

JFrame frame = new JFrame();

frame.setPreferredSize(new Dimension(1000, 1000));

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(new BorderLayout());

JPanel panelFile = new JPanel(new GridLayout(4,2));

panelFile.setBackground(Color.LIGHT\_GRAY);

panelFile.setPreferredSize(new Dimension(1000, 130));

panelFile.setLayout(new BoxLayout(panelFile, BoxLayout.Y\_AXIS));

panelFile.add(Box.createVerticalGlue());

final JLabel label = new JLabel("Selected file");

label.setAlignmentX(CENTER\_ALIGNMENT);

panelFile.add(label);

panelFile.add(Box.createRigidArea(new Dimension(10, 10)));

JButton button = new JButton("Review");

button.setAlignmentX(0);

panelFile.add(button);

panelT = new JPanel(new GridLayout(4,2));

panelFile.add(panelT);

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Открыть файл");

if (ret == JFileChooser.APPROVE\_OPTION) {

File file = fileopen.getSelectedFile();

label.setText(file.getPath());

panelT.removeAll();

RenderGui renderGui = new RenderGui(frame, file, panelT);

renderGui.renderImage();

}

}

});

frame.getContentPane().add(panelFile, BorderLayout.PAGE\_START);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

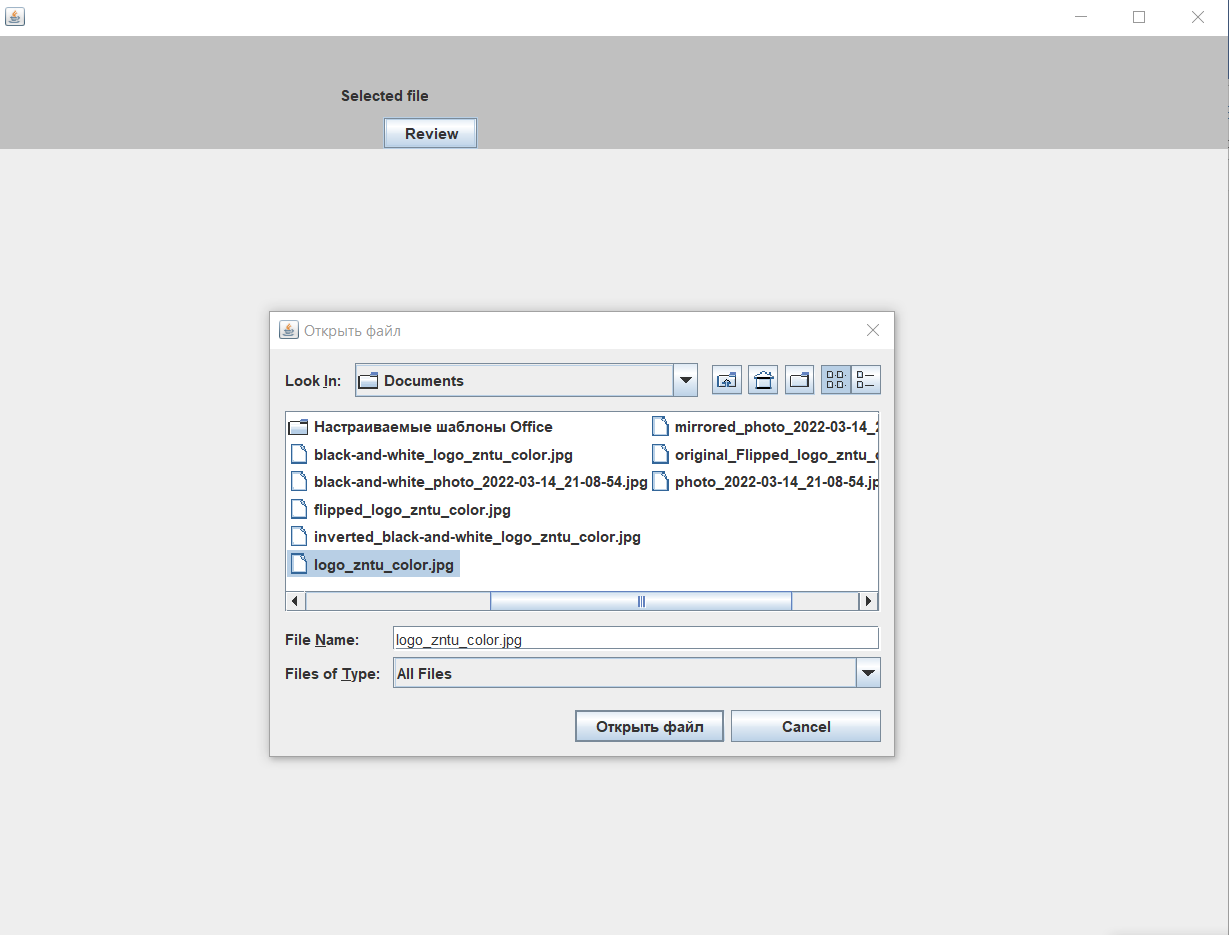


Рисунок 3.1 – Вигляд вікна яке було створено та поля з встановленням шляху

Тепер розглянемо додаток з програмного боку. Для обробки зображень використовується пакет java.awt та його клас BufferedImage, за його допомогою зчитане зображення представляється у вигляді байтового масиву який зберігається в оперативній пам’яті і задопомогою AffineTransform виконуються такі перетворення як: повертання на 90/180 градусів, а також за допомогою Graphics2D можна відзеркалити зображення по горизонталі, використавши метод Graphics2D.drawImage та передавши до нього зображення та його висоту та ширину зі знаком мінус для того, щоб його розвернуло. У лістингу 3.2 зображенно шматок коду в якому було використано дані методи та класи.

Лістинг 3.2 – Шматок коду який відповідає за обробку зображень

public BufferedImage createFlipped(BufferedImage image) {

AffineTransform at = new AffineTransform();

at.concatenate(AffineTransform.getScaleInstance(1, -1));

at.concatenate(AffineTransform.getTranslateInstance(0, -image.getHeight()));

return createTransformed(image, at, image.getWidth(), image.getHeight());

}

public BufferedImage createMirrored(BufferedImage image) {

BufferedImage newImage = new BufferedImage(

image.getWidth(), image.getHeight(),

