**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інженерії програмного забезпечення**

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно - орієнтоване програмування»

на тему: **Гра Сапер на мові java**

121 – КР.00 – 208.1910820

**Виконала: студентка 2 курсу, групи 208**

**спеціальності 121 «Інженерія програмного забезпечення»**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. О. Сабіна

(підпис, ініціали та прізвище)

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

(дата)

**Керівник: ст. викладач**

**Боровльова С.Ю.**

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_ \_\_\_\_\_\_\_\_

**м. Миколаїв – 2021 рік**

[ВСТУП 3](#_Toc71755866)

[Розділ 1 Теоретичні основи проектування логічної гри «Сапер» 5](#_Toc71755867)

[1.1 Поняття логічної гри 5](#_Toc71755868)

[2.2 Вибір мови програмування і середи розробки 5](#_Toc71755869)

[Розділ 2 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ 9](#_Toc71755870)

[2.1 Перерахування для відображення картинок 9](#_Toc71755871)

[2.2 Робота з координатами 10](#_Toc71755872)

[2.3 Клас для зберігання поля і декілька корисних функцій 10](#_Toc71755873)

[2.4 Фасадний клас Game 11](#_Toc71755874)

[2.5 Клас для зберігання боксів 14](#_Toc71755875)

[2.6 Створення двохвимірної матриці (бомби і прапорці) 14](#_Toc71755876)

[2.7 Відображення стану гри 16](#_Toc71755877)

[2.8 Створення верхнього шару гри 16](#_Toc71755878)

[Розділ 3 Тестування застосунку 21](#_Toc71755879)

[3.1 Представлення гри на різних етапах 21](#_Toc71755880)

[3.2 Запуск гри через архів(майже як .exe) 24](#_Toc71755881)

[висновки 25](#_Toc71755882)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 27](#_Toc71755883)

ВСТУП

В даній курсовій роботі розглядається приклад створення гри «Сапер» на мові java. Для програмної реалізації використовується IntelliJ IDEA Community Edition 2021.1. IntelliJ IDEA — комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та інші) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію.

Незважаючи на те, що IntelliJ IDEA - в першу чергу IDE для Java, вона розуміє і надає інтелектуальну допомогу при написанні коду на SQL, JPQL, HTML, JavaScript і багатьох інших мовах і дозволяє редагувати код, написаний нема на Java, всередині строкових літералів Java -коду.

Кожен компонент IntelliJ IDEA створений для того, щоб максимально підвищити продуктивність розробки. Розумний редактор коду в поєднанні з ергономічним дизайном роблять розробку не тільки ефективною, а й приємною.

Предмет дослідження - сучасні методи розробки програм таких, як об'єктно-орієнтоване програмування та візуальне проектування, а також структурний і модульне програмування.

Мета курсової роботи - систематизація, поглиблення і активне застосування знань з програмування, закріплення знань, отриманих в лекційному курсі, а також на практичних і лабора-торних заняттях.

Метод дослідження - вивчення літератури, складання і налагодження програм на комп'ютері.

Дана програма використовується для цікавого проведення часу за грою, яка вимагає розвиненого логічного мислення

Для досягнення поставленої мети необхідно виконати наступні завдання:

* Ознайомитися з правилами гри «Сапер»
* Обрання мови програмування
* Пошук бібліотек необхідних для розробки
* Підготовлення проекту з картинками
* Розробка моделей та функціоналу
* Тестування готового продукту

Курсова робота складається з 29 сторінок, містить 6 ілюстрацій, 5 літературних джерел.

Розділ 1   
Теоретичні основи проектування логічної гри «Сапер»

1.1 Поняття логічної гри

Сапер являє собою логічну гру, основною метою якої є знаходження всіх захованих бомб на мінному полі. Ваша задача відкрити всі осередки поля, що не містять бомб, заблокував (позначивши) при цьому осередку, в яких розташовані бомби.

Поле гри задано у вигляді двомірного масиву. У цьому масиві рас-покладені осередки. Спочатку вони всі закриті. Осередки можуть бути пу-стими, з цифрами і з бомбами. Для кожної такої комірки ми задаємо клас. При натисненні лівої кнопки миші відкривається осередок, при натисканні правої кнопки, виставляється прапорець, при цьому лівою кнопкою миші ви вже не можете натиснути на заблоковану осередок, але правою кнопкою можна зняти позначку прапорця. Осередок з цифрою позначає те, скільки бомб знаходиться в окрузі цього осередку. При першому натисканні кнопки миші на осередок не може відкритися осередок з бомбою.

Гра вважається програною, якщо ви відкрили осередок з бомбою.

Гра вважається виграною, якщо на ігровому полі все осередки з бом-бами відзначені і всі інші осередки - відкриті.

