Федеральное государственное автономное образовательное учреждение высшего образования «Научно-образовательная корпорация ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа по программированию №2

вариант: 310909

Преподаватель: Наумова Надежда Александровна

Выполнил: Степанов Арсений

Группа: P3109

Санкт-Петербург, 2022г

Оглавление

Цели

Освоить базовую работу с объектно-ориентированным программированием и пакетами, изучить принципы и основные термины и выполнить на основе полученных знаний лабораторную работу.

Задание

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.

Код классов

Main:

```java
package lab2;

import lab2.pokemon.*;
import ru.ifmo.se.pokemon.*;

public class Main {
    public static void main(String[] args) {
        Battle b = new Battle();
        Pokemon p1 = new MeloettaPokemon("Арбуз", 1);
```

```
        Pokemon p2 = new PetitilPokemon("Капуста", 1);
        Pokemon p3 = new LilligantPokemon("Дыня", 1);
        Pokemon p4 = new SewaddlePokemon("Кабачок", 1);
        Pokemon p5 = new SwadloonPokemon("Тыква", 1);
        Pokemon p6 = new LeavannyPokemon("Патиссон", 1);


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

        b.go();
    }
}
```

LeavannyPokemon:

```
package lab2.pokemon;

import lab2.move.Moves;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class LeavannyPokemon extends SwadloonPokemon {
    public LeavannyPokemon(String name, int lvl) {
        super(name, lvl);
        addType(Type.BUG);
        addType(Type.GRASS);
        setStats(75, 103, 80, 70, 80, 92);
        setMove(Moves.FACADE, Moves.STRING_SHOT, Moves.GRASS_WHISTLE, Moves.S-
TEEL_WING);
    }
}
```

LilligantPokemon:

```
package lab2.pokemon;

import lab2.move.Moves;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class LilligantPokemon extends PetitilPokemon {
    public LilligantPokemon(String name, int lvl ) {
        super(name, lvl);
        setStats(70, 65, 75, 110, 75, 90);
        setMove(Moves.DREAM_EATER, Moves.REST, Moves.FACADE,
Moves.SWORDS_DANCE);
    }
}
```

MeloettaPokemon:

```
package lab2.pokemon;

import lab2.move.Moves;
import ru.ifmo.se.pokemon.Pokemon;
```

```
import ru.ifmo.se.pokemon.Type;

public class MeloettaPokemon extends Pokemon {
    public MeloettaPokemon(String name, int lvl ) {
        super(name, lvl);
        addType(Type.NORMAL);
        addType(Type.PSYCHIC);
        setStats(100, 77, 77, 128, 128, 90);
        setMove(Moves.CALM_MIND, Moves.PSYCHIC, Moves.TEETER_DANCE,
Moves.SHADOW_CLAW);
    }
}
```

PetitilPokemon:

```
package lab2.pokemon;

import lab2.move.Moves;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class PetitilPokemon extends Pokemon {
    public PetitilPokemon(String name, int lvl ) {
        super(name, lvl);
        addType(Type.GRASS);
        setStats(45, 35, 50, 70, 50, 30);
        setMove(Moves.DREAM_EATER, Moves.REST, Moves.FACADE);
    }
}
```

SewaddlePokemon:

```
package lab2.pokemon;

import lab2.move.Moves;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class SewaddlePokemon extends Pokemon {
    public SewaddlePokemon(String name, int lvl ) {
        super(name, lvl);
        addType(Type.BUG);
        addType(Type.GRASS);
        setStats(70, 65, 75, 110, 75, 90);
        setMove(Moves.FACADE, Moves.STRING_SHOT);
    }
}
```

SwadloonPokemon:

```
package lab2.pokemon;

import lab2.move.Moves;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class SwadloonPokemon extends SewaddlePokemon {
    public SwadloonPokemon(String name, int lvl) {
        super(name, lvl);
        setStats(55, 63, 90, 50, 80, 42);
```

```
        setMove(Moves.FACADE, Moves.STRING_SHOT, Moves.GRASS_WHISTLE);
    }
}
```

CalmMindMove:

```
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class CalmMindMove extends StatusMove {
    public CalmMindMove() {
        super(Type.PSYCHIC, 0, 1.0);
    }

    @Override
    protected void applySelfEffects(Pokemon user) {
        user.addEffect(new Effect().turns(-1).stat(Stat.SPECIAL_ATTACK, 1));
        user.addEffect(new Effect().turns(-1).stat(Stat.SPECIAL_DEFENSE, 1));
        System.out.println(user + " повышает свою специальную атаку и защиту!");
    }

    @Override
    protected String describe() {
        return "успокаивает свой разум";
    }
}
```

