Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до лабораторної роботи № 3 з

курсу

*Прикладне програмувння*  на

тему:

**«** **Гра “Битва дроїдів”»**

Виконав: студент ОІ-24

**Стащишин Юрій**

Прийняв:

**Чорненький В. Я.**

Львів – 2024

***Лабораторна робота № 3***

**Тема:** Гра «Битва дроїдів»

**Мета роботи:** Написати свою реалізацію гри «Битва дроїдів»

*Мінімальний набір вимог до програми:*

1. Створіть базовий клас Droid, від якого будуть походити інші підкласи (види дроїдів), які будуть відрізнятися різними характеристиками. Мінімальний набір характеристик: name, health, damage.

2. Додайте можливість різних видів бою: 1 на 1, або команда на команду.

3. Класи потрібно грамотно розкласти по пакетах.

4. У програмі має бути консольне меню. Мінімальний набір команд: − створити дроїда (обраного виду); − показати список створених дроїдів; − запустити бій 1 на 1 (вибрати дроїдів, які будуть змагатися); − запустити бій команда на команду (сформувати команди суперників з дроїдів, яких ви створили у першому пункті); − записати проведений бій у файл; − відтворити проведений бій зі збереженого файлу; − вийти з програми.

**Код розробленої програми**

import menu.MainMenu;  
  
public class Main {  
 public static void main(String[] args) {  
 MainMenu menu = new MainMenu();  
 menu.showMenu();  
 }  
}

package menu;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
import battle.ThreeOnThreeBattle;  
import droids.\*;  
import battle.OneOnOneBattle;  
  
public class MainMenu {  
 private ArrayList<Droid> droids = new ArrayList<>();  
 private Scanner scanner = new Scanner(System.*in*);  
  
 public MainMenu() {  
 initializeDefaultDroids();  
 }  
  
 public void showMenu(){  
 while(true) {  
 System.*out*.println("---------------------------------------");  
 System.*out*.println("1. Створити дроїда");  
 System.*out*.println("2. Показати всіх дроїдів");  
 System.*out*.println("3. Почати битву 1 на 1");  
 System.*out*.println("4. Почати битву 3 на 3");  
 System.*out*.println("5. Записати битву 3 на 3 у файл");  
 System.*out*.println("6. Відтворити битву з файлу");  
 System.*out*.println("7. Вийти");  
  
 int choice = -1; // Ініціалізуємо choice з недопустимого значення  
 boolean validInput = false; // Прапорець для перевірки правильності введення  
  
 // Цикл для отримання правильного вводу  
 while (!validInput) {  
 try {  
 choice = scanner.nextInt();  
 scanner.nextLine(); // Очищуємо буфер  
 validInput = true; // Якщо ввід коректний, змінюємо прапорець  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть число від 1 до 7.");  
 scanner.nextLine(); // Очищуємо буфер вводу  
 }  
 }  
  
 switch (choice) {  
 case 1:  
 createDroid();  
 break;  
 case 2:  
 showDroids();  
 break;  
 case 3:  
 startBattle();  
 break;  
 case 4:  
 startThreeOnThreeBattle();  
 break;  
 case 5:  
 startThreeOnThreeBattleToFile();  
 break;  
 case 6:  
 replayBattleFromFile();  
 break;  
 case 7:  
 return;  
 }  
 }  
 }  
  
 private void initializeDefaultDroids() {  
 droids.add(new PoisonDroid("Viper", 500, 50, 0.5, 43));  
 droids.add(new BomberDroid("Bomber", 450, 40, 0.3));  
 droids.add(new WeaknessDroid("Weak", 630, 130, 0.25, 0.2));  
 droids.add(new Droid("Optimus", 600, 125, 0.34));  
 droids.add(new HealerDroid("Bumblebee", 550, 110, 0.9, 60));  
 droids.add(new Droid("Magnus", 400, 80, 0.23));  
 }  
  
 // створюємо дрона з перевіркою на правильність введення  
 private void createDroid() {  
 System.*out*.println("Якого ви хочете створити дроїда: ");  
 System.*out*.println("1 - Бойовий дроїд. \"Дроїд з простими характеристиками\"");  
 System.*out*.println("2 - Дроїд-лікар. \"Дроїд-лікар перед кожною своєю атакою лікує дроїда з найменшою кількістю здоров'я\"");  
 System.*out*.println("3 - Отруйний дроїд. \"Отруйний дроїд накладає ефект отравлення на кожного, кого він атакує. Яд б'є ворожого дроїда після кожної його атаки 3 рази\"");  
 System.*out*.println("4 - Дроїд-підривник. \"Дроїд-бомбер кидає гранати і завдає шкоди усім противникам\"");  
 System.*out*.println("5 - Дроїд-ослаблювач. \"Дроїд-ослаблювач після своєї атаки накладає на ворога ефект слабкості на 3 його наступні атаки (зменшує шкоду яку завдає ворог на певну кількість %)\"");  
  
 int type = -1;  
 boolean validInput = false;  
  
 // Цикл для отримання правильного вводу типу дроїда  
 while (!validInput) {  
 try {  
 type = scanner.nextInt();  
 scanner.nextLine(); // Очищення буфера  
 if (type < 1 || type > 5) {  
 System.*out*.println("Будь ласка, введіть число від 1 до 5.");  
 } else {  
 validInput = true; // Якщо ввід коректний, змінюємо прапорець  
 }  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть число від 1 до 5.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 System.*out*.println("Введіть назву дроїда: ");  
 String name = scanner.nextLine();  
  
 int health = 0;  
 int damage = 0;  
 double critChance = 0.0;  
  