2.2 Вибір мови програмування і середи розробки

При проектуванні даної роботи стало питання про метод реалізації програми. Необхідно було вибрати мову програмування, яка би надала необхідні можливості для швидкого створення гри Сапер, і в той же час була досить ефективною, щоб писати швидкі програми. Так як Java повністю задовольняє цим вимогам, і була мною вивчений більш повно, ніж C++ і C#, я вибрала саме її. Тепер розглянемо чим же приваблива Java.

Java розробила компанія Sun Microsystems на початку 90-х років XX століття. Провідну роль у створенні мови зіграв канадський інженер Джеймс Гослінг (James Gosling). На ранніх етапах розробки мова називалася Oak. Потім його перейменували в честь сорти кави Java. Зв'язок мови з напоєм відбивається в логотипі.

Джеймс Гослінг і його однодумці хотіли створити мову з си-подібним синтаксисом. У той же час він повинен бути простішим у порівнянні з C / C ++. Творці планували використовувати Java для програмування побутової електроніки. Проте практично відразу після випуску версії 1.0 в 1995 мова стали використовувати розробники серверного та клієнтського ПЗ.

Назва мови читається як «джава». Однак російськомовні користувачі в розмовній і навіть в письмовій мові іноді говорять «мову програмування ява». Це один із прикладів використання сленгу.

У 2010 році компанію Sun Microsystems купила Oracle. Після цього Джеймс Гослінг перейшов в Google, звідки теж незабаром звільнився.

Особливості мови Java

Java - мова програмування загального призначення. Відноситься до об'єктно-орієнтованим мовам програмування, до мовам з сильною типізацією.

Java є об'єктно-орієнтованою мовою, відноситься до мов програмування з сильною типізацією.

Творці реалізували принцип WORA: write once, run anywhere або «пиши один раз, запускай скрізь». Це означає, що написане на Java додаток можна запустити на будь-якій платформі, якщо на ній встановлена ​​середовище виконання Java (JRE, Java Runtime Environment).

Це завдання вирішується завдяки компіляції написаного на Java коду в байт-код. Цей формат виконує JVM або віртуальна машина Java. JVM - частина середовища виконання Java (JRE). Віртуальна машина не залежить від платформи.

В Java реалізований механізм управління пам'яттю, який називається складальником сміття або garbage collector. Розробник створює об'єкти, а JRE за допомогою збирача сміття очищає пам'ять, коли об'єкти перестають використовуватися. Пояснює експерт Микита Липський: «Є таке поняття - циклічний сміття. Всередині циклу на всі об'єкти є посилання, однак garbage collector в Java видалить його, якщо об'єкти не можуть використовуватися з програми ».

**Популярність Java**

Ви вже знаєте, що за результатами рейтингу State of Octoverse 2020 Java входить до трійки найбільш популярний мов програмування. «Джава» випереджає за популярністю PHP, C #, C ++, TypeScript і інші затребувані мови, а поступається тільки JavaScript і Python.

В індексі TIOBE на березень 2021 року Java займає друге місце. Індекс TIOBE - індикатор популярності мов програмування, який розраховується за складною методикою з урахуванням кількості пошукових запитів, які стосуються тій чи іншій мові. При цьому перше місце в цьому індексі займає мову програмування C.

У рейтингу RedMonk за червень 2020 року Java займає третє місце, поступаючись JavaScript і Python. Цей рейтинг розраховується на основі кількості репозиторіїв на GitHub, як і State of Octoverse, проте методика ранжирування тут відрізняється. Наприклад, RedMonk не враховує в розрахунках Форк репозиторіїв.

Тепер розглянемо більш детально середу розробки, в якій розроблявся даний проект. Програмне забезпечення JetBrains IntelliJ IDEA - це провідна середовище швидкої розробки на мові Java. IntelliJ IDEA являє собою високотехнологічний комплекс тісно інтегрованих інструментів програмування, що включає інтелектуальний редактор вихідних текстів з розвиненими засобами автоматизації, потужні інструменти рефакторинга коду, вбудовану підтримку технологій J2EE, механізми інтеграції з середовищем тестування Ant / JUnit і системами управління версіями, унікальний інструмент оптимізації та перевірки коду Code Inspection, а також інноваційний візуальний конструктор графічних інтерфейсів.

Унікальні можливості JetBrains IntelliJ IDEA позбавляють програміста від вантажу рутинної роботи, допомагають своєчасно усунути помилки і підвищити якість коду, піднімаючи продуктивність розробника на нову висоту.

**Безкоштовна версія найрозумнішою середовища розробки на основі відкритого коду включає в себе:**

* Розумне автодоповнення, інструменти для аналізу якості коду, зручна навігація, розширені рефакторингом і форматування для Java, Groovy, Scala, Clojure і Erlang.
* Професійний набір інструментів для розробки Android-додатків.
* Підтримка JavaFX 2.0, інтеграція з SceneBuilder; Дизайнер інтерфейсу для Swing.
* Інтеграція з автоматизованими інструментами побудови та управління проектом, включа Maven, Gradle, Ant та іншими.
* Інструменти для тестування з підтримкою JUnit, TestNG, Spock, ScalaTest і spec2.
* Інтеграція з системами управління версіями, включаючи Git, Subversion, Mercurial і CSV.