DreamEaterMove:

```
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class DreamEaterMove extends SpecialMove {
    public DreamEaterMove() {
        super(Type.PSYCHIC, 100, 1.0);
    }

    private long healAmount = 0;

    @Override
    protected void applyOppDamage(Pokemon victim, double v) {
        boolean isSleeping = victim.getCondition().equals(Status.SLEEP);
        if (!isSleeping) {
            System.out.println("Но " + victim + " не спит");
            v = 0;
        }
        healAmount = Math.round(v);
        super.applyOppDamage(victim, v);
    }

    @Override
    protected void applySelfDamage(Pokemon user, double v) {
        super.applySelfDamage(user, -healAmount);
    }

    @Override
    protected String describe() {
        return "насылает кошмары";
```

```
    }
}
```

FacadeMove:

```
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class FacadeMove extends PhysicalMove {
    public FacadeMove() {
        super(Type.NORMAL, 70, 1.0);
    }

    @Override
    protected double calcBaseDamage(Pokemon att, Pokemon def) {
        Status condition = att.getCondition();
        return super.calcBaseDamage(att, def)
                * (condition.equals(Status.POISON) || condition.equals(Sta-
tus.BURN) || condition.equals(Status.PARALYZE) ? 2.0 : 1.0);
    }

    @Override
    protected String describe() {
        return "наносит удар";
    }
}
```

GrassWhistleMove:

```
package lab2.move;

import ru.ifmo.se.pokemon.Effect;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.StatusMove;
import ru.ifmo.se.pokemon.Type;

public class GrassWhistleMove extends StatusMove {
    GrassWhistleMove() {
        super(Type.GRASS, 0 , 0.55);
    }

    @Override
    protected void applyOppEffects(Pokemon victim) {
        Effect.sleep(victim);
    }

    @Override
    protected String describe() {
        return "говорит про пользу сна";
    }
}
```

PsychicMove:

```java
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class PsychicMove extends SpecialMove {
    public PsychicMove() {
        super(Type.PSYCHIC, 90.0, 1.0);
    }

    @Override
    protected void applyOppEffects(Pokemon victim) {
        if (0.1 > Math.random()) {
            victim.addEffect(new Effect().turns(-1).stat(Stat.SPECIAL_DEFENSE, -
1));
            System.out.println("Специальная защита " + victim + " снижена!");

        }
    }

    @Override
    protected String describe() {
        return "включает увлекательную передачу по рен-тв";
    }
}
```

RestMove:

```java
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class RestMove extends StatusMove {
    public RestMove() {
        super(Type.PSYCHIC, 0, 1.0);
    }

    @Override
    protected void applySelfEffects(Pokemon user) {
        user.restore();
        user.setCondition(new Effect().condition(Status.SLEEP).at-
tack(0.0).turns(2));
    }

    @Override
    protected String describe() {
        return "вкинулся и откинулся";
    }
}
```

ShadowClawMove:

```java
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class ShadowClawMove extends PhysicalMove {
    public ShadowClawMove() {
        super(Type.GHOST, 70, 1.0);
    }
```

```
    @Override
    protected double calcCriticalHit(Pokemon att, Pokemon def) {
        if (0.125 > Math.random()) {
            System.out.println(def + " получил критический урон!");
            return 2.0;
        } else {
            return 1.0;
        }
    }

    @Override
    protected String describe() {
        return "использует теневой клык";
    }
}
```

SteelWingMove:

```
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class SteelWingMove extends PhysicalMove {
    public SteelWingMove() {
        super(Type.STEEL, 70, 0.9);
    }

    @Override
    protected void applySelfEffects(Pokemon user) {
        user.addEffect(new Effect().turns(-1).stat(Stat.DEFENSE,
1).chance(0.1));
    }

    @Override
    protected String describe() {
        return "наносит удар";
    }
}
```

StringShotMove:

```
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class StringShotMove extends StatusMove {
    public StringShotMove() {
        super(Type.BUG, 0, 0.95);
    }

    @Override
    protected void applyOppEffects(Pokemon victim) {
        victim.addEffect(new Effect().turns(-1).stat(Stat.SPEED, 2));
        System.out.println("Скорость " + victim + " снижена");
    }

    @Override
    protected String describe() {
        return "выстреливает паутинкой";
```

```
    }
}
```