 // Цикл для отримання правильного вводу здоров'я  
 while (true) {  
 try {  
 System.*out*.println("Введіть здоров'я дроїда: ");  
 health = scanner.nextInt();  
 if (health <= 0) {  
 System.*out*.println("Здоров'я має бути більше 0. Спробуйте ще раз.");  
 continue; // Продовжуємо запит  
 }  
 break; // Якщо ввід коректний, виходимо з циклу  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число для здоров'я.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Цикл для отримання правильного вводу шкоди  
 while (true) {  
 try {  
 System.*out*.println("Введіть шкоду дроїда: ");  
 damage = scanner.nextInt();  
 if (damage < 0) {  
 System.*out*.println("Шкода не може бути від'ємною. Спробуйте ще раз.");  
 continue; // Продовжуємо запит  
 }  
 break; // Якщо ввід коректний, виходимо з циклу  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число для шкоди.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Цикл для отримання правильного вводу шансу критичної шкоди  
 while (true) {  
 try {  
 System.*out*.println("Введіть шанс критичної шкоди (0.0 - 0.99): ");  
 critChance = scanner.nextDouble();  
 if (critChance < 0.0 || critChance >= 1.0) {  
 System.*out*.println("Шанс критичної шкоди має бути в діапазоні від 0.0 до 0.99. Спробуйте ще раз.");  
 continue; // Продовжуємо запит  
 }  
 break; // Якщо ввід коректний, виходимо з циклу  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число для шансу критичної шкоди.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 switch (type) {  
 case 1:  
 droids.add(new Droid(name, health, damage, critChance));  
 break;  
 case 2:  
 int healAmount = 0;  
 // Цикл для отримання правильного вводу healAmount  
 while (true) {  
 try {  
 System.*out*.println("Введіть число, на яке може полікувати дроїд: ");  
 healAmount = scanner.nextInt();  
 if (healAmount < 0) {  
 System.*out*.println("Кількість лікування не може бути від'ємною. Спробуйте ще раз.");  
 continue; // Продовжуємо запит  
 }  
 break; // Якщо ввід коректний, виходимо з циклу  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число для лікування.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
 droids.add(new HealerDroid(name, health, damage, critChance, healAmount));  
 break;  
 case 3:  
 int poisonDamage = 0;  
 // Цикл для отримання правильного вводу poisonDamage  
 while (true) {  
 try {  
 System.*out*.println("Введіть кількість отрути, яку може застосувати дроїд: ");  
 poisonDamage = scanner.nextInt();  
 if (poisonDamage < 0) {  
 System.*out*.println("Кількість отрути не може бути від'ємною. Спробуйте ще раз.");  
 continue; // Продовжуємо запит  
 }  
 break; // Якщо ввід коректний, виходимо з циклу  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число для отрути.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
 droids.add(new PoisonDroid(name, health, damage, critChance, poisonDamage));  
 break;  
 case 4:  
 droids.add(new BomberDroid(name, health, damage, critChance));  
 break;  
 case 5:  
 double weaknessChance = 0.0;  
 // Цикл для отримання правильного вводу weaknessChance  
 while (true) {  
 try {  
 System.*out*.println("Введіть ймовірність застосування слабкості (0.0 - 0.99): ");  
 weaknessChance = scanner.nextDouble();  
 if (weaknessChance < 0.0 || weaknessChance >= 1.0) {  
 System.*out*.println("Ймовірність має бути в діапазоні від 0.0 до 0.99. Спробуйте ще раз.");  
 continue; // Продовжуємо запит  
 }  
 break; // Якщо ввід коректний, виходимо з циклу  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число для ймовірності.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
 droids.add(new WeaknessDroid(name, health, damage, critChance, weaknessChance));  
 break;  
 default:  
 System.*out*.println("Неправильний тип дроїда. Спробуйте ще раз.");  
 break;  
 }  
  
 System.*out*.println("Дроїд " + name + " успішно створений!");  
 }  
  
  
 private void showDroids() {  
 int i = 1;  
 for (Droid droid : droids){  
 System.*out*.print(i + ") ");  
 droid.showDroidInfo();  
 i++;  
 }  
 }  
  
 private void startBattle() {  
 if (droids.size() < 2) {  
 System.*out*.println("Недостатньо дроїдів для битви.");  
 return;  
 }  
  
 int droid1Index = -1;  
 int droid2Index = -1;  
  
 // Цикл для вибору першого дроїда  
 while (true) {  
 System.*out*.println("Виберіть першого дроїда для битви: ");  
 showDroids();  
 try {  
 droid1Index = scanner.nextInt() - 1;  
 if (droid1Index < 0 || droid1Index >= droids.size()) {  
 System.*out*.println("Некоректний індекс дроїда. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо індекс некоректний, продовжуємо запит  
 }  
 break; // Виходимо з циклу, якщо ввід коректний  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Цикл для вибору другого дроїда  
 while (true) {  
 System.*out*.println("Виберіть другого дроїда для битви: ");  
 try {  
 droid2Index = scanner.nextInt() - 1; // Вводимо дроїда (зменшуємо на 1 для індексації)  
 if (droid2Index < 0 || droid2Index >= droids.size() || droid2Index == droid1Index) {  
 System.*out*.println("Некоректний індекс дроїда або дроїд вже обраний. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо індекс некоректний, продовжуємо запит  
 }  
 break; // Виходимо з циклу, якщо ввід коректний  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Отримання дроїдів  
 Droid droid1 = droids.get(droid1Index);  
 Droid droid2 = droids.get(droid2Index);  
  