Розділ 2   
ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

Почнемо з планування нашого проекту. Ми використовуємо картинки в нашому проекті і їх можна поділити на дві категорії:

* Нижній шар (bombMap);
* Верхній шар (flagMap).

2.1 Перерахування для відображення картинок

Для того щоб працювати з цими картинками в нас буде створене перерахування enum Box в котрому ми запишемо всі картинки нижнього і верхнього шару. Також для нашего перерахування ми створимо Object image(якась картинка, яка потрібна нам для їх відображення. Далі ми створимо дві функції :

* int getNumber() – метод, щоб дізнаватися яка цифра там знаходиться(тобто скільки клітинок навколо);
* Box nextNumberBox()-функція яка переходить з одного значення в інше.

package sweeper;  
  
public enum Box {  
 *ZERO*,  
 *NUM1*,  
 *NUM2*,  
 *NUM3*,  
 *NUM4*,  
 *NUM5*,  
 *NUM6*,  
 *NUM7*,  
 *NUM8*,  
 *BOMB*,  
 *OPENED*,  
 *CLOSED*,  
 *FLAGED*,  
 *BOMBED*,  
 *NOBOMB*;  
  
 public Object image;  
  
 Box getNextNumberBox()  
 {  
 return Box.*values*()[this.ordinal() + 1];  
 }  
  
 public int getNumer()  
 {  
 return this.ordinal();  
 }  
}

2.2 Робота з координатами

Далі створимо клас **Coord** для зберігання координат. В цьому класі є дві змінні int x, int y, конструктор Coord(int x, int y) і метод Boolean equals (Object o) , щоб порівнювати координати.

package sweeper;  
  
public class Coord  
{  
 public int x;  
 public int y;  
  
 public Coord(int x,int y)  
 {  
 this.x=x;  
 this.y=y;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if(o instanceof Coord)  
 {  
 Coord to=(Coord) o;  
 return to.x==x && to.y==y;  
 }  
 return super.equals(o);  
  
  
 }  
}

2.3 Клас для зберігання поля і декілька корисних функцій

Наступним кроком буде створення класу **Ranges**. Створимо змінну static Coord size, вона буде зберігати розміри по x та по y. Далі будемо перебирати всі координати у списку за допомогою static List<Coord> allCoords. Згодом нам знадобиться генератор static Random random, також сеттери і геттери(static setSize(Coord size), static Coord getSize()), отримання списку всіх координат- static List<Coord> getAllCoords(), функція отримання координатів навколо рандомної клітинки static Coord getRandomCoord(), функція, яка домогає дізнатися чи яка-небудь клітинка знаходиться в нашому полі.

package sweeper;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
public class Ranges  
{  
 private static Coord *size*;  
 private static ArrayList<Coord> *allCoords*;  
 private static Random *random*=new Random();  
  
  
 public static void setSize(Coord \_size)  
 {  
 *size*=\_size;  
 *allCoords*=new ArrayList<Coord>();  
 for(int y=0;y<*size*.y;y++)  
 for(int x=0;x<*size*.x;x++)  
 *allCoords*.add(new Coord(x,y));  
  
 }  
 public static Coord getSize()  
 {  
 return *size*;  
 }  
 public static ArrayList<Coord> getAllCoords()  
 {  
 return *allCoords*;  
 }  
 static boolean inRange(Coord coord){  
 return coord.x>=0 && coord.x< *size*.x &&  
 coord.y>=0 && coord.y< *size*.y;  
 }  
 static Coord getRandomCoord()  
 {  
 return new Coord(*random*.nextInt(*size*.x),  
 *random*.nextInt(*size*.y));  
 }  
 static ArrayList<Coord> getCoordsAround(Coord coord)  
 {  
 Coord around;  
 ArrayList<Coord> list=new ArrayList<Coord>();  
 for(int x= coord.x-1;x<= coord.x+1;x++)  
 for(int y= coord.y-1;y<= coord.y+1;y++)  
 if(*inRange*(around=new Coord(x,y)))  
 if(!around.equals(coord))  
 list.add(around);  
 return list;  
  
 }  
}

2.4 Фасадний клас Game

Далі створюємо клас **Game** за допомогою якого ми будемо управляти всіма процесами в грі. Там буде матриця Bomb bomb, Flag flag, GameState gameState-стан поточної гри (виграв чи програв), публічні методи:

