­­­­­­­­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 дисципліни

*Прикладне програмування*

на тему:

**“ Битва дроїдів ”**

Виконав: студент ОІ-24

**Зубач Руслан**

Прийняв: асистент каф. АСУ

Чорненький В.Я.

**Львів – 2024**

*Лабораторна робота №* *3*

**Завдання лабораторної роботи**

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 програми**

**Droid**

public class Droid {  
 private String name;  
 private int health;  
 private int damage;  
 private int power;

public void updateEffects(){  
 if (activeEffects.isEmpty()){  
 return;  
 }  
 List<String> effectsToRemove = new ArrayList<>();  
 for(Map.Entry<String, Integer> entry : effectDurations.entrySet()){  
 int remainingDuration = entry.getValue() - 1;  
  
 if(entry.getKey().equals("Отруєнний")){  
 takePoisonDamage(15);  
 }  
 if(remainingDuration <= 0){  
 System.*out*.println(name + " втратив ефект " + BattleVisualizer.*YELLOW* + entry.getKey() + BattleVisualizer.*RESET*);  
 effectsToRemove.add(entry.getKey());  
 }  
 else{  
 entry.setValue(remainingDuration);  
 }  
 }  
 activeEffects.removeAll(effectsToRemove);  
 for(String effect : effectsToRemove){  
 effectDurations.remove(effect);  
 }  
}  
  
public void attack(ArrayList<Droid> enemies) {  
 if(hasEffect("Знемовленний")){  
 System.*out*.println(name + " не може атакувати, оскільки він" + BattleVisualizer.*ORANGE* + " знемовленний" + BattleVisualizer.*RESET*);  
 return;  
 }  
 Random random = new Random();  
 int enemy = random.nextInt(enemies.size());  
 System.*out*.println(getName() + BattleVisualizer.*RED* + " атакує " + BattleVisualizer.*RESET* + enemies.get(enemy).getName());  
 if(this.hasEffect("Підсилений")){  
 enemies.get(enemy).takeDamage((int)(this.getDamage() \* 1.2), this,false);  
 }  
 else{  
 enemies.get(enemy).takeDamage(this.getDamage(), this,false);  
 }  
 if (enemies.get(enemy).getHealth() <= 0) {  
 System.*out*.println(enemies.get(enemy).getName() + BattleVisualizer.*RED* + " було знищено!" + BattleVisualizer.*RESET*);  
 enemies.remove(enemy);  
 }  
}

**HealerDroid**

public class HealerDroid extends Droid {  
 private int healCost;  
 private int purifyCost;  
 private int healAmount;

@Override  
public void takeTurn(ArrayList<Droid> allies, ArrayList<Droid> enemies) {  
 if (!hasEffect("Знемовленний")) {  
 int maxNegativeEffectsIndex = findMaxEffects(allies);  
 int weakestDroid = findLowestHealth(allies);  
 if (purifyCost <= getPower() && maxNegativeEffectsIndex != -1) {  
 cast(purifyCost);  
 purify(allies.get(maxNegativeEffectsIndex));  
 } else if (healCost <= getPower() && allies.get(weakestDroid).getHealth() <= 50) {  
 cast(healCost);  
 heal(allies.get(weakestDroid), healAmount);  
 } else {  
 attack(enemies);  
 }  
 }  
}

public void purify(Droid otherDroid){  
 System.*out*.println(getName() + BattleVisualizer.*GREEN* + " знімає негативні ефекти" + BattleVisualizer.*RESET* + " з " + otherDroid.getName());  
 otherDroid.removeNegativeEffects();  
}  
  
public void heal(Droid otherDroid,int healAmount) {  
 System.*out*.println(getName() + BattleVisualizer.*GREEN* + " лікує "+ BattleVisualizer.*RESET* + otherDroid.getName());  
 otherDroid.gainHealth(healAmount);  
}

**StunDroid**

public class StunDroid extends Droid {  
 private int stunCost;  
 private int stunDuration;  
 private double reflectedPercentage;

@Override  
public void takeTurn(ArrayList<Droid> allies, ArrayList<Droid> enemies) {  
 if(getHealth() > 0 && !hasEffect("Знемовленний")){  
 Random rand = new Random();  
 if(stunCost <= getPower() && rand.nextBoolean()){  
 cast(getStunCost());  
 stun(enemies);  
 }  
 else{  
 attack(enemies);  
 }  
 }  
}

public void stun(ArrayList<Droid> enemies){  
 Random random = new Random();  
 int enemy = random.nextInt(enemies.size());  
 System.*out*.println(getName() + BattleVisualizer.*ORANGE* + " знемовлює " + BattleVisualizer.*RESET* + enemies.get(enemy).getName() + " на " + BattleVisualizer.*YELLOW* + getStunDuration() + BattleVisualizer.*RESET* + " ходів");  
 enemies.get(enemy).addEffect("Знемовленний", getStunDuration());  
}  
  