 // Розпочинаємо бій  
 OneOnOneBattle battle = new OneOnOneBattle();  
 battle.fight(droid1, droid2);  
 }  
  
  
 private void startThreeOnThreeBattle() {  
 if (droids.size() < 6) {  
 System.*out*.println("Недостатньо дроїдів для битви 3 на 3.");  
 return;  
 }  
  
 showDroids();  
 List<Droid> team1 = new ArrayList<>();  
 List<Droid> team2 = new ArrayList<>();  
  
 // Цикл для вибору дроїдів першої команди  
 System.*out*.println("Виберіть дроїдів для першої команди: ");  
 while (team1.size() < 3) {  
 System.*out*.println("Оберіть дроїда для першої команди: ");  
 try {  
 int index = scanner.nextInt() - 1;  
 if (index < 0 || index >= droids.size() || team1.contains(droids.get(index))) {  
 System.*out*.println("Некоректний індекс дроїда або дроїд вже обраний. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо індекс некоректний, продовжуємо запит  
 }  
 team1.add(droids.get(index));  
 System.*out*.println(droids.get(index).getName() + " добавлено в першу команду");  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Цикл для вибору дроїдів другої команди  
 System.*out*.println("Виберіть дроїдів для другої команди: ");  
 while (team2.size() < 3) {  
 System.*out*.println("Оберіть дроїда для другої команди: ");  
 try {  
 int index = scanner.nextInt() - 1; // Вводимо дроїда (зменшуємо на 1 для індексації)  
 if (index < 0 || index >= droids.size() || team2.contains(droids.get(index)) || team1.contains(droids.get(index))) {  
 System.*out*.println("Некоректний індекс дроїда або дроїд вже обраний. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо індекс некоректний, продовжуємо запит  
 }  
 team2.add(droids.get(index));  
 System.*out*.println(droids.get(index).getName() + " добавлено в другу команду");  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Вибір тактики бою  
 int battleStrategy;  
 while (true) {  
 System.*out*.println("Виберіть тактику бою 3 на 3:");  
 System.*out*.println("1. Випадкова атака");  
 System.*out*.println("2. Атака по слабкому (менше здоров'я)");  
  
 try {  
 battleStrategy = scanner.nextInt();  
 if (battleStrategy < 1 || battleStrategy > 2) {  
 System.*out*.println("Некоректний вибір тактики. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо вибір некоректний, продовжуємо запит  
 }  
 break; // Виходимо з циклу, якщо ввід коректний  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Розпочинаємо бій  
 ThreeOnThreeBattle battle = new ThreeOnThreeBattle();  
 battle.fight(team1, team2, battleStrategy);  
 }  
  
 private void startThreeOnThreeBattleToFile() {  
 if (droids.size() < 6) {  
 System.*out*.println("Недостатньо дроїдів для битви 3 на 3.");  
 return;  
 }  
  
 showDroids();  
 List<Droid> team1 = new ArrayList<>();  
 List<Droid> team2 = new ArrayList<>();  
  
 // Цикл для вибору дроїдів першої команди  
 System.*out*.println("Виберіть дроїдів для першої команди: ");  
 while (team1.size() < 3) {  
 System.*out*.println("Оберіть дроїда для першої команди: ");  
 try {  
 int index = scanner.nextInt() - 1; // Вводимо дроїда (зменшуємо на 1 для індексації)  
 if (index < 0 || index >= droids.size() || team1.contains(droids.get(index))) {  
 System.*out*.println("Некоректний індекс дроїда або дроїд вже обраний. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо індекс некоректний, продовжуємо запит  
 }  
 team1.add(droids.get(index));  
 System.*out*.println(droids.get(index).getName() + " добавлено в першу команду");  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Цикл для вибору дроїдів другої команди  
 System.*out*.println("Виберіть дроїдів для другої команди: ");  
 while (team2.size() < 3) {  
 System.*out*.println("Оберіть дроїда для другої команди: ");  
 try {  
 int index = scanner.nextInt() - 1; // Вводимо дроїда (зменшуємо на 1 для індексації)  
 if (index < 0 || index >= droids.size() || team2.contains(droids.get(index)) || team1.contains(droids.get(index))) {  
 System.*out*.println("Некоректний індекс дроїда або дроїд вже обраний. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо індекс некоректний, продовжуємо запит  
 }  
 team2.add(droids.get(index));  
 System.*out*.println(droids.get(index).getName() + " добавлено в другу команду");  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Вибір тактики бою  
 int battleStrategy;  
 while (true) {  
 System.*out*.println("Виберіть тактику бою 3 на 3:");  
 System.*out*.println("1. Випадкова атака");  
 System.*out*.println("2. Атака по слабкому (менше здоров'я)");  
  
 try {  
 battleStrategy = scanner.nextInt();  
 if (battleStrategy < 1 || battleStrategy > 2) {  
 System.*out*.println("Некоректний вибір тактики. Будь ласка, спробуйте ще раз.");  
 continue; // Якщо вибір некоректний, продовжуємо запит  
 }  
 break; // Виходимо з циклу, якщо ввід коректний  
 } catch (java.util.InputMismatchException e) {  
 System.*out*.println("Будь ласка, введіть дійсне число.");  
 scanner.nextLine(); // Очищення буфера вводу  
 }  
 }  
  