SwordsDanceMove:

```
package lab2.move;

import ru.ifmo.se.pokemon.*;

public class SwordsDanceMove extends StatusMove {
    public SwordsDanceMove() {
        super(Type.NORMAL, 0, 1.0);
    }

    @Override
    protected void applySelfEffects(Pokemon user) {
        user.addEffect(new Effect().stat(Stat.ATTACK, 2).turns(-1));
    }

    @Override
    protected String describe() {
        return "исполняет воинственную жигу-дрыгу";
    }
}
```

TeeterDanceMove:

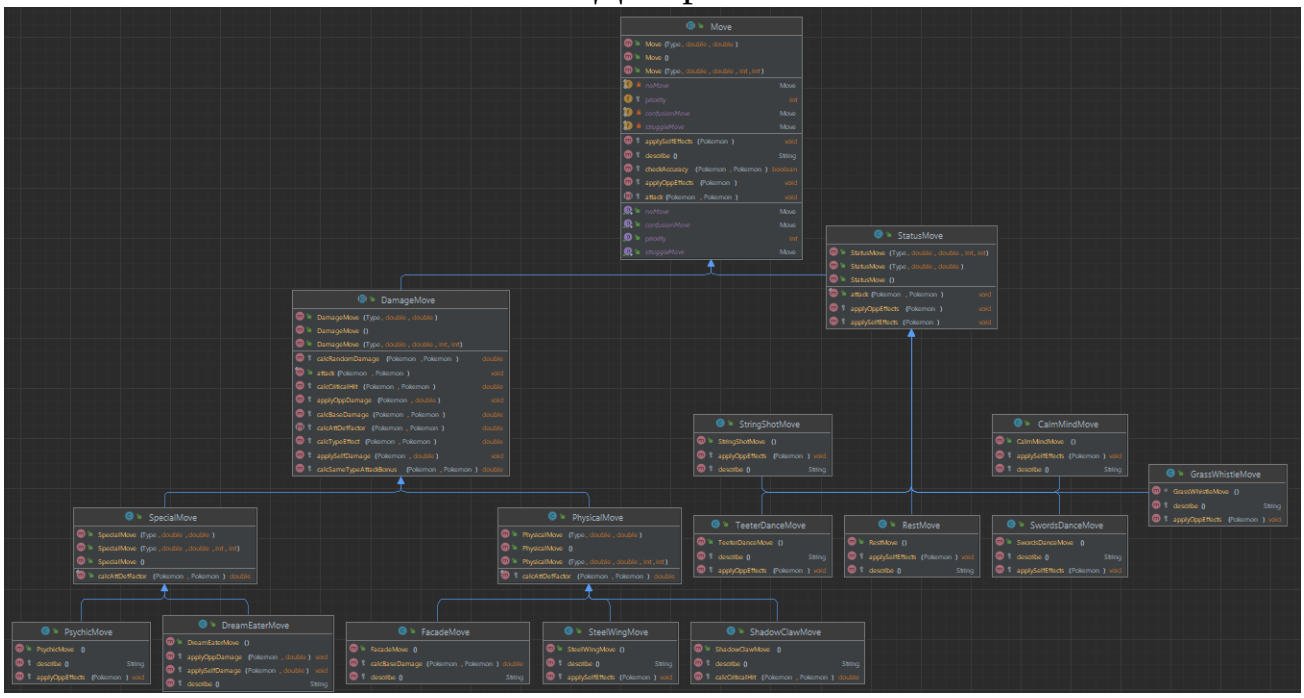
```
package lab2.move;

import ru.ifmo.se.pokemon.Effect;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.StatusMove;
import ru.ifmo.se.pokemon.Type;

public class TeeterDanceMove extends StatusMove {
    public TeeterDanceMove() {
        super(Type.NORMAL, 0, 1.0);
    }

    @Override
    protected void applyOppEffects(Pokemon victim) {
        Effect.confuse(victim);
        System.out.println(victim + " сконфужен данным обстоятельством");
    }

    @Override
    protected String describe() {
        return "нереально флексит";
    }
}
```