* Game(int cols,int rows,int bombs)-конструктор, котрому ми передамо розміри нашого екрану і кількість бомб.
* void start()-для запуску гри.
* Box getBox (Coord coord)-метод для отримання, який бокс в вказаних координатах де знаходиться .
* void pressLeftButton(Coord coord)-ліва кнопка.
* void pressRightButton(Coord coord)-права кнопка.
* GameState getGameState()-метод для отримання стану гри.
* int getTotalBombs()-скільки всього бомб.
* int getTotalFlags()-скільки всього флагів.

package sweeper;  
  
public class Game  
  
{  
 private Bomb bomb;  
 private Flag flag;  
 private GameState state;  
  
 public GameState getState()  
 {  
 return state;  
 }  
  
  
 public Game(int cols,int rows,int bombs)  
 {  
 Ranges.*setSize*(new Coord(cols,rows));  
 bomb=new Bomb(bombs);  
 flag=new Flag();  
  
 }  
 public void start()  
 {  
 bomb.start();  
 flag.start();  
 state=GameState.*PLAYED*;  
  
 }  
 public Box getBox(Coord coord)  
 {  
 if(flag.get(coord)==Box.*OPENED*)  
 return bomb.get(coord);  
 else  
 return flag.get(coord);  
  
  
 }  
  
 public void pressLeftButton(Coord coord)  
 {  
 if (gameOver()) return;  
 openBox(coord);  
 checkWinner();  
  
 }  
  
 private void checkWinner()  
 {  
 if(state==GameState.*PLAYED*)  
 if(flag.getCountOfClosedBoxes()==bomb.getTotalBombs())  
 state=GameState.*WINNER*;  
 }  
 private void openBox(Coord coord)  
 {  
 switch (flag.get(coord))  
 {  
 case *OPENED*:setOpenedToClosedBoxesAroundNumber(coord);return;  
 case *FLAGED*:return;  
 case *CLOSED*:  
 switch (bomb.get(coord))  
 {  
 case *ZERO*:openBoxesAround(coord);return;  
 case *BOMB*:openBombs(coord);return;  
 default :flag.setOpenedToBox(coord);return;  
 }  
 }  
 }  
  
 private void setOpenedToClosedBoxesAroundNumber(Coord coord)  
 {  
 if(bomb.get(coord)!=Box.*BOMB*)  
 if(flag.getCountOfFlagedBoxesAround(coord)==bomb.get(coord).getNumer())  
 for(Coord around:Ranges.*getCoordsAround*(coord))  
 if(flag.get(around)==Box.*CLOSED*)  
 openBox(around);  
 }  
  
 private void openBombs(Coord bombed)  
 {  
 state=GameState.*BOMBED*;  
 flag.setBombedToBox(bombed);  
 for(Coord coord:Ranges.*getAllCoords*())  
 if(bomb.get(coord)==Box.*BOMB*)  
 flag.setOpenedToClosedBox(coord);  
 else  
 flag.setNobombToFlagedSafeBox(coord);  
  
 }  
  
 private void openBoxesAround(Coord coord)  
 {  
 flag.setOpenedToBox(coord);  
 for(Coord around:Ranges.*getCoordsAround*(coord))  
 openBox(around);  
 }  
  
 public void pressRightButton(Coord coord)  
 {  
 if (gameOver()) return;  
 flag.toggleFlagedToBox(coord);  
 }  
  
 private boolean gameOver()  
 {  
 if(state==GameState.*PLAYED*)  
 return false;  
 start();  
 return true;  
 }  
}

2.5 Клас для зберігання боксів

Далі створюємо **клас Matrix** для зберігання всіх наших боксів. Створюємо двовимірний масив Box [] [] matrix. Також буде конструктор, який /coord)-отримання, що знаходиться в тому чи іншому місці, void set (Coord coord, Box box).

package sweeper;  
  
class Matrix  
{  
 private Box [] [] matrix;  
 Matrix(Box defaultBox)  
 {  
 matrix=new Box[Ranges.*getSize*().x][Ranges.*getSize*().y];  
 for (Coord coord :Ranges.*getAllCoords*())  
 matrix[coord.x][coord.y]=defaultBox;  
 }  
 Box get(Coord coord)  
 {  
 if(Ranges.*inRange*(coord))  
 return matrix [coord.x][coord.y];  
 return null;  
  
 }  
 void set (Coord coord,Box box)  
 {  
 if(Ranges.*inRange*(coord))  
 matrix [coord.x][coord.y]=box;  
 }  
}

2.6 Створення двохвимірної матриці (бомби і прапорці)

Далі створюємо **клас Bomb** для роботи з нашими бомбами і масив з бомбами Matrix bombMap, змінна в якій зберігається загальна кількість бомб int totalBombs, Bomb(int totalBombs)-конструктор, який ініціалізує, встановлює кількість бомб, void start()-метод, який розставляє бомби, Box get (Coord coord)- гетер, щоб дізнатися що знаходиться в тому чи іншому місці, int geTotalBombs() -для кількості бомб, метод для фіксації кількості бомб-void fixBombsCount(), метод для розміщення бомб -void placeBombs(), який включає в себе два методи для розміщення бомб-void placeBomb(), void incNumbersAroundBombs(Coord coord)-збільшення чисел навколо бомби.

