МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
ИТМО»

Лабораторная работа №2
по дисциплине “Программирование”
Вариант 453

Студент:
Ковалев Александр Юрьевич
Преподаватель:
Бобрусь Александр Владимирович

Санкт-Петербург
2024

Оглавление

Задание

На основе базового класса Pokemon написать свои классы для заданных видов покемонов. Каждый вид покемона должен иметь один или два типа и стандартные базовые характеристики:

- очки здоровья (HP)
- атака (attack)
- защита (defense)
- специальная атака (special attack)
- специальная защита (special defense)
- скорость (speed)

Классы покемонов должны наследоваться в соответствии с цепочкой эволюции покемонов. На основе базовых
классов PhysicalMove, SpecialMove и StatusMove реализовать свои классы для заданных видов атак. Все разработанные классы, не имеющие наследников, должны быть реализованы таким образом, чтобы от них нельзя было наследоваться.
Атака должна иметь стандартные тип, силу (power) и точность (accuracy). Должны быть реализованы стандартные эффекты атаки. Назначить каждому виду покемонов атаки в соответствии с вариантом. Уровень покемона выбирается минимально необходимым для всех реализованных атак.
Используя класс симуляции боя Battle, создать 2 команды покемонов (каждый покемон должен иметь имя) и запустить бой.
Базовые классы и симулятор сражения находятся в jar-архиве (обновлен 9.10.2018, исправлен баг с добавлением атак и кодировкой). Документация в формате javadoc - здесь.
Информацию о покемонах, цепочках эволюции и атаках можно найти на сайтах http://poke-universe.ru, http://pokemondb.net, http://veekun.com/dex/pokemon

Цель работы: на простом примере разобраться с основными концепциями ООП и научиться использовать их в программах.

Что надо сделать (краткое описание)

1. Ознакомиться с документацией, обращая особое внимание на классы Pokemon и Move. При дальнейшем выполнении лабораторной работы читать документацию еще несколько раз.
2. Скачать файл Pokemon.jar. Его необходимо будет использовать как для компиляции, так и для запуска программы. Распаковывать его не надо! Нужно научиться подключать внешние jar-файлы к своей программе.
3. Написать минимально работающую программу и посмотреть как она работает.

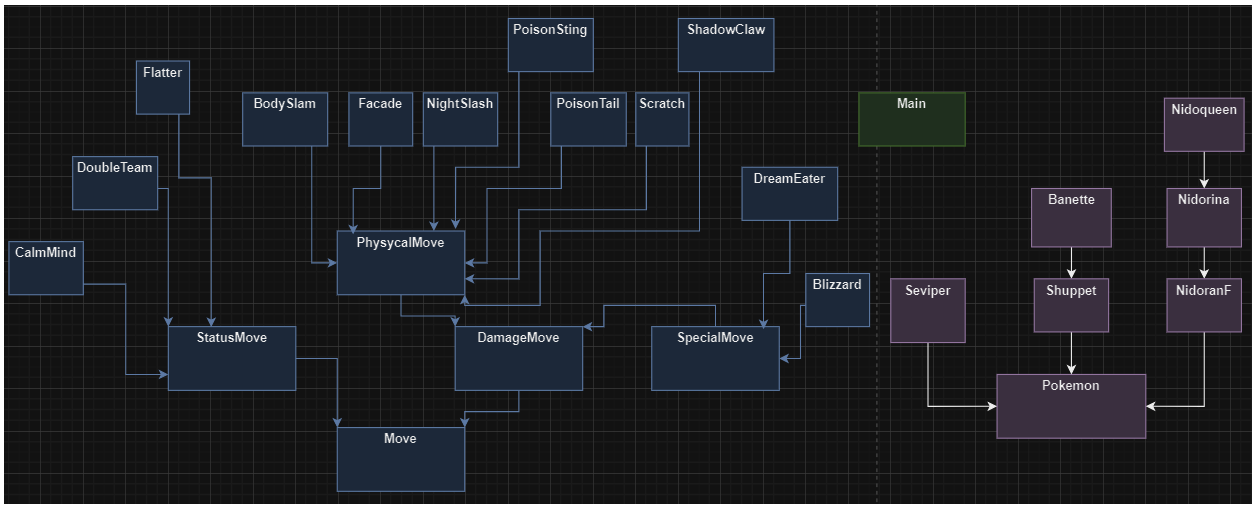
   ```
   Battle b = new Battle();
   Pokemon p1 = new Pokemon("Чужой", 1);
   Pokemon p2 = new Pokemon("Хищник", 1);
   b.addAlly(p1);
   b.addFoe(p2);
   b.go();
   ```
4. Создать один из классов покемонов для своего варианта. Класс должен наследоваться от базового класса Pokemon. В конструкторе нужно будет задать типы покемона и его базовые характеристики. После этого попробуйте добавить покемона в сражение.
5. Создать один из классов атак для своего варианта (лучше всего начать с физической или специальной атаки). Класс должен наследоваться от класса PhysicalMove или SpecialMove. В конструкторе нужно будет задать тип атаки, ее силу и точность. После этого добавить атаку покемону и проверить ее действие в сражении. Не забудьте переопределить метод describe, чтобы выводилось нужное сообщение.
6. Если действие атаки отличается от стандартного, например, покемон не промахивается, либо атакующий покемон также получает повреждение, то в классе атаки нужно дополнительно переопределить соответствующие методы (см. документацию). При реализации атак, которые меняют статус покемона (наследники StatusMove), скорее

