

НИУ ИТМО

Программирование

Отчет по лабораторной работе №2

Вариант 78643

Преподаватель: Иманзаде Фахри Рашидович
Выполнил: Носов Александр Дмитриевич R3137

Санкт-Петербург
2022 год

ITMO
Лабораторная работа №2
Вариант №78643
Программирование Java
Носов Александр Дмитриевич
Иманзаде Фахри Рашидович
2022

Текст задания.

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.

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](#).


Информацию о покемонах, цепочках эволюции и атаках можно найти на сайтах.

Покемоны.

Введите вариант: 78643

Ваши покемоны:

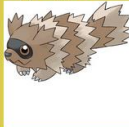
Terrakion



Атаки:

- Quick Attack
- Stone Edge
- Swagger
- Poison Jab


Zigzagoon



Атаки:

- Thunderbolt
- Tail Whip
- Rest


Linoone



Атаки:

- Thunderbolt
- Tail Whip
- Rest
- Shadow Claw


Trapinch



Атаки:

- Mud-Slap
- Swagger


Vibrava



Атаки:

- Mud-Slap
- Swagger
- Bug Buzz

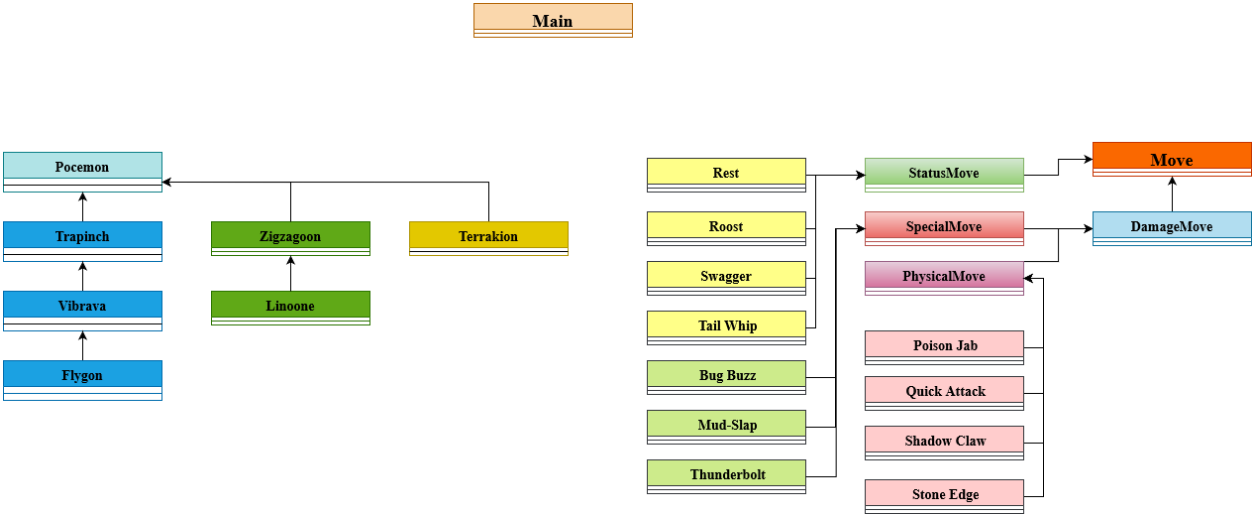
Flygon



Атаки:

- Mud-Slap
- Swagger
- Bug Buzz
- Roost

Диаграмма классов реализованной объектной модели.



Исходный код программы.

Class Main.

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

public class Main {
    public static void main(String[] args)
    {
        Battle b = new Battle();
        Pokemon p1 = new Terrakion("Великолепный
покемон", 5);
        Pokemon p2 = new Zigzagoon("Попик", 7);
        Pokemon p3 = new Linoone("Пегас", 6);
        Pokemon p4 = new Vibrava("Данила", 5);
        Pokemon p5 = new Trapinch("Саня", 8);
        Pokemon p6 = new Flygon("Лаба", 7);

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

Блок покемонов.