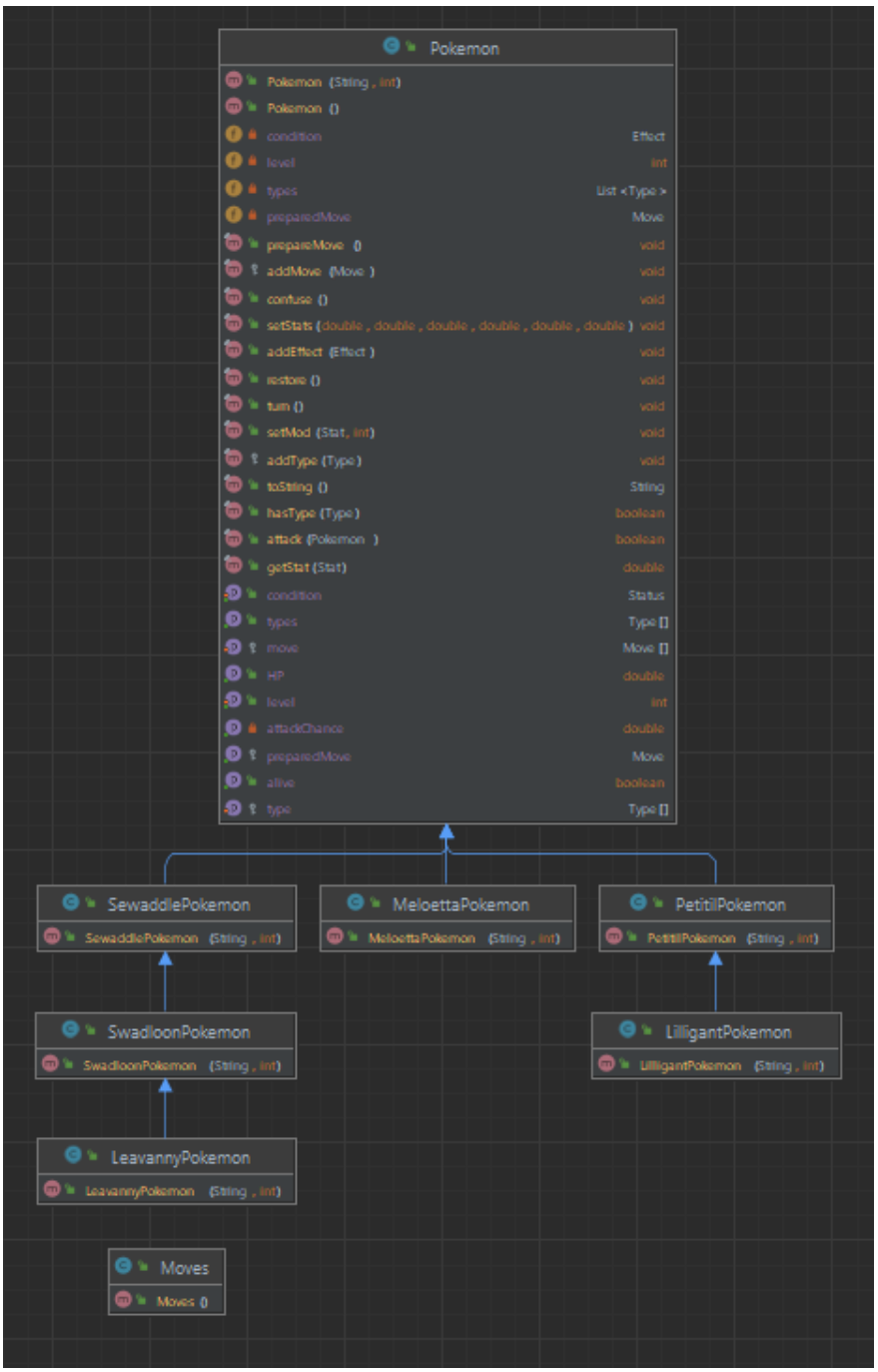
Moves:

```
package lab2.move;

import ru.ifmo.se.pokemon.Move;

public class Moves {
    public static Move PSYCHIC = new PsychicMove();
    public static Move CALM_MIND = new CalmMindMove();
```

```
    public static Move TEETER_DANCE = new TeeterDanceMove();
    public static Move SHADOW_CLAW = new ShadowClawMove();
    public static Move DREAM_EATER = new DreamEaterMove();
    public static Move FACADE = new FacadeMove();
    public static Move REST = new RestMove();
    public static Move SWORDS_DANCE = new SwordsDanceMove();
    public static Move STRING_SHOT = new StringShotMove();
    public static Move GRASS_WHISTLE = new GrassWhistleMove();
    public static Move STEEL_WING = new SteelWingMove();
}
```

UML Диаграммы





Вывод программы

MeloettaPokemon Арбуз из команды полосатых вступает в бой!
SewaddlePokemon Кабачок из команды красных вступает в бой!