всего придется разобраться с классом Effect. Он позволяет на один или несколько ходов изменить состояние покемона или модификатор его базовых характеристик.

7. Доделать все необходимые атаки и всех покемонов, распределить покемонов по командам, запустить сражение.

Ваши покемоны:

UML-Диаграмма классов



Исходный код программы

Атаки

Blizzard

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class Blizzard extends SpecialMove{
    public Blizzard(){
        super(Type.ICE,110,70);
    }
```

```
    @Override
    protected void applyOppEffects(Pokemon pokemon) {
        pokemon.setCondition(new
Effect().chance(0.1).condition(Status.FREEZE));
    }
    @Override
    protected String describe(){
        return "использует атаку Blizzard";
    }
}
```

Body Slam

```
package Attacks;

import ru.ifmo.se.pokemon.*;

public class BodySlam extends PhysicalMove {
    public BodySlam() {
        super(Type.NORMAL,60,100);
    }

    @Override
    protected String describe() {
        return "Использует атаку Body Slam";
    }

    protected void applyOppEffects(Pokemon p){
        p.setCondition(new Effect().chance(0.3).condition(Status.PARALYZE));
    }
}
```

Calm Mind

```
package Attacks;

import ru.ifmo.se.pokemon.*;

public class CalmMind extends StatusMove {
    public CalmMind(){
        super(Type.PSYCHIC,0,0);
    }

    protected void applyOppEffects(Pokemon p){
        p.setCondition(new Effect().turns(1).stat(Stat.SPECIAL_ATTACK, 6));
        p.setCondition(new Effect().turns(1).stat(Stat.SPECIAL_DEFENSE, 6));
    }
    @Override
    protected String describe(){
        return "использует атаку Calm Mind";
    }
}
```

Double Team

```
package Attacks;

import ru.ifmo.se.pokemon.*;

public class DoubleTeam extends StatusMove {
```

```
    public DoubleTeam(){
        super(Type.NORMAL,0,0);
    }
    protected void applySelfEffects(Pokemon p){
        p.setCondition(new Effect().turns(1).stat(Stat.EVASION, 6));
    }
    @Override
    protected String describe(){
        return "использует атаку Double Team";
    }
}
```

Dream Eater

```
package Attacks;

import ru.ifmo.se.pokemon.*;

public class DreamEater extends SpecialMove {
    public DreamEater(){
        super(Type.PSYCHIC,100,100);
    }

    protected void applySelfEffects(Pokemon p) {
        p.addEffect(new Effect().stat(Stat.HP, (int)
p.getStat(Stat.SPECIAL_ATTACK)));
    }
    @Override
    protected String describe() {
        return "использует атаку Dream Eater";
    }
}
```

Facade

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class Facade extends PhysicalMove{
    public Facade(){
        super(Type.NORMAL,70,100);
    }
    protected double calcCriticalHit(Pokemon att, Pokemon def){
        if ((att.getCondition()==Status.BURN)||
(att.getCondition()==Status.POISON)||(att.getCondition()==Status.PARALYZE)){
            System.out.println("Critical hit!");
            return 2.0;
        } else {
            return 1.0;
        }
    }
    @Override
    protected String describe(){
        return "использует атаку Facade";
    }
}
```

Flatter

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class Flatter extends StatusMove{
```