Class Terrakion.

```
package pokemons;

import attacks.physicalAttacks.Poison_Jab;
import attacks.physicalAttacks.Quick_Attack;
import attacks.physicalAttacks.Stone_Edge;
import attacks.statusAttacs.Swagger;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Terrakion extends Pokemon {
    public Terrakion(String name, int lvl)
    {
        super(name, lvl);
        this.setType(Type.ROCK, Type.FIGHTING);
        this.setStats(91, 129, 90, 72, 90, 108);
        this.addMove(new Quick_Attack());
        this.addMove(new Stone_Edge());
        this.addMove(new Swagger());
        this.addMove(new Poison_Jab());
    }
}
```

Class Zigzagoon.

```
package pokemons;

import attacks.specialAttakcs.Thunderbolt;
import attacks.statusAttacs.Rest;
import attacks.statusAttacs.Tail_Whip;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Zigzagoon extends Pokemon {
    public Zigzagoon(String name, int lvl)
    {
        super(name, lvl);
        this.setType(Type.NORMAL);
        this.setStats(38, 30, 41, 30, 41, 60);
        this.addMove(new Thunderbolt());
        this.addMove(new Tail_Whip());
        this.addMove(new Rest());
    }
}
```

Class Linoon.

```
package pokemons;

import attacks.physicalAttacks.Shadow_Claw;
import ru.ifmo.se.pokemon.Type;

public class Linoone extends Zigzagoon
{
    public Linoone(String name, int lvl)
    {
        super(name, lvl);
        this.setType(Type.NORMAL);
        this.setStats(78, 70, 61, 50, 61, 100);
        this.addMove(new Shadow_Claw());
    }
}
```

Class Trapinch.

```
package pokemons;

import attacks.specialAttacks.Mud_Slap;
import attacks.statusAttacks.Swagger;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Trapinch extends Pokemon {
    public Trapinch(String name, int lvl){
        super(name, lvl);
        this.setType(Type.GROUND);
        this.setStats(45, 100, 45, 45, 45, 10);
        this.addMove(new Swagger());
        this.addMove(new Mud_Slap());
    }
}
```

Class Vibrava.

```
package pokemons;

import attacks.specialAttacks.Bug_Buzz;
import ru.ifmo.se.pokemon.Type;

public class Vibrava extends Trapinch{
    public Vibrava(String name, int lvl){
        super(name, lvl);
        this.setType(Type.GROUND, Type.DRAGON);
        this.setStats(50, 70, 50, 50, 50, 70);
        this.addMove(new Bug_Buzz());
    }
}
```

Class Flygon.

```
package pokemons;

import attacks.statusAttacs.Rest;
import ru.ifmo.se.pokemon.Type;

public class Flygon extends Vibrava{
    public Flygon(String name, int lvl){
        super(name, lvl);
        this.setType(Type.GROUND, Type.DRAGON);
        this.setStats(80, 100, 80, 80, 80, 100);
        this.addMove(new Roost());
    }
}
```

Блок атак.

Блок статусных атак.

Class Rest.

```
package attacks.statusAttacs;

import ru.ifmo.se.pokemon.*;

public class Rest extends StatusMove {
    public Rest() {super(Type.PSYCHIC, 0, 100);}

    @Override
    protected String describe() {
        return "Применяет Rest";
    }

    @Override
    protected void applySelfEffects(Pokemon pokemon) {
        Effect d = new Effect();
        d.stat(Stat.HP);
        Effect.sleep(pokemon);
        pokemon.addEffect(d);
        super.applySelfEffects(pokemon);
    }
}
```

Class Roost.

```
package attacks.statusAttacs;

import ru.ifmo.se.pokemon.*;

public class Roost extends StatusMove {
    public Roost() {super(Type.FLYING, 0, 0);}

    @Override
    protected String describe() {return "Применяет Roost";}

    @Override
    protected void applySelfEffects(Pokemon pokemon) {
        Effect d = new Effect();
        d.stat(Stat.HP);
        pokemon.addEffect(d);
        super.applySelfEffects(pokemon);
    }
}
```


Class Swagger.

```
package attacks.statusAttacs;

import ru.ifmo.se.pokemon.*;

public class Swagger extends StatusMove {

    public Swagger() { super(Type.NORMAL, 0, 85); }

    @Override
    protected String describe() { return "Применяет Swagger"; }

    @Override
    protected void applySelfEffects(Pokemon p) {
        Effect d = new Effect().stat(Stat.ATTACK, 2);
        p.addEffect(d);
        super.applyOppEffects(p);
    }

    @Override
    protected void applyOppEffects(Pokemon pokemon) {
        pokemon.confuse();
    }
}
```