BufferedImage.TYPE\_INT\_ARGB);

Graphics2D g = newImage.createGraphics();

g.drawImage(image, 0 + image.getWidth(), 0, -image.getWidth(), image.getHeight(), null);

g.dispose();

return newImage;

}

public BufferedImage createRotated(BufferedImage image) {

AffineTransform at = AffineTransform.getRotateInstance(

Math.PI / 2, image.getWidth() / 1.5, image.getHeight() / 2);

return createTransformed(image, at, image.getHeight(), image.getWidth());

}

Також було використано конвертування байтового масиву у масив типу int для виконання деяких математичних махінацій при переведенні зображення у чорно/білий формат. Спочатку байтовий масив за допомогою двой масивів переводиться у багато вимірний масив чисел, кожен елемент якого описує певний піксель зображення, після чого ми знаходимо середню арифметичну інтенсивність пікселя за всіма кольорами просумуючи 3 діапазони кольорів та поділивши та 3, ми отримаємо новий колір покселю, якій в наступному записуємо її в кожен колір, зсуваючи байти RGB на свої місця. Після чого ми отримуємо оброблене зображення у вигляді масиву чисел та конвертуємо його у масив байт який у подальшому буде збережено або відображено. У лістингу 3.3 зображено код який було описано вище.

Лістинг 3.2 – Шматок коду який відповідає за перетворення зображення у чорно/біле

public int[] copyFromBufferedImage(BufferedImage img) {

int[] pict = new int[img.getHeight() \* img.getWidth()];

for (int i = 0; i < img.getHeight(); i++)

for (int j = 0; j < img.getWidth(); j++)

pict[i \* img.getWidth() + j] = img.getRGB(j, i) & 0xFFFFFF; // 0xFFFFFF: записываем только 3 младших байта RGB

return pict;

}

public BufferedImage convertToBlackAndWhite(BufferedImage img) {

int height = img.getHeight();

int width = img.getWidth();

int[] pixels = copyFromBufferedImage(img);

for (int i = 0; i < height; i++)

for (int j = 0; j < width; j++) {

int intens = (getRed(pixels[i \* width + j]) +

getGreen(pixels[i \* width + j]) +

getBlue(pixels[i \* width + j])) / 3;

pixels[i \* width + j] = intens + (intens << 8) + (intens << 16);

}

BufferedImage bi = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

for (int i = 0; i < height; i++)

{

for (int j = 0; j < width; j++)

{

bi.setRGB(j, i, pixels[i \* width + j]);

}

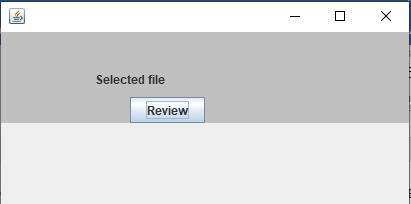
}

return bi;

}

### 3.4 Огляд можливостей розробленого додатку

Огляд почнеться с повного зображення додатку на рисунку 3.2показано те як додаток виглядає при першому запуску.