```
    public Flatter(){
        super(Type.DARK,0,0);
    }

    protected void applyOppEffects(Pokemon pokemon) {
        pokemon.setCondition(new
Effect().turns(1).condition(Status.PARALYZE));
    }

    protected void applySelfEffects(Pokemon pokemon) {
        pokemon.setCondition(new
Effect().turns(1).stat(Stat.SPECIAL_ATTACK,6));
    }
    @Override
    protected String describe(){
        return "использует атаку Flatter";
    }
}
```

Night Slash

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class NightSlash extends PhysicalMove{
    public NightSlash(){
        super(Type.DARK,70,100);
    }

    protected double calcCriticalHit(Pokemon var1, Pokemon var2) {
        if (var1.getStat(Stat.SPEED)*3 / 512.0 > Math.random()) {
            System.out.println("Critical hit!");
            return 2.0;
        } else {
            return 1.0;
        }
    }
    @Override
    protected String describe(){
        return "использует атаку Night Slash";
    }
}
```

Poison Sting

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class PoisonSting extends PhysicalMove{
    public PoisonSting(){
        super(Type.POISON,15,100);
    }

    protected void applyOppEffects(Pokemon pokemon) {
        pokemon.setCondition(new
Effect().chance(0.3).condition(Status.POISON));
    }

    @Override
    protected String describe(){
        return "использует атаку Poison Sting";
    }
```

```
}
```

Poison Tail

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class PoisonTail extends PhysicalMove{
    public PoisonTail(){
        super(Type.POISON,50,100);
    }

    @Override
    protected String describe(){
        return "использует атаку Poison Tail";
    }

    protected void applyOppEffects(Pokemon p){
        p.setCondition(new Effect().chance(0.1).condition(Status.POISON));
    }

    protected double calcCriticalHit(Pokemon var1, Pokemon var2) {
        if (var1.getStat(Stat.SPEED)*3 / 512.0 > Math.random()) {
            System.out.println("Critical hit!");
            return 2.0;
        } else {
            return 1.0;
        }
    }
}
```

Scratch

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class Scratch extends PhysicalMove{
    public Scratch(){
        super(Type.NORMAL,40,100);
    }

    protected double calcTypeEffect(Pokemon pokemon, Pokemon pokemon1) {
        return 1.0;
    }
    @Override
    protected String describe(){
        return "использует атаку Scratch";
    }
}
```

Shadow Claw

```
package Attacks;
import ru.ifmo.se.pokemon.*;

public class ShadowClaw extends PhysicalMove{
    public ShadowClaw(){
        super(Type.GHOST,70,100);
    }

    protected double calcCriticalHit(Pokemon var1, Pokemon var2) {
```

```
        if (var1.getStat(Stat.SPEED)*3 / 512.0 > Math.random()) {
            System.out.println("Critical hit!");
            return 2.0;
        } else {
            return 1.0;
        }
    }

    @Override
    protected String describe(){
        return "использует атаку Shadow Claw";
    }
}
```

Покемоны

Seviper

```
package Pokemons;

import Attacks.*;
import ru.ifmo.se.pokemon.*;

public final class Seviper extends Pokemon {
    public Seviper(String name, int level) {
        super(name, level);
        this.setType(Type.POISON);
        this.setStats(73, 100, 60, 100, 60, 65);
        this.setMove(new BodySlam(), new NightSlash(), new PoisonTail(), new
Facade());
    }
}
```

Shuppet

```
package Pokemons;

import ru.ifmo.se.pokemon.*;
import Attacks.*;
public class Shuppet extends Pokemon {
    public Shuppet(String name, int level) {
        super(name, level);
        this.setStats(44, 75, 35, 63, 33, 45);
        this.setType(Type.GHOST);
        this.setMove(new CalmMind(), new DoubleTeam(), new DreamEater());
    }
}
```

Banette

```
package Pokemons;

import ru.ifmo.se.pokemon.*;
import Attacks.*;

public final class Banette extends Shuppet {
    public Banette(String name, int level) {
        super(name, level);
        this.setType(Type.GHOST);
        this.setStats(64,115,65,83,63,65);
        this.setMove(new ShadowClaw());
```

```
    }
}
```

NidoranF

```
package Pokemons;

import ru.ifmo.se.pokemon.*;
import Attacks.*;

public class NidoranF extends Pokemon {
    public NidoranF(String name, int level){
        super(name,level);
        this.setType(Type.POISON);
        this.setStats(55,47,52,40,40,41);
        this.setMove(new Blizzard(), new PoisonSting());
    }
}
```

Nidorina

```
package Pokemons;
import ru.ifmo.se.pokemon.*;
import Attacks.*;

public class Nidorina extends NidoranF{
    public Nidorina(String name,int level){
        super(name,level);
        this.setType(Type.POISON);
        this.setStats(70,62,67,55,55, 56);
        this.setMove(new Flatter());
    }
}
```

Nidoqueen