Class Tail_Whip

```
package attacks.statusAttacs;

import ru.ifmo.se.pokemon.*;

public class Tail_Whip extends StatusMove {
    public Tail_Whip() { super(Type.NORMAL, 0, 100); }

    @Override
    protected String describe() {
        return "Применяет Tail_Whip!";
    }

    @Override
    protected void applyOppEffects(Pokemon pokemon) {
        Effect d = new Effect().stat(Stat.DEFENSE, -1);
        pokemon.addEffect(d);
        super.applyOppEffects(pokemon);
    }
}
```

Блок специальных атак.

Class Bug_Buzz

```
package attacks.specialAttacks;

import ru.ifmo.se.pokemon.*;

public class Bug_Buzz extends SpecialMove {
    public Bug_Buzz() {super(Type.BUG, 90, 100);}

    @Override
    protected String describe() {return "Применяет Bug Buzz";}

    @Override
    protected void applyOppEffects(Pokemon pokemon) {
        Effect d = new Effect().chance(0.1).stat(Stat.SPECIAL_DEFENSE, -1);
        pokemon.addEffect(d);
        super.applyOppEffects(pokemon);
    }
}
```

Class Mud_Slap

```
package attacks.specialAttacks;

import ru.ifmo.se.pokemon.*;

public class Mud_Slap extends SpecialMove {
    public Mud_Slap() {super(Type.GROUND, 20, 100);}

    @Override
    protected String describe() {return "Применяет Mud_Slap";}

    @Override
    protected void applyOppEffects(Pokemon pokemon) {
        Effect d = new Effect().stat(Stat.ACCURACY, -1);
        pokemon.addEffect(d);
        super.applyOppEffects(pokemon);
    }
}
```

Class Thunderbolt

```
package attacks.specialAttacks;

import ru.ifmo.se.pokemon.*;

public class Thunderbolt extends SpecialMove {
    public Thunderbolt() {super(Type.ELECTRIC, 90, 100);}

    @Override
    protected String describe() {
        return "Применяет Thunderbolt";
    }

    @Override
    protected void applyOppEffects(Pokemon pokemon) {
        Effect d = new Effect().chance(0.1).condition(Status.PARALYZE);
        pokemon.addEffect(d);
        super.applyOppEffects(pokemon);
    }
}
```

Блок физических атак.

Class Poison_Jab

```
package attacks.physicalAttacks;

import ru.ifmo.se.pokemon.*;

public class Poison_Jab extends PhysicalMove {
    public Poison_Jab() {super(Type.POISON, 80, 100);}

    @Override
    protected String describe() {
        return "Применяет Poison_Jab";
    }

    @Override
    protected void applyOppEffects(Pokemon pokemon) {
        Effect d = new Effect().chance(0.3).condition(Status.POISON);
        pokemon.addEffect(d);
        super.applyOppEffects(pokemon);
    }
}
```

Class Quick_Attack

```
package attacks.physicalAttacks;

import ru.ifmo.se.pokemon.*;

public class Quick_Attack extends PhysicalMove {
    public Quick_Attack() {super(Type.NORMAL, 40, 100, 1, 1);}

    @Override
    protected String describe() {
        return "Применяет Quick_Attack";
    }
}
```

Class Shadow_Claw

```
package attacks.physicalAttacks;

import ru.ifmo.se.pokemon.PhysicalMove;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Stat;
import ru.ifmo.se.pokemon.Type;

public class Shadow_Claw extends PhysicalMove {

    public Shadow_Claw(){super(Type.GHOST, 70, 100);}

    @Override
    protected String describe() {return "Применяет Shadow_Claw";}

    @Override
    protected double calcCriticalHit(Pokemon pokemon, Pokemon pokemon1) {
        if (pokemon.getStat(Stat.SPEED) / 128.0 > Math.random()) {
            System.out.println("Увеличенный критический удар!");
            return 2.0;
        } else {
            return 1.0;
        }
    }
}
```

Class Stone_Edge

```
package attacks.physicalAttacks;

import ru.ifmo.se.pokemon.*;

public class Stone_Edge extends PhysicalMove {
    public Stone_Edge(){ super(Type.ROCK, 100, 80);}

    @Override
    protected String describe() {
        return "Применяет Stone_Edge";
    }

    @Override
    protected double calcCriticalHit(Pokemon pokemon, Pokemon pokemon1) {
        if (pokemon.getStat(Stat.SPEED) * 3 / 512.0 > Math.random()) {
            System.out.println("Увеличенный критический удар!");
            return 2.0;
        } else {
            return 1.0;
        }
    }
}
```

Результаты работы программы.