 // Розпочинаємо бій  
 ThreeOnThreeBattle battle = new ThreeOnThreeBattle();  
 battle.fightToFile(team1, team2, battleStrategy);  
 }  
  
  
 private void replayBattleFromFile() {  
 System.*out*.println("Введіть ім'я файлу для відтворення битви: "); // C:\LP\семестр 3\прикладне програмування\lab3\battle\_results.txt  
 String filename = scanner.nextLine();  
 try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {  
 String line;  
 System.*out*.println("Відтворення битви з файлу: " + filename);  
 while ((line = reader.readLine()) != null) {  
 System.*out*.println(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

package droids;  
  
import java.util.Random;  
  
public class Droid {  
 private String name;  
 private int health; // поточне здоров'я  
 private int damage;  
 private int maxHealth;  
 private double critChance;  
 // ефекти на дроїді  
 private int poisonDuration = 0; // кількість ударів отрутою  
 private int takenPoisonDamage = 0; // кількість шкоди від отруєння  
 private int weaknessDuration = 0; // кількість ослаблених ударів  
 private double takenDamageReduction = 0; // відсоток на який зменшується сила удару  
  
 public Droid(String name, int health, int damage, double critChance){  
 this.name = name;  
 this.health = health;  
 this.damage = damage;  
 this.maxHealth = health;  
 this.critChance = critChance;  
 this.poisonDuration = 0;  
 this.takenPoisonDamage = 0;  
 this.weaknessDuration = 0;  
 this.takenDamageReduction = 0;  
 }  
  
 public boolean isCriticalHit() {  
 Random rand = new Random();  
 return rand.nextDouble() < critChance;  
 }  
  
 public void takeDamage(int damage){  
 health -= damage;  
 if (health <= 0)  
 health = 0;  
 }  
  
 public void takePoisonDamage(int poisonDamage){  
 health -= poisonDamage;  
 if (health <= 0)  
 health = 0;  
 setPoisonDuration(getPoisonDuration() - 1);  
 }  
  
 public void attack(Droid target) {  
 int damageDealt = damage;  
 if (getWeaknessDuration() > 0){  
 System.*out*.print("Атака слабша на " + getTakenDamageReduction()\*100 + "%. ");  
 damageDealt = (int) (damageDealt \* (1 - getTakenDamageReduction()));  
 setWeaknessDuration(getWeaknessDuration() - 1);  
 }  
  
 // Якщо критичний удар, додаємо подвоєну шкоду  
 if (isCriticalHit()) {  
 damageDealt \*= 2;  
 System.*out*.println(name + " атакує критично і наносить " + damageDealt + " шкоди дроїду " + target.getName());  
 } else {  
 System.*out*.println(name + " атакує і наносить " + damageDealt + " шкоди дроїду " + target.getName());  
 }  
  
 target.takeDamage(damageDealt);  
 // після кожного удару, якщо дрон отруєний, він буде отримувати шкоду  
 if (getPoisonDuration() > 0){  
 takePoisonDamage(getTakenPoisonDamage());  
 }  
  
 }  
  
 public void setHealth(int health) {  
 this.health = health; // максимальне здоров'я обмежене 100  
 if (this.health > maxHealth){  
 this.health = maxHealth;  
 }  
 }  
  
 public boolean isAlive(){  
 return health > 0;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getHealth() {  
 return health;  
 }  
  
 public int getDamage() {  
 return damage;  
 }  
  
 public int getMaxHealth() {  
 return maxHealth;  
 }  
  
 public int getPoisonDuration() {  
 return poisonDuration;  
 }  
  
 public int getTakenPoisonDamage() {  
 return takenPoisonDamage;  
 }  
  
 public int getWeaknessDuration() {  
 return weaknessDuration;  
 }  
  
 public double getTakenDamageReduction() {  
 return takenDamageReduction;  
 }  
  
 public void setPoisonDuration(int poisonDuration){  
 this.poisonDuration = poisonDuration;  
 }  
  
 public void setTakenPoisonDamage(int takenPoisonDamage) {  
 this.takenPoisonDamage = takenPoisonDamage;  
 }  
  
 public void setWeaknessDuration(int weaknessDuration) {  
 this.weaknessDuration = weaknessDuration;  
 }  
  
 public void setTakenDamageReduction(double takenDamageReduction) {  
 this.takenDamageReduction = takenDamageReduction;  
 }  
  
 public void showDroidInfo() {  
 System.*out*.println(getName() + " - Здоров'я: " + getMaxHealth() + " Шкода: " + getDamage() + " Шанс критичного удару: " + critChance \* 100 + "%.");  
 }  
}

package droids;  
import java.util.List;  
  
public class BomberDroid extends Droid {  
 public BomberDroid(String name, int health, int damage, double critChace){  
 super(name, health, damage, critChace);  
 }  
  
 public void attack(List<Droid> enemyTeam) {  
 int damageDealt = getDamage();  
  
 if (isCriticalHit()){  
 damageDealt \*= 2;  
 System.*out*.println(getName() + " атакує критично і наносить кожному по " + damageDealt + " шкоди!");  
 }else{  
 System.*out*.println(getName() + " атакує і наносить кожному по " + damageDealt + " шкоди!");  
 }  
  
 for (Droid enemy : enemyTeam){  
 enemy.takeDamage(damageDealt);  
 }  
 }  
  
 public void printDroidFeature(){  
 System.*out*.println("Дроїд-бомбер кидає гранати і завдає шкоди усім противникам");  
 }  
}

package droids;  
  
public class HealerDroid extends Droid{  
 private int healAmount;  
  
 public HealerDroid(String name, int health, int damage, double critChance, int healAmount){  
 super(name, health, damage, critChance);  
 this.healAmount = healAmount;  
 }  
  
 public void heal(Droid target){  
 if (target.isAlive()){  
 target.setHealth(target.getHealth() + healAmount);  
 System.*out*.println(this.getName() + " полікував " + target.getName() + " на " + healAmount + " здоров'я");  
 }  
 }  
  
 public int getHealAmount(){  
 return healAmount;  
 }  
  
 public void showDroidInfo() {  
 System.*out*.println(getName() + " - Здоров'я: " + getMaxHealth() + " Щкода: " + getDamage() + " Лікування: " + getHealAmount());  
 }  
  
 public void printDroidFeature(){  
 System.*out*.println("Дроїд-лікар перед кожною своєю атакою лікує дроїда з найменшою кількістю здоров'я");  
 }  
}