@Override  
*// Повернення шкоди*public void takeDamage(int damage, Droid attacker, boolean isReflected) {  
 super.takeDamage(damage, attacker, isReflected);  
  
 if(!isReflected){  
 int reflectedDamage = (int) (damage \* getReflectedPercentage());  
 if (this.getHealth() - reflectedDamage <= 0){  
 reflectedDamage = getHealth() - 1;  
 System.*out*.println(getName() + " повертає " + BattleVisualizer.*RED* + reflectedDamage + BattleVisualizer.*RESET* + " шкоди " + attacker.getName() + " (на межі смерті)");  
 attacker.takeDamage(reflectedDamage, this, true);  
 }  
 else {  
 System.*out*.println(getName() + " повертає " + BattleVisualizer.*RED* + reflectedDamage + BattleVisualizer.*RESET* + " шкоди " + attacker.getName());  
 attacker.takeDamage(reflectedDamage, this, true);  
 }  
  
 }  
}

**AssassinDroid**

public class AssassinDroid extends Droid {  
 private int critChance;  
 private int dodgeChance;

@Override  
*// Шанс завдати критичної шкоди*public void attack(ArrayList<Droid> enemies) {  
 if (hasEffect("Знемовленний")) {  
 System.*out*.println(getName() + " не може атакувати, оскільки він" + BattleVisualizer.*ORANGE* + " знемовленний" + BattleVisualizer.*RESET*);  
 return;  
 }  
 Random random = new Random();  
 int enemy = random.nextInt(enemies.size());  
  
 if (random.nextInt(0, 100) <= getCritChance()) {  
 if(this.hasEffect("Підсилений")){  
 int critDamage = (int)(getDamage() \* 1.2 \* 2);  
 System.*out*.println(getName() + BattleVisualizer.*MAROON* + " завдає критичної шкоди " + BattleVisualizer.*RESET* + enemies.get(enemy).getName());  
 enemies.get(enemy).takeDamage(critDamage, this,false);  
 }  
 else {  
 int critDamage = getDamage() \* 2;  
 System.*out*.println(getName() + BattleVisualizer.*MAROON* + " завдає критичної шкоди " + BattleVisualizer.*RESET* + enemies.get(enemy).getName());  
 enemies.get(enemy).takeDamage(critDamage, this,false);  
 }  
 if (enemies.get(enemy).getHealth() <= 0) {  
 System.*out*.println(enemies.get(enemy).getName() + BattleVisualizer.*RED* + " було знищено!" + BattleVisualizer.*RESET*);  
 enemies.remove(enemy);  
 }  
 } else {  
 super.attack(enemies);  
 }  
}  
  
  
  
@Override  
*// Шанс ухилитись від атаки*public void takeDamage(int damage, Droid attacker, boolean isReflected) {  
 Random random = new Random();  
 if(!hasEffect("Знемовленний") && !hasEffect("Отруєнний") && random.nextInt(0, 100) <= getDodgeChance()){  
 System.*out*.println(getName() + BattleVisualizer.*DARK\_BLUE* + " ухилився " + BattleVisualizer.*RESET* + "від атаки");  
 }  
 else{  
 super.takeDamage(damage, attacker, isReflected);  
 }  
}

**AlchemistDroid**

public class AlchemistDroid extends Droid {  
 private int poisonCost;  
 private int poisonDuration;  
 private int boostCost;  
 private int boostDuration;

public void takeTurn(ArrayList<Droid> allies, ArrayList<Droid> enemies) {  
 Random random = new Random();  
 int droid = random.nextInt(enemies.size());  
 if(!hasEffect("Знемовленний")) {  
 if(getPower() >= getPoisonCost() && random.nextBoolean() && !enemies.get(droid).hasEffect("Отруєнний")) {  
 cast(getPoisonCost());  
 poison(enemies.get(droid));  
 }  
 else if(!random.nextBoolean()){  
 cast(getBoostCost());  
 boost(allies.get(droid));  
 }  
 else {  
 attack(enemies);  
 }  
 }  
}