(107 строк)

Terrakion Великолепный покемон из команды синих вступает в бой!

Zigzagoon Попик из команды фиолетовых вступает в бой!

Terrakion Великолепный покемон Применяет Swagger.

Zigzagoon Попик Применяет Thunderbolt.

Terrakion Великолепный покемон теряет 7 здоровья.

Terrakion Великолепный покемон Применяет Poison_Jab.

Zigzagoon Попик теряет 12 здоровья.

Zigzagoon Попик Применяет Thunderbolt.

Terrakion Великолепный покемон теряет 7 здоровья.

Terrakion Великолепный покемон Применяет Swagger.

Zigzagoon Попик растерянно попадает по себе.

Zigzagoon Попик теряет 4 здоровья.

Terrakion Великолепный покемон Применяет Swagger.

Zigzagoon Попик Применяет Thunderbolt.

Критический удар!

Terrakion Великолепный покемон теряет 10 здоровья.

Terrakion Великолепный покемон Применяет Swagger.

Zigzagoon Попик Применяет Rest.

Zigzagoon Попик засыпает

Terrakion Великолепный покемон Применяет Swagger.

Terrakion Великолепный покемон Применяет Poison_Jab.

Zigzagoon Попик теряет 21 здоровья.

Zigzagoon Попик теряет сознание.

Vibrava Данила из команды фиолетовых вступает в бой!

Terrakion Великолепный покемон Применяет Quick_Attack.

Vibrava Данила теряет 15 здоровья.

Vibrava Данила Применяет Swagger.

Terrakion Великолепный покемон Применяет Poison_Jab.

Vibrava Данила теряет 9 здоровья.

Vibrava Данила отравлен

Vibrava Данила теряет сознание.

Trapinch Саня из команды фиолетовых вступает в бой!

Terrakion Великолепный покемон растерянно попадает по себе.

Terrakion Великолепный покемон теряет 8 здоровья.

Terrakion Великолепный покемон теряет сознание.

Linoone Пегас из команды синих вступает в бой!

Linoone Пегас Применяет Rest.

Linoone Пегас засыпает

Trapinch Саня Применяет Swagger.

Linoone Пегас Применяет Tail_Whip!.

Trapinch Саня Применяет Mud_Slap.

Linoone Пегас теряет 6 здоровья.

Linoone Пегас Применяет Shadow_Claw.

Увеличенный критический удар!

Trapinch Саня теряет 12 здоровья.

Trapinch Саня Применяет Mud_Slap.

Linoone Пегас теряет 7 здоровья.

Linoone Пегас Применяет Tail_Whip!.

Trapinch Саня Применяет Mud_Slap.

Linoone Пегас теряет 7 здоровья.

Linoone Пегас Применяет Rest.

Linoone Пегас засыпает

Trapinch Саня Применяет Mud_Slap.

Linoone Пегас теряет 8 здоровья.

Linoone Пегас теряет сознание.

Flygon Лаба из команды синих вступает в бой!

Flygon Лаба Применяет Swagger.

Trapinch Саня Применяет Swagger.

Flygon Лаба растерянно попадает по себе.

Flygon Лаба теряет 2 здоровья.

Trapinch Саня Применяет Mud_Slap.

Flygon Лаба теряет 6 здоровья.

Flygon Лаба Применяет Mud_Slap.

Trapinch Саня теряет 7 здоровья.

Trapinch Саня борется с соперником.

Flygon Лаба теряет 12 здоровья.

Trapinch Саня теряет 3 здоровья.

Flygon Лаба растерянно попадает по себе.

Flygon Лаба теряет 3 здоровья.

Trapinch Саня борется с соперником.

Flygon Лаба теряет 13 здоровья.

Trapinch Саня теряет 3 здоровья.

Flygon Лаба теряет сознание.

В команде синих не осталось покемонов.

Команда фиолетовых побеждает в этом бою!

Process finished with exit code 0

Дополнительное задание.