package sweeper;  
  
class Bomb  
{  
 private Matrix bombMap;  
 private int totalBombs;  
  
 Bomb(int totalBombs)  
 {  
 this.totalBombs=totalBombs;  
 fixBombCount();  
 }  
 void start()  
 {  
 bombMap=new Matrix(Box.*ZERO*);  
 for(int j=0;j<totalBombs;j++)  
 placeBomb();  
  
 }  
  
 Box get(Coord coord)  
 {  
 return bombMap.get(coord);  
 }  
 private void fixBombCount()  
 {  
 int maxBombs=Ranges.*getSize*().x\*Ranges.*getSize*().y/2;  
 if(totalBombs>maxBombs)  
 totalBombs=maxBombs;  
 }  
 private void placeBomb()  
 {  
  
 while(true){  
 Coord coord=Ranges.*getRandomCoord*();  
 if(Box.*BOMB*==bombMap.get(coord))  
 continue;  
 bombMap.set(coord,Box.*BOMB*);  
 incNumbersAroundBomb(coord);  
 break;  
 }  
  
 }  
  
 private void incNumbersAroundBomb(Coord coord)  
 {  
  
 for(Coord around :Ranges.*getCoordsAround*(coord))  
 if(Box.*BOMB* != bombMap.get(around))  
 bombMap.set(around,bombMap.get(around).getNextNumberBox());  
 }  
  
 int getTotalBombs()  
 {  
 return totalBombs;  
 }  
}

2.7 Відображення стану гри

Створимо перерахування для відображення стану гри **enum GameState.** В нас буде три типу стану, а саме:

* PLAYED-ми ще граємо;
* BOMBED-ми програли;
* WINNER-ми виграли.

package sweeper;  
  
public enum GameState  
{  
 *PLAYED*,  
 *BOMBED*,  
 *WINNER*}

2.8 Створення верхнього шару гри

Останнім класом ми створюємо **class Flag**, який буде влючати в себе нестапне:

* Matrix flagMap-матриця;
* int totalFlags-кількість флагів всього;
* int countOfClosedBoxes-кількість закритих боксів;
* void start()-функція для запуску;
* Box get (Coord coord)-функція для отримання тієї чи іншої картинки;
* void setOpenedToBox (Coord coord)-установка відкритого боксу;
* void setBombedToBox (Coord coord)-установка боксу, коли програв;
* void toggleFlagedToBox (Coord coord)-функція для встановлення прапорця або для прибирання;
* void setNobombToFlagedSafeBox(Coord coord)-функція для встановлення, що бомби немає ;
* void setOpenedToClosedBombBox(Coord coord)-функція для відкриття не закритих боксів з бомбами;
* int getCountOfFlagedBoxesAround(Coord coord)-функція для підрахування прапорців між всіма клітинками;
* int getCountOfClosedBoxes()-функція для підрахування скільки всього закритих клітинок;
* int getTotalFlags()-функція для підрахування скільки всього прапорців.

package sweeper;  
  
public class Flag  
{  
  
 private Matrix flagMap;  
 private int countOfClosedBoxes;  
  
  
 void start()  
 {  
 flagMap=new Matrix(Box.*CLOSED*);  
 countOfClosedBoxes=Ranges.*getSize*().x\*Ranges.*getSize*().y;  
  
 }  
 Box get(Coord coord)  
 {  
 return flagMap.get(coord);  
  
 }  
  
 void setOpenedToBox(Coord coord)  
 {  
 flagMap.set(coord, Box.*OPENED*);  
 countOfClosedBoxes--;  
 }  
 public void toggleFlagedToBox(Coord coord)  
 {  
 switch(flagMap.get(coord)){  
 case *FLAGED*:setClosedToBox(coord);break;  
 case *CLOSED*:setFlagedToBox(coord);break;  
  
 }  
 }  
  
 private void setClosedToBox(Coord coord)  
 {  
 flagMap.set(coord,Box.*CLOSED*);  
 }  
  
 private void setFlagedToBox(Coord coord)  
 {  
 flagMap.set(coord,Box.*FLAGED*);  
 }  
  
 int getCountOfClosedBoxes()  
 {  
 return countOfClosedBoxes;  
 }  
  
 void setBombedToBox(Coord coord)  
 {  
 flagMap.set(coord,Box.*BOMBED*);  
 }  
  
 void setOpenedToClosedBox(Coord coord)  
 {  
 if(flagMap.get(coord)==Box.*CLOSED*)  
 flagMap.set(coord,Box.*OPENED*);  
 }  
  
 void setNobombToFlagedSafeBox(Coord coord)  
 {  
 if(flagMap.get(coord)==Box.*FLAGED*)  
 flagMap.set(coord,Box.*NOBOMB*);  
 }  
  
  
 int getCountOfFlagedBoxesAround(Coord coord) {  
 int count=00;  
 for(Coord around:Ranges.*getCoordsAround*(coord))  
 if(flagMap.get(around)==Box.*FLAGED*)  
 count++;  
 return count;  
 }  
}