public void poison(Droid enemy) {  
 System.*out*.println(getName() + BattleVisualizer.*PURPLE* + " отруює " + BattleVisualizer.*RESET* + enemy.getName() + " на " + BattleVisualizer.*YELLOW* + getPoisonDuration() + BattleVisualizer.*RESET* + " ходів");  
 enemy.addEffect("Отруєнний", getPoisonDuration());  
}  
  
public void boost(Droid ally) {  
 System.*out*.println(getName() + BattleVisualizer.*PINK* + " підсилює " + BattleVisualizer.*RESET* + ally.getName() + " на " + BattleVisualizer.*YELLOW* + getBoostDuration() + BattleVisualizer.*RESET* + " ходів");  
 ally.addEffect("Підсилений", getBoostDuration());  
}

**Battle**

public static void battle(ArrayList<Droid> teamBlue, ArrayList<Droid> teamRed) {  
 Random random = new Random();  
  
 boolean teamTurn = random.nextBoolean();  
 System.*out*.println(BattleVisualizer.*BOLD* + BattleVisualizer.*YELLOW* + "=== Початок битви ===" + BattleVisualizer.*RESET*);  
  
 while (!teamBlue.isEmpty() && !teamRed.isEmpty()) {  
  
 if (teamTurn) {  
 System.*out*.println(BattleVisualizer.*BLUE* + BattleVisualizer.*BOLD* + "Хід синьої команди" + BattleVisualizer.*RESET*);  
 Collections.*shuffle*(teamBlue);  
 for (Droid d : teamBlue) {  
 if (teamRed.isEmpty()) break;  
 d.takeTurn(teamBlue, teamRed);  
 d.updateEffects();  
 *delay*();  
 }  
 }  
 else {  
 System.*out*.println(BattleVisualizer.*RED* + BattleVisualizer.*BOLD* + "Хід червоних" + BattleVisualizer.*RESET*);  
 Collections.*shuffle*(teamRed);  
 for (Droid d : teamRed) {  
 if (teamBlue.isEmpty()) break;  
 d.takeTurn(teamRed, teamBlue);  
 d.updateEffects();  
 *delay*();  
 }  
 }  
 teamTurn = !teamTurn;  
  
 if(teamBlue.isEmpty()){  
 System.*out*.println(BattleVisualizer.*YELLOW* + BattleVisualizer.*BOLD* + "Перемогли червоні" + BattleVisualizer.*RESET*);  
 }  
 else if(teamRed.isEmpty()) {  
 System.*out*.println(BattleVisualizer.*YELLOW* + BattleVisualizer.*BOLD* + "Перемогли сині" + BattleVisualizer.*RESET*);  
 }  
 }  
}

**Menu**

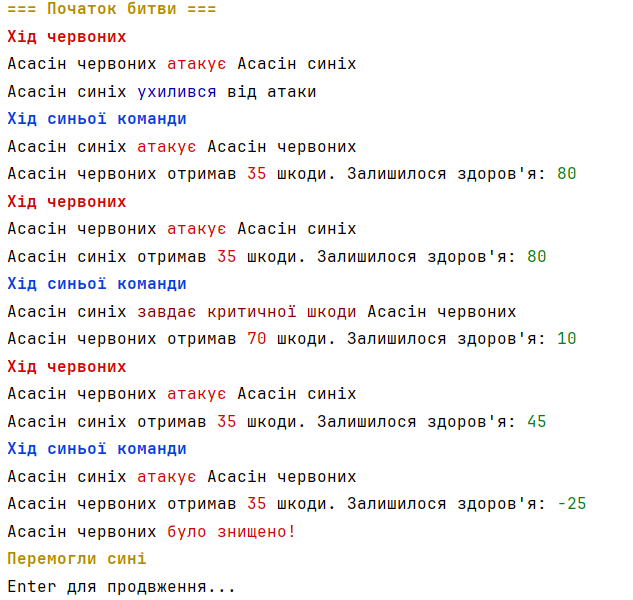
public void showMenu() {  
 while (true) {  
 System.*out*.println("1. Створити дроїда");  
 System.*out*.println("2. Видалити дроїда");  
 System.*out*.println("3. Показати список дроїдів в командах");  
 System.*out*.println("4. Запустити бій команда на команду");  
 System.*out*.println("5. Записати проведений бій у файл");  
 System.*out*.println("6. Відтворити проведений бій зі збереженого файлу");  
 System.*out*.println("7. Вийти з програми");  
 System.*out*.print("Оберіть дію: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine();  
  
 switch (choice) {  
 case 1:  
 createDroid();  
 pause();  
 break;  
 case 2:  
 removeDroid();  
 pause();  
 break;  
 case 3:  
 showTeams();  
 pause();  
 break;  
 case 4:  
 startBattle();  
 pause();  
 break;  
 case 5:  
 saveBattleToFile();  
 pause();  
 break;  
 case 6:  
 loadBattleFromFile();  
 pause();  
 break;  
 case 7:  
 System.*out*.println("Вихід з програми...");  
 scanner.close();  
 return;  
 default:  
 System.*out*.println("Неправильний вибір. Спробуйте ще раз.");  
 }  
 }  
}  
  