Текст задания: написать программу, которая при вводе в конструкторе покемона вместо имени пустую строку “null”, то программа должна сообщить об ошибке, а также если в конструкторе покемона в поле уровня ввести отрицательное значение, то не включать этого покемона в бой (программа должна сообщить, если покемонов не будет хватать для начала боя).

Для реализации этой задачи я создал класс FixPokemons (наследника класса Pokemon), в котором добавляем два метода(проверку имени и уровня)

```
package pokemons;

import ru.ifmo.se.pokemon.Pokemon;

public class FixPokemons extends Pokemon {
    final boolean assignet_lvl;
    final boolean assignet_name;

    public FixPokemons(String name, int lvl){
        super(name, lvl);
        if (lvl < 1){
            assignet_lvl = false;
            System.out.println("Уровень покемона" + name + "не входит в требуемый диапазон (lvl > 1)");
        }else {
            assignet_lvl = true;
        }

        if (name == null){
            assignet_name = false;
        }else{
            assignet_name = true;
        }
    }

    public boolean get_lvl(){return assignet_lvl;}
    public boolean get_name(){return assignet_name;}
}
```

Также был создан класс FixBattle, в котором бой запускался при особых условиях.

```
import pokemons.FixPokemons;
import ru.ifmo.se.pokemon.Battle;

public class FixBattle {
    Battle pokemonBattle = new Battle();

    private int allyCnt_lvl = 0;
    private int foeCnt_lvl = 0;
    private int allyCnt_name = 0;
    private int foeCnt_name = 0;

    public FixBattle(FixPokemons[] pokemonsAlly, FixPokemons[] pokemonsFoe) {
        for (FixPokemons fixPokemons : pokemonsAlly) {
            if (fixPokemons.get_lvl() != 0) {
                pokemonBattle.addAlly(fixPokemons);
                allyCnt_lvl++;
            }
            if (fixPokemons.get_name() == false) {
                foeCnt_name++;
            }
        }
        for (FixPokemons fixPokemons : pokemonsFoe) {
            if (fixPokemons.get_lvl() != 0) {
                pokemonBattle.addFoe(fixPokemons);
                foeCnt_lvl++;
            }
            if (fixPokemons.get_name() == false) {
                allyCnt_name++;
            }
        }
    }

    public void go() {
        if (allyCnt_name != 0 || foeCnt_name != 0) {
            System.out.println("Покемону не присвоено имя");
        } else if (allyCnt_lvl == 0 && foeCnt_lvl == 0) {
            System.out.println("Уровень ниодного из покемонов не удовлетворяет условию, бой не может состояться");
        } else if (allyCnt_lvl + foeCnt_lvl == 1) {
            System.out.println("Одного покемона недостаточно для проведения боя");
        } else if (allyCnt_lvl == 0 || foeCnt_lvl == 0) {
            System.out.println("У всех покемонов одной из команд присвоены уровни, неудвлетворяющие условию, поэтому бой не может начаться");
        } else {
            pokemonBattle.go();
        }
    }
}
```

После этого был исправлен класс Main(в него добавлена новая реализация боя), а также в классах покемонов был исправлен предок с Pokemon на FixPokemon.

```
import pokemons.*;

public class Main {
    public static void main(String[] args)
    {
        FixPokemons Terrakion = new Terrakion("Великолепный покемон",-5);
        FixPokemons Zigzagoon = new Zigzagoon("Суир", -7);
        FixPokemons Linoone = new Linoone("Пегас", 6);
        FixPokemons Vibrava = new Vibrava("Данила", -5);
        FixPokemons Trapinch = new Trapinch("Саня", 8);
        FixPokemons Flygon = new Flygon("Лаба", -7);

        FixPokemons[] Foe = {Terrakion, Zigzagoon, Trapinch};
        FixPokemons[] Ally = {Linoone, Vibrava, Flygon};
        FixBattle b = new FixBattle(Ally, Foe);
        b.go();
    }
}
```

Выводы.

В процессе выполнения лабораторной работы я:

- Разобрался в основных понятиях ООП (наследование, инкапсуляция, полиморфизм)
- Узнал о модификаторах доступа.
- Разграничил для себя классы и объекты классов.
- Научился разным способам работы с методами.
- Применил полученные знания для создания цепочек наследования классов покемонов и атак.