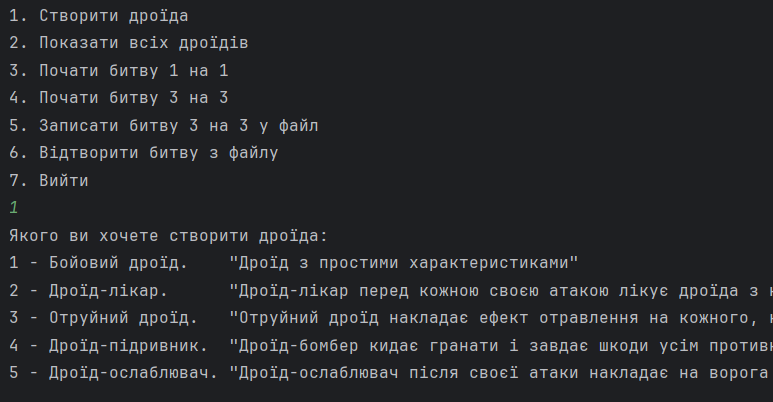
package droids;  
  
public class PoisonDroid extends Droid{  
 private int poisonDamage;  
  
 public PoisonDroid(String name, int health, int damage, double critChance, int poisonDamage) {  
 super(name, health, damage, critChance);  
 this.poisonDamage = poisonDamage;  
 }  
  
 public void attack(Droid target){  
 super.attack(target);  
 System.*out*.println(getName() + " отруїв дроїда " + target.getName() + " на " + poisonDamage + " шкоди на 3 наступні ходи дроїда");  
 target.setPoisonDuration(3);  
 target.setTakenPoisonDamage(poisonDamage);  
 }  
  
 public void printDroidFeature(){  
 System.*out*.println("Отруйний дроїд накладає ефект отравлення на кожного, кого він атакує. Яд б'є ворожого дроїда після кожної його атаки 3 рази");  
 }  
  
 public int getPoisonDamage() {  
 return poisonDamage;  
 }  
}

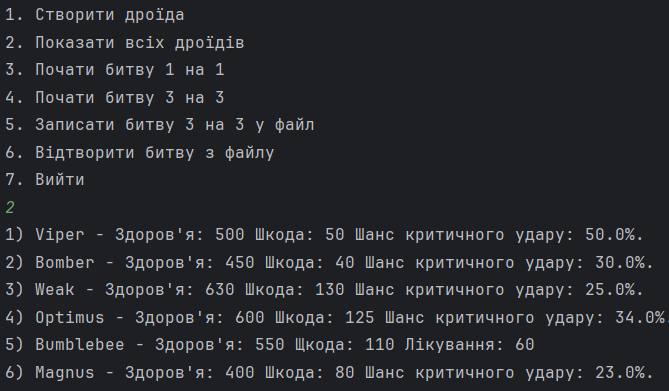
package droids;  
  
public class WeaknessDroid extends Droid {  
 private double damageReduction;  
  
 public WeaknessDroid(String name, int health, int damage, double critChance, double damageReduction){  
 super(name, health, damage, critChance);  
 this.damageReduction = damageReduction;  
 }  
  
 public void attack(Droid target){  
 super.attack(target);  
 System.*out*.println(getName() + " наклав слабкість на " + target.getName() + " на 3 наступні ходи");  
 target.setWeaknessDuration(3);  
 target.setTakenDamageReduction(damageReduction);  
 }  
  
 public void printDroidFeature(){  
 System.*out*.println("Дроїд-ослаблювач після своєї атаки накладає на ворога ефект слабкості на 3 його наступні атаки (зменшує шкоду яку завдає ворог на певну кількість %)");  
 }  
}

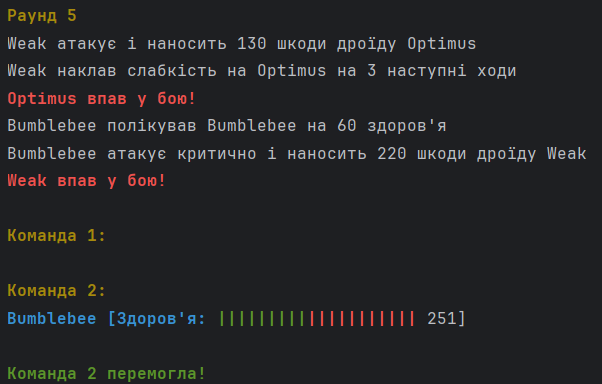
package battle;  
import droids.Droid;  
  
public class OneOnOneBattle {  
 public static final String *RESET* = "\u001B[0m";  
 public static final String *RED* = "\u001B[31m";  
 public static final String *GREEN* = "\u001B[32m";  
 public static final String *YELLOW* = "\u001B[33m";  
 public static final String *CYAN* = "\u001B[36m";  
 public static final String *BLUE* = "\u001B[34m";  
 public static final String *BOLD* = "\u001B[1m";  
  
 public void fight(Droid droid1, Droid droid2){  
 System.*out*.println("------------------------------------------");  
 System.*out*.println(*BOLD* + *CYAN* + droid1.getName() + " vs " + droid2.getName() + *RESET*);  
 System.*out*.println();  
  
 printHealthStatus(droid1, droid2);  
  
 int round = 1;  
 while (droid1.isAlive() && droid2.isAlive()){  
 System.*out*.println(*BOLD* + *YELLOW* + "Раунд " + round + *RESET*);  
  
 droid1.attack(droid2);  
 printHealthStatus(droid1, droid2);  
  
 if (!droid2.isAlive()){  
 break;  
 }  
  
 droid2.attack(droid1);  
 printHealthStatus(droid1, droid2);  
  