Основний клас JavaSweeper у якому все реалізовано.

import sweeper.Box;  
import sweeper.Coord;  
import sweeper.Game;  
import sweeper.Ranges;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
public class JavaSweeper extends JFrame  
  
{  
 private Game game;  
 private JPanel panel;  
 private JLabel label;  
 private final int COLS=9;  
 private final int ROWS=9;  
 private final int BOMBS=10;  
 private final int IMAGE\_SIZE=50;  
  
 public static void main(String[] args)  
 {  
 new JavaSweeper();  
 }  
 private JavaSweeper()  
 {  
 game=new Game(COLS,ROWS,BOMBS);  
 game.start();  
 setImages();  
 initLabel();  
 initPanel();  
 initFrame();  
  
  
 }  
 private void initLabel()  
 {  
 label=new JLabel("Welcome!");  
 add(label,BorderLayout.*SOUTH*);  
 }  
 private void initPanel()  
 {  
 panel=new JPanel()  
 {  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 for(Coord coord:Ranges.*getAllCoords*())  
 g.drawImage((Image) game.getBox(coord).image,  
 coord.x\*IMAGE\_SIZE,coord.y\*IMAGE\_SIZE, this);  
 }  
 };  
  
 panel.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mousePressed(MouseEvent e)  
 {  
 int x=e.getX()/IMAGE\_SIZE;  
 int y=e.getY()/IMAGE\_SIZE;  
 Coord coord=new Coord(x,y);  
 if(e.getButton()==MouseEvent.*BUTTON1*)  
 game.pressLeftButton(coord);  
 if(e.getButton()==MouseEvent.*BUTTON3*)  
 game.pressRightButton(coord);  
 if(e.getButton()==MouseEvent.*BUTTON2*)  
 game.start();  
 label.setText(getMessage());  
 panel.repaint();  
 }  
 });  
 panel.setPreferredSize(new Dimension(  
 Ranges.*getSize*().x\*IMAGE\_SIZE,  
 Ranges.*getSize*().y\*IMAGE\_SIZE));  
 add(panel);  
 }  
  
 private String getMessage()  
 {  
 switch (game.getState()){  
 case *PLAYED*:return "Think twice!";  
 case *BOMBED*:return "YOU LOSE!BIG BA-DA-BOOM!";  
 case *WINNER*:return "CONGRATULATIONS";  
 default:return "WELCOME";  
 }  
 }  
  
 private void initFrame()  
 {  
  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 setTitle("Java Sweeper");  
 setResizable(false);  
 setVisible(true);  
 pack();  
 setLocationRelativeTo(null);  
 setIconImage(getImage("icon"));  
 }  
 private void setImages()  
 {  
 for(Box box: Box.*values*())  
 box.image=getImage(box.name());  
 }  
  
 private Image getImage(String name)  
 {  
 String filename="img/"+name.toLowerCase()+".png";  
 ImageIcon icon=new ImageIcon(getClass().getResource(filename));  
 return icon.getImage();  
 }  
  