3.2 **–** Вигляд додатку при першому запуску.

При натисканню на кнопку “Review” відкриється вікно в якому можна за допомогою навігації обрати зображення яке потрібно обробити. На рисунку 4.3 зображено те як воно виглядає.

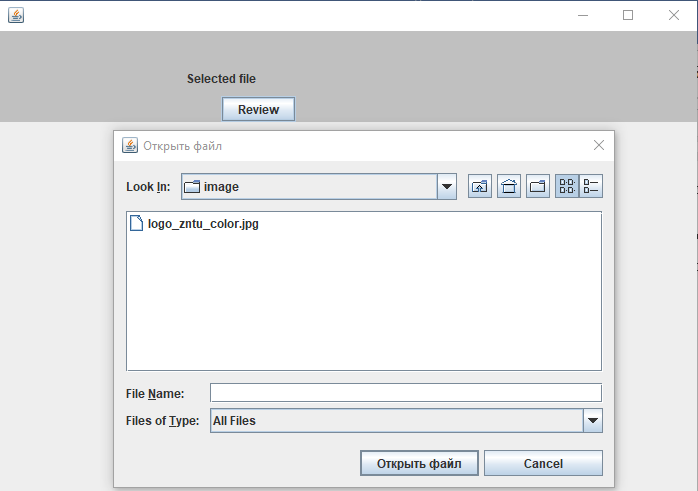


Рисунок 4.3 – Демонстрація роботи вікна файлів

Після вибору зображення знизу з’являться мініатюри з накладенеми на них фільтрами. На рисунку 3.4 зображено те як виглядає додаток після вибору зобрження.

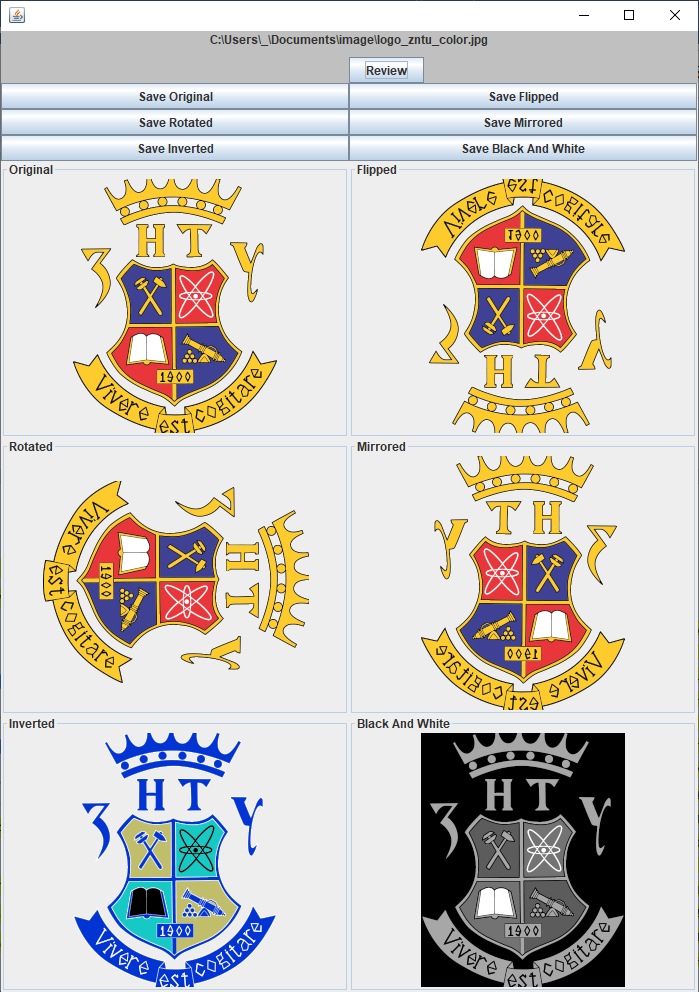


Рисунок 3.4 – Демонстрація роботи додатку

При натисканні на будь-яку кнопку save буде відображено діалогове вікно в якому буде повідомлено про успішне чи ні збереження . На рисунку 3.5 зображено те як виглядає діалогове вікно.