 System.*out*.println(*BOLD* + "------------------------------------------" + *RESET*);  
 round++;  
 }  
  
 // повертаємо кількість здоров'я дроїдам після бою  
 droid1.setHealth(droid1.getMaxHealth());  
 droid2.setHealth(droid2.getMaxHealth());  
  
 if (droid1.isAlive()) {  
 System.*out*.println(*BOLD* + *GREEN* + droid1.getName() + " переміг!" + *RESET*);  
 } else if (droid2.isAlive()) {  
 System.*out*.println(*BOLD* + *GREEN* + droid2.getName() + " переміг!" + *RESET*);  
 } else {  
 System.*out*.println(*BOLD* + *CYAN* + "Обидва дроїди загинули, це нічия!" + *RESET*);  
 }  
 }  
  
 private void printHealthStatus(Droid droid1, Droid droid2){  
 System.*out*.println(*BLUE* + droid1.getName() + " [Здоров'я: " + healthBar(droid1.getHealth(), droid1.getMaxHealth()) + " " + droid1.getHealth() + "]" + *RESET*);  
 System.*out*.println(*BLUE* + droid2.getName() + " [Здоров'я: " + healthBar(droid2.getHealth(), droid2.getMaxHealth()) + " " + droid2.getHealth() + "]" + *RESET*);  
 }  
  
 private String healthBar(int health, int maxHealth){  
 int totalBars = 20;  
 int filledBars = (int) ((health / (double) maxHealth) \* totalBars);  
 StringBuilder bar = new StringBuilder();  
  
 for (int i = 0; i < filledBars; i++){  
 bar.append(*GREEN* + "|" + *RESET*);  
 }  
 for(int i = filledBars; i < totalBars; i++){  
 bar.append(*RED* + "|" + *RESET*);  
 }  
  
 return bar.toString();  
 }  
}

package battle;  
  
import droids.BomberDroid;  
import droids.Droid;  
import droids.HealerDroid;  
  
import java.util.List;  
import java.util.Random;  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintStream;  
  
public class ThreeOnThreeBattle {  
 public static final String *RESET* = "\u001B[0m";  
 public static final String *RED* = "\u001B[31m";  
 public static final String *GREEN* = "\u001B[32m";  
 public static final String *YELLOW* = "\u001B[33m";  
 public static final String *CYAN* = "\u001B[36m";  
 public static final String *BLUE* = "\u001B[34m";  
 public static final String *BOLD* = "\u001B[1m";  
  
 private Random rand = new Random();  
  
 public void fight(List<Droid> team1, List<Droid> team2, int strategy){  
 System.*out*.println("------------------------------------------");  
 System.*out*.println(*BOLD* + *CYAN* + "Бій між командами " + team1.get(0).getName() + " та " + team2.get(0).getName() + *RESET*);  
 System.*out*.println();  
  
 int round = 1;  
 System.*out*.println(*BOLD* + *YELLOW* + "Команда 1:"+ *RESET*);  
 for (Droid droid : team1) {  
 printHealthStatus(droid);  
 }  
 System.*out*.println();  
  
 System.*out*.println(*BOLD* + *YELLOW* + "Команда 2:"+ *RESET*);  
 for (Droid droid : team2) {  
 printHealthStatus(droid);  
 }  
 System.*out*.println();  
  
 while(!team1.isEmpty() && ! team2.isEmpty()){  
 System.*out*.println(*BOLD* + *YELLOW* + "Раунд " + round + *RESET*);  
  
 // цикл який працює поки в обох командах є хоч один дроїд  
 for (int i = 0; i < Math.*max*(team1.size(), team2.size()); i++){  
 if (!team1.isEmpty() && !team2.isEmpty()){  
 Droid attacker;  
 Droid defender;  
  
 if (i <= team1.size()){  
 // додаткові перевірки через out of baunce  
 if (i == team1.size() && i != 0){  
 attacker = team1.get(i-1);  
 } else {  
 attacker = team1.get(i);  
 }  
 defender = chooseDefender(team2, strategy);  
  
 // якщо лікар то шукає тімейта з найменшою кількістю здоров'я  
 if (attacker instanceof HealerDroid){  
 Droid minHealthDroid = team1.get(0);  
 for (Droid droid : team1){  
 if (droid.getHealth() < minHealthDroid.getHealth()){  
 minHealthDroid = droid;  
 }  
 }  
 ((HealerDroid) attacker).heal(minHealthDroid);  
 }  
  
 if (attacker instanceof BomberDroid) {  
 // Якщо дроїд є підривником, викликаємо bombAttack  
 ((BomberDroid) attacker).attack(team2);  
 } else {  
 // Звичайна атака для інших типів дроїдів  
 attacker.attack(defender);  
 }  
  
 // цикл для перевірки чи не помер дроїд під час бою  
 for (Droid droid : team2) {  
 if (!droid.isAlive()) {  
 System.*out*.println(*BOLD* + *RED* + droid.getName() + " впав у бою!" + *RESET*);  
 droid.setHealth(droid.getMaxHealth());  
 team2.remove(droid);  
 break;  
 }  
 }  
 }  
  
 if (team1.isEmpty() || team2.isEmpty()) {  
 break;  
 }  
  
 if (i <= team2.size()){  
 if (i == team2.size() && i != 0){  
 attacker = team2.get(i-1);  
 } else {  
 attacker = team2.get(i);  
 }  
 defender = chooseDefender(team1, strategy);  
  