}

# Розділ 3 Тестування застосунку

3.1 Представлення гри на різних етапах

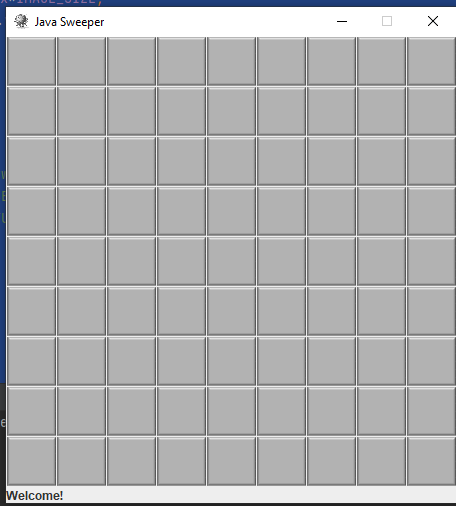


Рисунок 3.1 – Початок гри

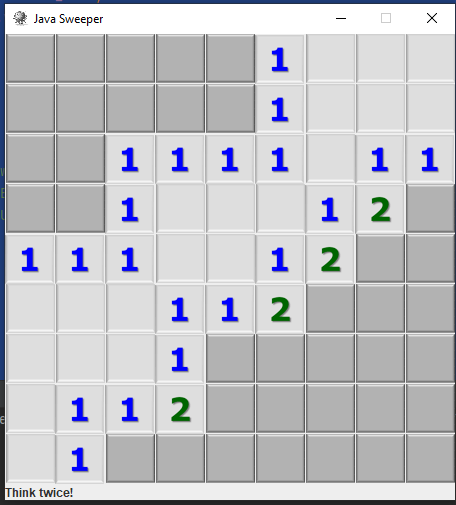


Рисунок 3.2 – Гра розпочата

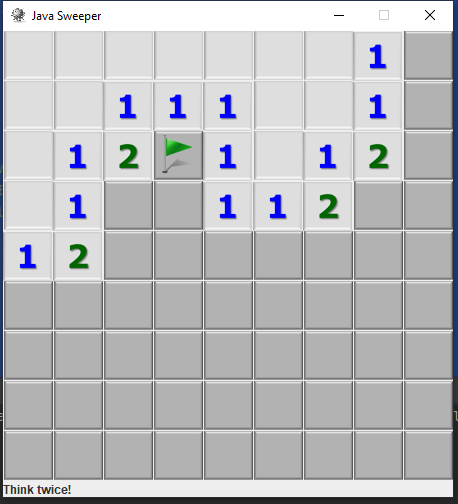


Рисунок 3.3 – Встановлення прапорця

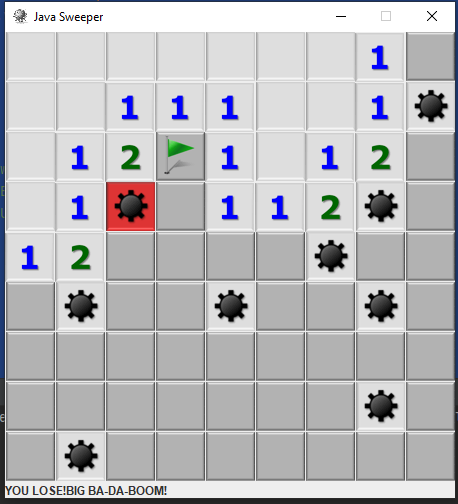


Рисунок 3.4 – Програш

3.2 Запуск гри через архів(майже як .exe)

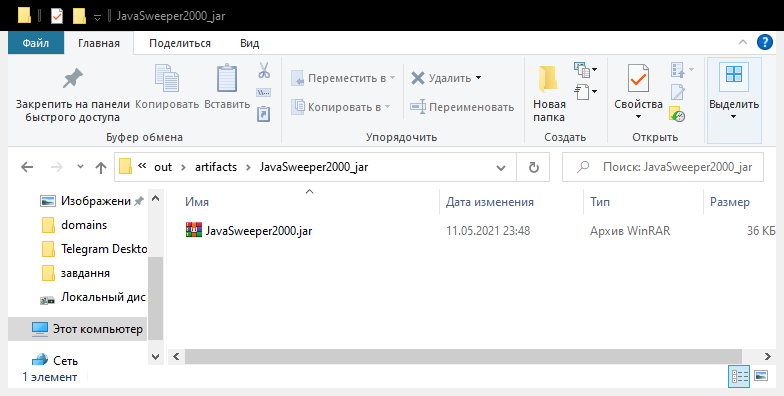


Рисунок 3.5 – Запуск гри через архів

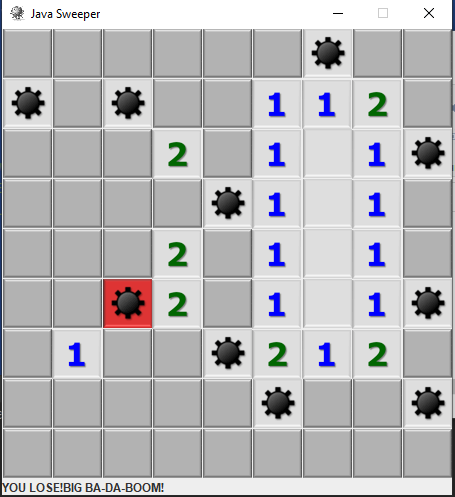


Рисунок 3.6 – Гра запущена через архів

# 

# висновки

В ході виконання даного курсового проекту були розроблена програма на мові java. А також вивчені можливості данної мови.

Систематизовані і закріплені практичні навички використання існуючих засобів обслуговування системних програмістів, а також теоретичні знання з основних розділів курсу "Об'єктно-орієнтоване програмування". Основну увагу приділено вивченню спо-собів проектування додатків, об'єктно-орієнтованого.

При виконанні курсового проекту вироблено знайомство з журналами та іншими інформаційними джерелами з об'єктно-орієнтованому і системного програмування з метою аналізу стану розв'язуваної задачі.

Отримано практичні навички роботи в середовищі JetBrains IntelliJ IDEA.

В ході розробки застосунку враховувалася складність поставлених задач та системні ресурси які можна виділити, щоб продукт мав низькі системні вимоги.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондаренко М.Ф., Бритик В.І., Свинар М.К. Конспект лекцій. Частина I "Алгоритмічні мови та програмування". "Компанія СМІТ" - Харків, 2004. -221 с.