SewaddlePokemon Кабачок выстреливает паутинкой.
Скорость SewaddlePokemon Кабачок снижена

SewaddlePokemon Кабачок выстреливает паутинкой.
Скорость SewaddlePokemon Кабачок снижена

SewaddlePokemon Кабачок наносит удар.
MeloettaPokemon Арбуз теряет 4 здоровья.

MeloettaPokemon Арбуз включает увлекательную передачу по рен-тв.
SewaddlePokemon Кабачок теряет 4 здоровья.

SewaddlePokemon Кабачок наносит удар.
MeloettaPokemon Арбуз теряет 5 здоровья.

MeloettaPokemon Арбуз включает увлекательную передачу по рен-тв.
SewaddlePokemon Кабачок теряет 6 здоровья.

SewaddlePokemon Кабачок выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз включает увлекательную передачу по рен-тв.
SewaddlePokemon Кабачок теряет 5 здоровья.
Специальная защита SewaddlePokemon Кабачок снижена!
SewaddlePokemon Кабачок теряет сознание.
SwadloonPokemon Тыква из команды красных вступает в бой!
MeloettaPokemon Арбуз использует теневой клык.
SwadloonPokemon Тыква теряет 6 здоровья.

SwadloonPokemon Тыква выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз использует теневой клык.
SwadloonPokemon Тыква теряет 3 здоровья.

SwadloonPokemon Тыква говорит про пользу сна.
MeloettaPokemon Арбуз засыпает

SwadloonPokemon Тыква выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз успокаивает свой разум.
MeloettaPokemon Арбуз повышает свою специальную атаку и защиту!

SwadloonPokemon Тыква выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз успокаивает свой разум.
MeloettaPokemon Арбуз повышает свою специальную атаку и защиту!

SwadloonPokemon Тыква наносит удар.
MeloettaPokemon Арбуз теряет 3 здоровья.

MeloettaPokemon Арбуз успокаивает свой разум.
MeloettaPokemon Арбуз повышает свою специальную атаку и защиту!

SwadloonPokemon Тыква выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз нереально флексит.
SwadloonPokemon Тыква сконфужен данным обстоятельством

SwadloonPokemon Тыква выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз использует теневой клык.
SwadloonPokemon Тыква получил критический урон!
SwadloonPokemon Тыква теряет 8 здоровья.
SwadloonPokemon Тыква теряет сознание.
LeavannyPokemon Патиссон из команды красных вступает в бой!
MeloettaPokemon Арбуз успокаивает свой разум.
MeloettaPokemon Арбуз повышает свою специальную атаку и защиту!

LeavannyPokemon Патиссон говорит про пользу сна.
MeloettaPokemon Арбуз засыпает

LeavannyPokemon Патиссон говорит про пользу сна.

MeloettaPokemon Арбуз нереально флексит.
LeavannyPokemon Патиссон сконфужен данным обстоятельством

LeavannyPokemon Патиссон промахивается

MeloettaPokemon Арбуз успокаивает свой разум.
MeloettaPokemon Арбуз повышает свою специальную атаку и защиту!

LeavannyPokemon Патиссон выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз нереально флексит.
LeavannyPokemon Патиссон сконфужен данным обстоятельством

LeavannyPokemon Патиссон выстреливает паутинкой.
Скорость MeloettaPokemon Арбуз снижена

MeloettaPokemon Арбуз использует теневой клык.
LeavannyPokemon Патиссон получил критический урон!
LeavannyPokemon Патиссон теряет 11 здоровья.

LeavannyPokemon Патиссон говорит про пользу сна.
MeloettaPokemon Арбуз засыпает

LeavannyPokemon Патиссон растерянно попадает по себе.
LeavannyPokemon Патиссон теряет 5 здоровья.
LeavannyPokemon Патиссон теряет сознание.
В команде красных не осталось покемонов.
Команда полосатых побеждает в этом бою!

Process finished with exit code 0

Итоги

Я научился работать с классами и базовыми принципами объектно-ориентированного программирования, а также с пакетами. Полученные в процессе работы знания являются фундаментальными и будут полезны при будущей работе и учёбе.