 // якщо лікар то шукає тімейта з найменшою кількістю здоров'я  
 if (attacker instanceof HealerDroid){  
 Droid minHealthDroid = team2.get(0);  
 for (Droid droid : team2){  
 if (droid.getHealth() < minHealthDroid.getHealth()){  
 minHealthDroid = droid;  
 }  
 }  
 ((HealerDroid) attacker).heal(minHealthDroid);  
 }  
 if (attacker instanceof BomberDroid) {  
 // Якщо дроїд є підривником, викликаємо bombAttack  
 ((BomberDroid) attacker).attack(team1);  
 } else {  
 // Звичайна атака для інших типів дроїдів  
 attacker.attack(defender);  
 }  
 // цикл для перевірки чи не помер дроїд під час бою  
 for (Droid droid : team1) {  
 if (!droid.isAlive()) {  
 System.*out*.println(*BOLD* + *RED* + droid.getName() + " впав у бою!" + *RESET*);  
 droid.setHealth(droid.getMaxHealth());  
 team1.remove(droid);  
 break;  
 }  
 }  
 }  
 }  
 }  
  
 System.*out*.println();  
 System.*out*.println(*BOLD* + *YELLOW* + "Команда 1:"+ *RESET*);  
 for (Droid droid : team1) {  
 printHealthStatus(droid);  
 }  
 System.*out*.println();  
  
 System.*out*.println(*BOLD* + *YELLOW* + "Команда 2:"+ *RESET*);  
 for (Droid droid : team2) {  
 printHealthStatus(droid);  
 }  
 System.*out*.println();  
  
 if (team1.isEmpty() || team2.isEmpty()) {  
 break;  
 }  
  
 System.*out*.println(*BOLD* + "------------------------------------------" + *RESET*);  
 round++;  
 }  
  
 // повертаємо здоров'я дроїдам  
 for (Droid droid : team1){  
 droid.setHealth(droid.getMaxHealth());  
 }  
  
 for (Droid droid : team2){  
 droid.setHealth(droid.getMaxHealth());  
 }  
  
 if (team1.isEmpty()) {  
 System.*out*.println(*BOLD* + *GREEN* + "Команда 2 перемогла!" + *RESET*);  
 } else {  
 System.*out*.println(*BOLD* + *GREEN* + "Команда 1 перемогла!" + *RESET*);  
 }  
 }  
  
 public void fightToFile(List<Droid> team1, List<Droid> team2, int strategy) {  
 // Вказуємо ім'я файлу для запису  
 File outputFile = new File("battle\_results.txt");  
 PrintStream originalOut = System.*out*; // Зберігаємо стандартний вихід  
 try {  
 // Створюємо PrintStream для запису у файл  
 PrintStream fileOut = new PrintStream(outputFile);  
 System.*setOut*(fileOut); // Перенаправляємо стандартний вихід у файл  
  
 System.*out*.println("------------------------------------------");  
 System.*out*.println("Бій між командами " + team1.get(0).getName() + " та " + team2.get(0).getName());  
 System.*out*.println();  
  
 int round = 1;  
 System.*out*.println("Команда 1:");  
 for (Droid droid : team1) {  
 System.*out*.println(droid.getName() + " [Здоров'я: " + droid.getHealth() + "]");  
 }  
 System.*out*.println();  
  
 System.*out*.println("Команда 2:");  
 for (Droid droid : team2) {  
 System.*out*.println(droid.getName() + " [Здоров'я: " + droid.getHealth() + "]");  
 }  
 System.*out*.println();  
  
 while (!team1.isEmpty() && !team2.isEmpty()) {  
 System.*out*.println("Раунд " + round);  
  
 for (int i = 0; i < Math.*max*(team1.size(), team2.size()); i++) {  
 if (!team1.isEmpty() && !team2.isEmpty()) {  
 Droid attacker;  
 Droid defender;  
  
 // Атаки команди 1  
 if (i < team1.size()) {  
 attacker = team1.get(i);  
 defender = chooseDefender(team2, strategy);  
  
 // якщо лікар то шукає тімейта з найменшою кількістю здоров'я  
 if (attacker instanceof HealerDroid){  
 Droid minHealthDroid = team1.get(0);  
 for (Droid droid : team1){  
 if (droid.getHealth() < minHealthDroid.getHealth()){  
 minHealthDroid = droid;  
 }  
 }  
 ((HealerDroid) attacker).heal(minHealthDroid);  
 }  
  
 if (attacker instanceof BomberDroid) {  
 ((BomberDroid) attacker).attack(team2);  
 } else {  
 attacker.attack(defender);  
 }  
  
 for (Droid droid : team2) {  
 if (!droid.isAlive()) {  
 System.*out*.println(droid.getName() + " впав у бою!");  
 droid.setHealth(droid.getMaxHealth());  
 team2.remove(droid);  
 break;  
 }  
 }  
 }  
  
 // Атаки команди 2  
 if (i < team2.size()) {  
 attacker = team2.get(i);  
 defender = chooseDefender(team1, strategy);  
  