```
package Pokemons;
import ru.ifmo.se.pokemon.*;
import Attacks.*;
public final class Nidoqueen extends Nidorina {
    public Nidoqueen(String name, int level){
        super(name,level);
        this.setType(Type.POISON,Type.GROUND);
        this.setStats(90,92,87,75,85,76);
        this.setMove(new Scratch());
    }
}
```

Main-класс

```
import ru.ifmo.se.pokemon.*;
import Pokemons.*;


public class Main {
    public static void main(String[] args) {
        Battle b = new Battle();
        Banette p1 = new Banette("Кукла", 50);
        Nidoqueen p2 = new Nidoqueen("Танк", 50);
        NidoranF p3 = new NidoranF("Хомяк", 50);
        Nidorina p4 = new Nidorina("Дикобраз", 50);
        Seviper p5 = new Seviper("Змеюга", 50);
        Shuppet p6 = new Shuppet("Призрак", 50);
```

```
        b.addAlly(p1);
        b.addAlly(p2);
        b.addAlly(p3);
        b.addFoe(p4);
        b.addFoe(p5);
        b.addFoe(p6);
        b.go();
    }
}
```

Результат работы программы:

Banette Кукла из команды черных вступает в бой!
Nidorina Дикобраз из команды фиолетовых вступает в бой!
Banette Кукла использует атаку Shadow Claw.
Critical hit!
Nidorina Дикобраз теряет 65 здоровья.

Nidorina Дикобраз промахивается

Banette Кукла использует атаку Shadow Claw.
Nidorina Дикобраз теряет 47 здоровья.

Nidorina Дикобраз промахивается

Banette Кукла использует атаку Shadow Claw.
Critical hit!
Nidorina Дикобраз теряет 88 здоровья.
Nidorina Дикобраз теряет сознание.
Seviper Змеюга из команды фиолетовых вступает в бой!
Seviper Змеюга использует атаку Facade.
Seviper Змеюга теряет 19 здоровья.

Seviper Змеюга использует атаку Facade.
Seviper Змеюга теряет 31 здоровья.

Banette Кукла использует атаку Shadow Claw.
Seviper Змеюга теряет 36 здоровья.

Seviper Змеюга использует атаку Poison Tail.
Critical hit!
Banette Кукла теряет 35 здоровья.

Banette Кукла использует атаку Shadow Claw.
Critical hit!

Banette Кукла теряет 152 здоровья.
Оба покемона теряют сознание.
Nidoqueen Танк из команды черных вступает в бой!
Nidoqueen Танк использует атаку Scratch.
Seviper Змеюга теряет 15 здоровья.

Seviper Змеюга использует атаку Night Slash.
Critical hit!
Nidoqueen Танк теряет 43 здоровья.

Nidoqueen Танк использует атаку Scratch.
Seviper Змеюга теряет 17 здоровья.

Seviper Змеюга использует атаку Night Slash.
Nidoqueen Танк теряет 23 здоровья.

Nidoqueen Танк использует атаку Scratch.
Seviper Змеюга теряет 17 здоровья.

Seviper Змеюга использует атаку Night Slash.
Nidoqueen Танк теряет 25 здоровья.

Nidoqueen Танк использует атаку Scratch.
Seviper Змеюга теряет 17 здоровья.
Seviper Змеюга теряет сознание.
Shuppet Призрак из команды фиолетовых вступает в бой!
Nidoqueen Танк использует атаку Scratch.
Shuppet Призрак теряет 23 здоровья.
Shuppet Призрак не замечает воздействие типа NORMAL

Shuppet Призрак промахивается

Nidoqueen Танк использует атаку Scratch.
Критический удар!
Shuppet Призрак теряет 43 здоровья.
Shuppet Призрак не замечает воздействие типа NORMAL

Shuppet Призрак промахивается

Nidoqueen Танк использует атаку Scratch.
Shuppet Призрак теряет 17 здоровья.
Shuppet Призрак не замечает воздействие типа NORMAL

Shuppet Призрак промахивается

Nidoqueen Танк использует атаку Scratch.
Критический удар!
Shuppet Призрак теряет 57 здоровья.
Shuppet Призрак не замечает воздействие типа NORMAL
Shuppet Призрак теряет сознание.
В команде фиолетовых не осталось покемонов.
Команда черных побеждает в этом бою!

Вывод

В ходе этой лабораторной работы я познакомился с основами Объектно-Ориентированного программирования, работой с классами, конструкторами, полями и модификаторами доступа, научился подключать внешние jar-архивы к программе.