*// Створення дроїда*private void createDroid() {  
 System.*out*.println("1. Алхімік (отруює ворогів та посилює союзників)");  
 System.*out*.println("2. Ассасін (швидкий воїн з шансом на ухилення від атаки та критчну шкоду)");  
 System.*out*.println("3. Цілитель (лікує та знімає негативні ефекти з союзників)");  
 System.*out*.println("3. Знемовлювач (знемовлює ворогів та повертає їм частину завданої шкоди)");  
 System.*out*.print("Оберіть тип дроїда: ");  
  
 int choice = scanner.nextInt();  
 scanner.nextLine();  
  
 Droid droid = null;  
 switch (choice) {  
 case 1:  
 droid = createAlchemistDroid();  
 break;  
 case 2:  
 droid = createAssassinDroid();  
 break;  
 case 3:  
 droid = createHealerDroid();  
 break;  
 case 4:  
 droid = createStunDroid();  
 break;  
 default:  
 System.*out*.println("Неправильний вибір.");  
 return;  
 }  
  
 System.*out*.println("1. Синя команда");  
 System.*out*.println("2. Червона команда");  
 System.*out*.print("Виберіть команду для цього дроїда: ");  
  
 int teamChoice = scanner.nextInt();  
 scanner.nextLine();  
 if (teamChoice == 1) {  
 teamBlue.add(droid);  
 System.*out*.println("Дроїд " + droid.getName() + " додано до синьої команди!");  
 } else if (teamChoice == 2) {  
 teamRed.add(droid);  
 System.*out*.println("Дроїд " + droid.getName() + " додано до червоної команди!");  
 } else {  
 System.*out*.println("Неправильний вибір команди.");  
 }  
}

private void saveBattleToFile() {  
 System.*out*.print("Введіть назву файлу для запису бою: ");  
 String filename = scanner.nextLine();  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {  
 writer.write("Синя команда:\n");  
 for (Droid d : teamBlue) {  
 writer.write(d.toString() + "\n");  
 }  
 writer.write("Червона команда:\n");  
 for (Droid d : teamRed) {  
 writer.write(d.toString() + "\n");  
 }  
 writer.write("Переможець: " + (blueTeamCopy.isEmpty() ? "Червоні" : "Сині") + "\n");  
 System.*out*.println("Бій успішно записано у файл.");  
 } catch (IOException e) {  
 System.*out*.println("Помилка при записі файлу: " + e.getMessage());  
 }  
}  
  
private void loadBattleFromFile() {  
 System.*out*.print("Введіть назву файлу для завантаження бою: ");  
 String filename = scanner.nextLine();  
 try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 System.*out*.println(line);  
 }  
 } catch (IOException e) {  
 System.*out*.println("Помилка при завантаженні файлу: " + e.getMessage());  
 }  
}

**BattleVisualizer**

public class BattleVisualizer {  
 public static final String *RESET* = "\u001B[0m"; *// Reset all styles* public static final String *RED* = "\u001B[31m"; *// Red text* public static final String *GREEN* = "\u001B[32m"; *// Green text* public static final String *YELLOW* = "\u001B[33m"; *// Yellow text* public static final String *BLUE* = "\u001B[34m"; *// Blue text* public static final String *BOLD* = "\u001B[1m"; *// Bold text* public static final String *ORANGE* = "\u001B[38;5;214m"; *// Orange text* public static final String *PINK* = "\u001B[38;5;13m"; *// Pink text* public static final String *MAROON* = "\u001B[38;5;88m"; *// Maroon text* public static final String *DARK\_BLUE* = "\u001B[38;5;19m"; *// Dark Blue text* public static final String *PURPLE* = "\u001B[35m";  
}

**Результат**

****

**Git:** <https://github.com/wixieee/AP_LPNU/tree/master/LR3PP>

**Висновок**

На цій лабораторній роботі я навчився працювати з дочірніми класами та отримав досвід роботи з файлами