 // якщо лікар то шукає тімейта з найменшою кількістю здоров'я  
 if (attacker instanceof HealerDroid){  
 Droid minHealthDroid = team2.get(0);  
 for (Droid droid : team2){  
 if (droid.getHealth() < minHealthDroid.getHealth()){  
 minHealthDroid = droid;  
 }  
 }  
 ((HealerDroid) attacker).heal(minHealthDroid);  
 }  
  
 if (attacker instanceof BomberDroid) {  
 ((BomberDroid) attacker).attack(team1);  
 } else {  
 attacker.attack(defender);  
 }  
  
 for (Droid droid : team1) {  
 if (!droid.isAlive()) {  
 System.*out*.println(droid.getName() + " впав у бою!");  
 droid.setHealth(droid.getMaxHealth());  
 team1.remove(droid);  
 break;  
 }  
 }  
 }  
 }  
 }  
  
 System.*out*.println();  
 System.*out*.println("Команда 1:");  
 for (Droid droid : team1) {  
 System.*out*.println(droid.getName() + " [Здоров'я: " + droid.getHealth() + "]");  
 }  
 System.*out*.println();  
  
 System.*out*.println("Команда 2:");  
 for (Droid droid : team2) {  
 System.*out*.println(droid.getName() + " [Здоров'я: " + droid.getHealth() + "]");  
 }  
 System.*out*.println();  
  
 if (team1.isEmpty() || team2.isEmpty()) {  
 break;  
 }  
  
 System.*out*.println("------------------------------------------");  
 round++;  
 }  
  
 // повертаємо здоров'я дроїдам  
 for (Droid droid : team1){  
 droid.setHealth(droid.getMaxHealth());  
 }  
  
 for (Droid droid : team2){  
 droid.setHealth(droid.getMaxHealth());  
 }  
  
 if (team1.isEmpty()) {  
 System.*out*.println("Команда 2 перемогла!");  
 } else {  
 System.*out*.println("Команда 1 перемогла!");  
 }  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace(); // Виводимо стек трейс у консоль, якщо файл не знайдено  
 } finally {  
 // Повертаємо стандартний вихід у консоль  
 System.*setOut*(originalOut);  
 System.*out*.println("Результати бою записано у файл: " + outputFile.getAbsolutePath());  
 }  
 }  
  
  
  
 private Droid chooseDefender(List<Droid> team, int strategy){  
 if (strategy == 1){  
 return team.get(rand.nextInt(team.size()));  
 } else if (strategy == 2){  
 Droid minHealthDroid = team.get(0);  
 for (Droid droid : team){  
 if (droid.getHealth() < minHealthDroid.getHealth()){  
 minHealthDroid = droid;  
 }  
 }  
 return minHealthDroid;  
 }  
 return null;  
 }  
  
 private void printHealthStatus(Droid droid) {  
 System.*out*.println(*BOLD* + *BLUE* + droid.getName() + " [Здоров'я: " + healthBar(droid.getHealth(), droid.getMaxHealth()) + " " + droid.getHealth() + "]" + *RESET*);  
 }  
  
 private String healthBar(int health, int maxHealth){  
 int totalBars = 20;  
 int filledBars = (int) ((health / (double) maxHealth) \* totalBars);  
 StringBuilder bar = new StringBuilder();  
  
 for (int i = 0; i < filledBars; i++){  
 bar.append(*BOLD* + *GREEN* + "|" + *RESET*);  
 }  
 for(int i = filledBars; i < totalBars; i++){  
 bar.append(*BOLD* + *RED* + "|" + *RESET*);  
 }  
  
 return bar.toString();  
 }  
}

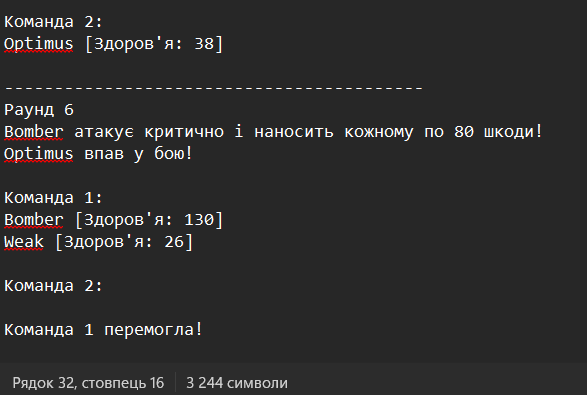
**Результат**

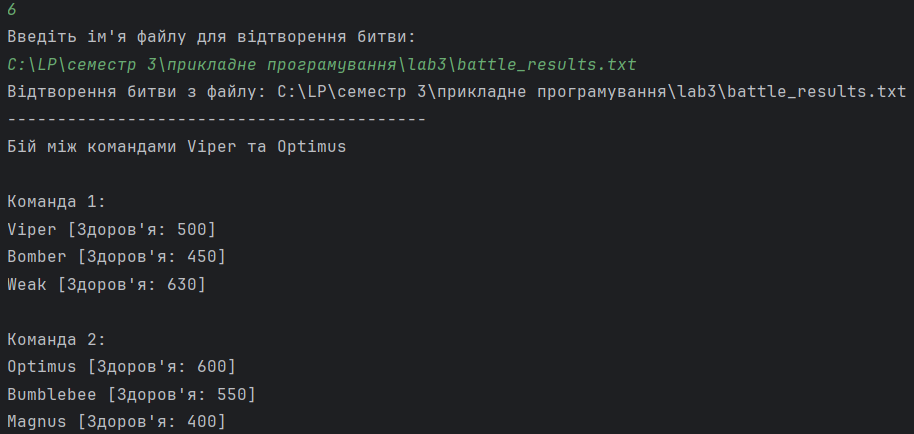
****

****

****

****

****

****

**Висновок**

Завдяки цій лабораторній роботі я навчився розробляти програми з великою кількістю класів, вивчив наслідування класів, дізнався як записувати текст з програми до файлу і читати з файлу. Було розроблено програму у якій все це реалізовано.

Посилання на Git: https://github.com/Float36/labs