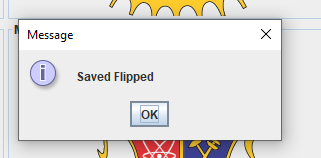


Рисунок 3.5 – Демонстрація відображення діалогового вікна

І останнім залишається зобразити те, в якому вигляді зберігається зображення. На рисунку 3.6 зображено те виглядають оброблені зображення у директорії.

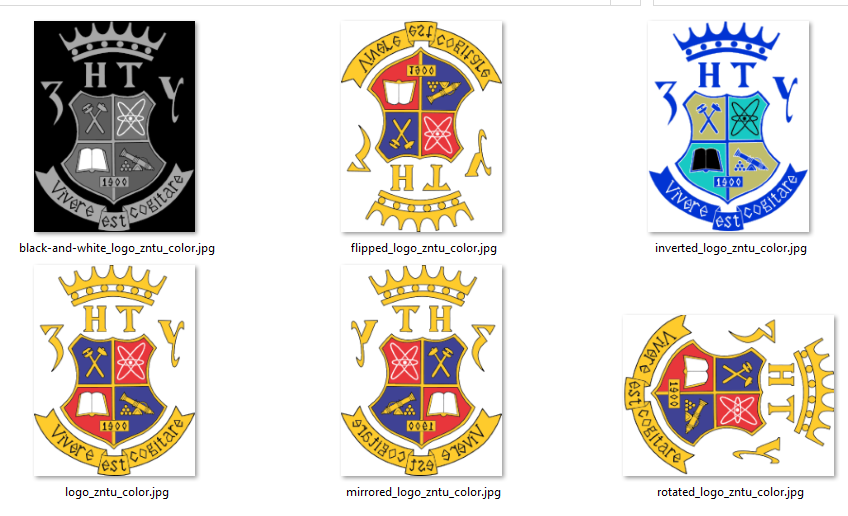


Рисунок 3.6 – Демонстрація оброблених зображень у дерикторії

### Висновок

У курсовому проекті було проведено ознайомлення з середою розробки Intelij IDE та пакетами java.awt для роботи з зображеннями та javax.swing для створення графічної оболонк. Результатом цього проекту було розроблено додаток за допомогою якого можна займатися обробкою зображень на комп’ютерах які не потужні, та такий додаток повністю вілповідає вимогам поставленим на початку роботи, так як він вміє інвертувати зображення, відображати його по горизонталі, поверстати на 90 та 180 градусів а також робити його чорнобілим. Програма не займає багато місця, не вимоглива до встановленого програмного забезпечення.

Перевагами даного додатку є:

· Можливість запустити на будь-якій операційній системі яка підтримує віртуальну машину Java та на пристрої будь-якої потужності бо продукт маловимогливий до системних ресурсів;

- Також не мало важливою перевагою є те, що розроблювальний додаток повністю безкоштовний.

Недоліком є неможливість накладання фільтрів які не було запрограмовано на стадії проектування. Але це і мінусом важко назвати, бо у будь який час невистачаючий фільтер можна додати до розробленого додатку.

В цілому, поставлена ​​на початку курсового проекту мета була досягнута. У програмі виконуються роботи з зображення за допомогою java, та додаток має зручний графічний інтерфейс.