2. Павловська Т.О. С / С ++ програмування на мові високого рівня. "Пітер" - Санкт Петербург, 2002. -460 с.

3. Річард С. Лінкер, Том Арчер. Програмування для Windows 98. Біблія розробника. "Діалектика" - Москва, 1999. -864 с .: іл. - Па-ра. тит. англ. Уч. сел.

4. Джесс Ліберті. С ++ за 21 день. "Вільямс" - Москва, 2000. -816 с .: іл. - Парал. тит. англ.

**ЗВІТ**

про унікальність курсової роботи на тему:

« Гра «Сапер» на мові java »

студентка спеціальності 121 Інженерія програмного забезпечення, 208 групи

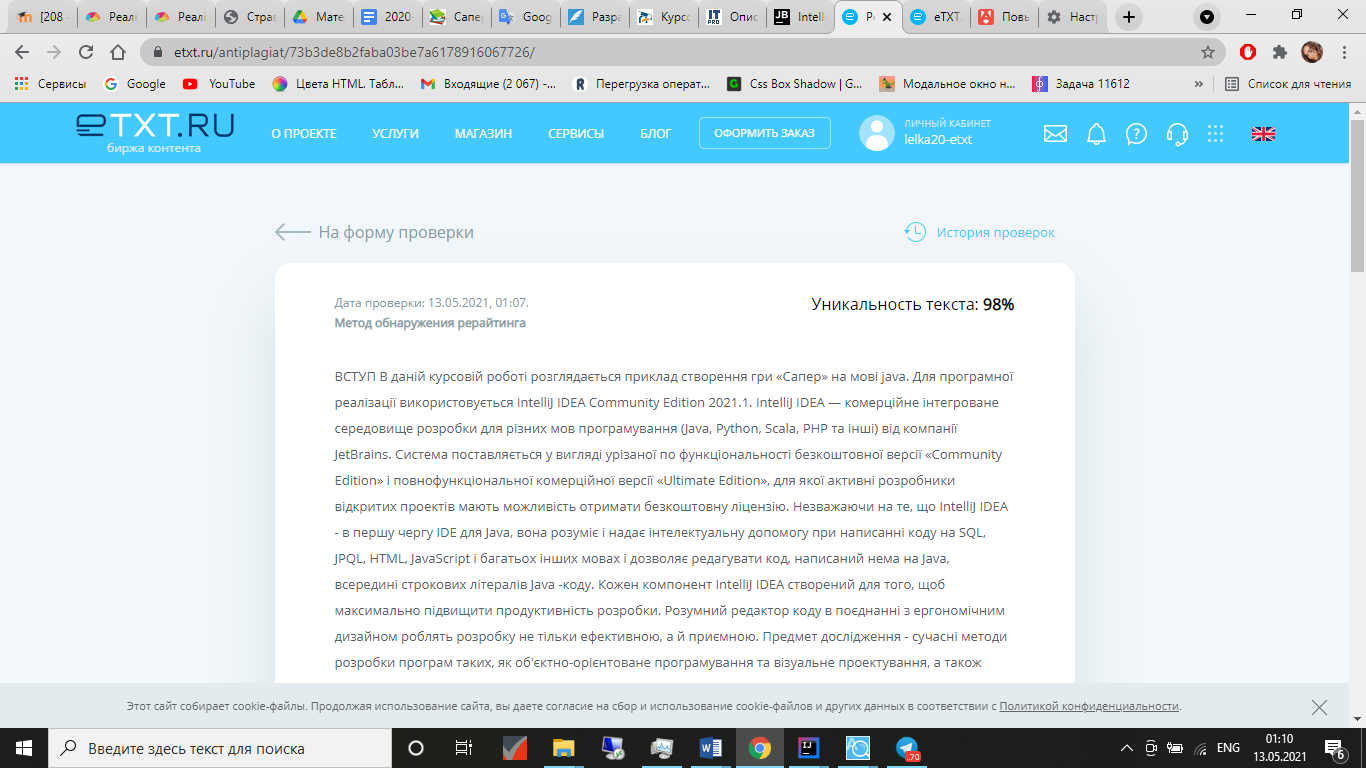
|  |
| --- |
| Сабіна В. О. |
| прізвище, ім’я, по-батькові |

|  |  |
| --- | --- |
| Шифр роботи: | 121 – КР.ПЗ.00 – 208.1910820 |

|  |  |
| --- | --- |
| Перевірку тексту здійснено сервісом: | Програма «AntiPlagiarism.NET» |
|  |  |

Результат перевірки тексту курсового проекту на унікальність складає 98%.

**Скріншот** результату  наведений на рисунку нижче.



Ст. викладач кафедри ІПЗ                                                  С. Ю. Боровльова

|  |  |  |  |
| --- | --- | --- | --- |
| Дата: | « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 2021 р. |  |