### Перелік джерел посилань

1. Джеймс Гослінг, Білл Джой, Гай Стіл, Гілад Брача, Алекс Баклі. Мова програмування Java SE 8. Детальний опис, 5-е видання = The Java Language Specification, Java SE 8 Edition (5th Edition) (Java Series). - М. : «Вільямс» , 2015. - 672 с. - ISBN 978-5-8459-1875-8 .
2. Брюс Еккель. Філософія Java = Thinking in Java. - 4-е изд. - СПб. : Пітер , 2018. - 1168 с. - ISBN 978-5-496-01127-3 .
3. Герберт Шилдт. Полный справочник по Java SE 6 = Java: The Complete Reference. — 7-е изд. — М.: «Вильямс», 2007. — С. 1040. — ISBN 0-07-226385-7.
4. Кей С. Хорстманн. Java. Бібліотека професіонала, тому 2. Розширені засоби програмування. 10-е видання = Core Java. Volume II - Advanced Feature (Tenth Edition). - М. : «Вільямс» , 2017. - 976 с. - ISBN 978-5-9909445-0-3 .
5. Бенджамін Дж. Еванс, Джеймс Гоф, Кріс Ньюленд. Java: оптимізація програм. Практичні методи підвищення продуктивності додатків в JVM. - М .: Діалектика, 2019. - 448 с. - ISBN 978-5-907114-84-5.
6. Герберт Шілдт. Java. Повне керівництво, 10-е видання = Java. The Complete Reference, 10th Edition. - М .: «Діалектика», 2018. - 1488 с. - ISBN 978-5-6040043-6-4.Роджерс Р., Ломбардо Д. Android. Розробка додатків. - М. : ЕКОМ Паблишерз , 2010. - 400 с.- ISBN 978-5-9790-0113-5.
7. Голощапов А. Google Android: програмування для мобільних пристроїв. - СПб. : БХВ-Петербург , 2010. - 448 с. - ISBN 978-5-9775-0562-8 .
8. Донн Фелкер. Android: розробка додатків для чайників = Android Application Development For Dummies. - М. : Діалектика , 2011. - 336 с. - ISBN 978-5-8459-1748-5 .
9. Іан Соммервілла. Інженерія програмного забезпечення = Software Engineering. - 6-е изд. - М. : «Вільямс» , 2002. - С. 642. - ISBN 5-8459-0330-0 .

### Додаток А

Лістинг А.1 – Код файлу Main.java

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import javax.swing.\*;

public class Main extends JFrame

{

private static JPanel panelT;

public static void main(String[] args)

{

SwingUtilities.invokeLater(new Runnable()

{

@Override

public void run()

{

createAndShowGUI();

}

});

}

private static void createAndShowGUI()

{

JFrame frame = new JFrame();

frame.setPreferredSize(new Dimension(1000, 1000));

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(new BorderLayout());

JPanel panelFile = new JPanel(new GridLayout(4,2));

panelFile.setBackground(Color.LIGHT\_GRAY);

panelFile.setPreferredSize(new Dimension(1000, 130));

panelFile.setLayout(new BoxLayout(panelFile, BoxLayout.Y\_AXIS));

panelFile.add(Box.createVerticalGlue());

final JLabel label = new JLabel("Selected file");

label.setAlignmentX(CENTER\_ALIGNMENT);

panelFile.add(label);

panelFile.add(Box.createRigidArea(new Dimension(10, 10)));

JButton button = new JButton("Review");

button.setAlignmentX(0);

panelFile.add(button);

panelT = new JPanel(new GridLayout(4,2));

panelFile.add(panelT);

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Открыть файл");

if (ret == JFileChooser.APPROVE\_OPTION) {

File file = fileopen.getSelectedFile();

label.setText(file.getPath());

panelT.removeAll();

RenderGui renderGui = new RenderGui(frame, file, panelT);

renderGui.renderImage();

}

}

});

frame.getContentPane().add(panelFile, BorderLayout.PAGE\_START);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

}

Лістинг 1.2 – Код файлу RenderGui.java

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import javax.swing.\*;

public class Main extends JFrame

{

private static JPanel panelT;

public static void main(String[] args)

{

SwingUtilities.invokeLater(new Runnable()

{

@Override

public void run()

{

createAndShowGUI();

}

});

}

private static void createAndShowGUI()

{

JFrame frame = new JFrame();

frame.setPreferredSize(new Dimension(1000, 1000));

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(new BorderLayout());

JPanel panelFile = new JPanel(new GridLayout(4,2));

panelFile.setBackground(Color.LIGHT\_GRAY);

panelFile.setPreferredSize(new Dimension(1000, 130));

panelFile.setLayout(new BoxLayout(panelFile, BoxLayout.Y\_AXIS));

panelFile.add(Box.createVerticalGlue());

final JLabel label = new JLabel("Selected file");

label.setAlignmentX(CENTER\_ALIGNMENT);

panelFile.add(label);

panelFile.add(Box.createRigidArea(new Dimension(10, 10)));

JButton button = new JButton("Review");

button.setAlignmentX(0);

panelFile.add(button);

panelT = new JPanel(new GridLayout(4,2));

panelFile.add(panelT);

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Открыть файл");

if (ret == JFileChooser.APPROVE\_OPTION) {

File file = fileopen.getSelectedFile();

label.setText(file.getPath());

panelT.removeAll();

RenderGui renderGui = new RenderGui(frame, file, panelT);

renderGui.renderImage();

}

}

});

frame.getContentPane().add(panelFile, BorderLayout.PAGE\_START);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

}

Лістинг 1.3 – Код файлу ImageHelper.java

import javax.imageio.\*;

import java.awt.image.BufferedImage;

import java.io.\*;

import java.awt.\*;

import java.awt.geom.AffineTransform;

import java.awt.image.LookupOp;

import java.awt.image.LookupTable;

public class ImageHelper {

public int getRed(int color) {

return color >> 16;

} // получить красную составляющую цвета

public int getGreen(int color) {

return (color >> 8) & 0xFF;

} // получить зеленую составляющую цвета

public int getBlue(int color) {

return color & 0xFF;

} // получить синюю составляющую цвета

public BufferedImage convertToARGB(BufferedImage image) {

BufferedImage newImage = new BufferedImage(

image.getWidth(), image.getHeight(),

BufferedImage.TYPE\_INT\_ARGB);

Graphics2D g = newImage.createGraphics();

g.drawImage(image, 0, 0, null);

g.dispose();

return newImage;

}

public BufferedImage createFlipped(BufferedImage image) {

AffineTransform at = new AffineTransform();

at.concatenate(AffineTransform.getScaleInstance(1, -1));

at.concatenate(AffineTransform.getTranslateInstance(0, -image.getHeight()));

return createTransformed(image, at, image.getWidth(), image.getHeight());

}

public BufferedImage createMirrored(BufferedImage image) {

BufferedImage newImage = new BufferedImage(

image.getWidth(), image.getHeight(),

BufferedImage.TYPE\_INT\_ARGB);

Graphics2D g = newImage.createGraphics();

g.drawImage(image, 0 + image.getWidth(), 0, -image.getWidth(), image.getHeight(), null);

g.dispose();

return newImage;

}

public BufferedImage createRotated(BufferedImage image) {

AffineTransform at = AffineTransform.getRotateInstance(

Math.PI / 2, image.getWidth() / 1.5, image.getHeight() / 2);

return createTransformed(image, at, image.getHeight(), image.getWidth());

}

public BufferedImage createTransformed(

BufferedImage image, AffineTransform at, int width, int height) {

BufferedImage newImage = new BufferedImage(

width, height,

BufferedImage.TYPE\_INT\_ARGB);

Graphics2D g = newImage.createGraphics();

g.transform(at);

g.drawImage(image, 0, 0, null);

g.dispose();

return newImage;

}

public BufferedImage createInverted(BufferedImage image) {

if (image.getType() != BufferedImage.TYPE\_INT\_ARGB) {

image = convertToARGB(image);

}

LookupTable lookup = new LookupTable(0, 4) {

@Override

public int[] lookupPixel(int[] src, int[] dest) {

dest[0] = (int) (255 - src[0]);

dest[1] = (int) (255 - src[1]);

dest[2] = (int) (255 - src[2]);

return dest;

}

};

LookupOp op = new LookupOp(lookup, new RenderingHints(null));

return op.filter(image, null);

}

public BufferedImage convertToBlackAndWhite(BufferedImage img) {

int height = img.getHeight();

int width = img.getWidth();

int[] pixels = copyFromBufferedImage(img);

for (int i = 0; i < height; i++)

for (int j = 0; j < width; j++) {

// находим среднюю арифметическую интенсивность пикселя по всем цветам

int intens = (getRed(pixels[i \* width + j]) +

getGreen(pixels[i \* width + j]) +

getBlue(pixels[i \* width + j])) / 3;

// ... и записываем ее в каждый цвет за раз , сдвигая байты RGB на свои места

pixels[i \* width + j] = intens + (intens << 8) + (intens << 16);

}

BufferedImage bi = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

for (int i = 0; i < height; i++)

for (int j = 0; j < width; j++)

bi.setRGB(j, i, pixels[i \* width + j]);

return bi;

}

public int[] copyFromBufferedImage(BufferedImage img) {

int[] pict = new int[img.getHeight() \* img.getWidth()];

for (int i = 0; i < img.getHeight(); i++)

for (int j = 0; j < img.getWidth(); j++)

pict[i \* img.getWidth() + j] = img.getRGB(j, i) & 0xFFFFFF; // 0xFFFFFF: записываем только 3 младших байта RGB

return pict;

}

public void saveAsJpeg(File image, String type, BufferedImage buffImg) throws IOException {

try {

ImageIO.write(buffImg, "PNG", new File(image.getParent() + "\\" + type + "\_" + image.getName()));

} catch (IOException ignored) {

}

}

}