

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

Лабораторная работа №2
по дисциплине “Программирование”

Студент: Студент группы Р3132, Юксель Хамза
Преподаватель: Пименов Данила Дмитриевич

Санкт-Петербург
2024

ЗАДАНИЕ:	3
ИСХОДНЫЙ КОД ПРОГРАММЫ:	4
PACKAGE-MYMOVES:	4
<i>myMoves.phsicalMoves:</i>	4
DoubleSlap.java:	4
PetalBlizzard.java:	5
Present.java:	5
<i>myMoves.specialMoves:</i>	6
DracoMeteor.java:	6
DragonRage.java:	7
DreamEater.java:	7
IceBeam.java:	8
<i>myMoves.statusMoves:</i>	9
Confide.java:	9
DoubleTeam.java:	10
Swagger.java:	11
Rest.java:	11
PACKAGE-MYPOKEMONS:	12
<i>Blissey.java:</i>	12
<i>Chansey.java:</i>	13
<i>Happiny.java:</i>	13
<i>Kyurem.java:</i>	14
<i>Lilligant.java:</i>	14
<i>Petilil.java:</i>	15
PACKAGE-PROG_LAB2:	15
<i>Program.java:</i>	15
ДИАГРАММА КЛАССОВ РЕАЛИЗОВАННОЙ ОБЪЕКТНОЙ МОДЕЛИ (UNIFIED MODELING LANGUAGE UML DIAGRAM)	16
РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ:	17
ВЫВОДЫ ПО РАБОТЕ	19

Задание:

Лабораторная работа #2

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>

Комментарии

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.

Введите вариант:

Ваши покемоны:



Рисунок 1

Исходный Код Программы:

Package-myMoves:

myMoves.physicalMoves:

DoubleSlap.java:

```
package myMoves.physicalMoves;

import ru.ifmo.se.pokemon.*;

public class DoubleSlap extends PhysicalMove {
    private double initialPower;

    public DoubleSlap(double pow, double acc) {
        super(Type.NORMAL, pow, acc);

        this.initialPower = pow;
    }

    @Override
    protected void applyOppEffects(Pokemon p) {

        double chance = Math.random();
        int hits;

        if (chance < 0.375) {          // 3/8 chance for 2 hits
            hits = 2;
        } else if (chance < 0.75) {    // 3/8 chance for 3 hits
            hits = 3;
        } else if (chance < 0.875) {  // 1/8 chance for 4 hits
            hits = 4;
        } else {                      // 1/8 chance for 5 hits
            hits = 5;
        }
        this.hits = hits; // Store hits for describe method

        Effect e = new Effect().turns(hits).stat(Stat.ATTACK, (int)(hits*initialPower));
        p.addEffect(e);
    }

    private int hits; // To store number of hits for description

    @Override // overridden original one
    protected String describe(){
        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1] + " Hits " + hits + " times";
    }
}
```

```
}
```

PetalBlizzard.java:

```
package myMoves.physicalMoves;

import ru.ifmo.se.pokemon.*;

public class PetalBlizzard extends PhysicalMove {

    public PetalBlizzard(double pow, double acc) {
        super(Type.GRASS, pow, acc);
    }

    @Override
    protected void applyOppDamage(Pokemon def, double damage) {
        super.applyOppDamage(def, power);

    }

    @Override // overridden original one
    protected String describe(){
        // class.pokemon.SimleMove
        String info[] = this.getClass().toString().split("\\.");
        return "Using "+ info[info.length-1];
        // does SimpleMove

    }
}
```

Present.java:

```
package myMoves.physicalMoves;

import ru.ifmo.se.pokemon.*;

public class Present extends PhysicalMove{
    private double initialPower;

    public Present(double pow, double acc) {
        super(Type.NORMAL, pow, acc);

        this.initialPower = pow;
    }

    @Override
    protected void applyOppEffects(Pokemon p) {
```

```

// Random chance to either deal damage or heal
if (Math.random() < 0.8) {
    // Deal random damage between 40-120 power
    this.initialPower = 40 + (int)(Math.random() * 81);
} else {
    // Heal for 1/4 max HP
    double healing = p.getStat(Stat.HP) * 0.25;
    p.setMod(Stat.HP, -(int)healing);
}
}

@Override // overridden original one
protected String describe(){
    String info[] = this.getClass().toString().split("\\.");
    return "Using " + info[info.length-1];
}

```

}

myMoves.specialMoves:

DracoMeteor.java:

```

package myMoves.specialMoves;

import ru.ifmo.se.pokemon.*;

public class DracoMeteor extends SpecialMove {

    public DracoMeteor(double pow, double acc){
        super(Type.DRAGON,pow, acc);
    }

    @Override
    protected void applySelfEffects(Pokemon p) {
        //Draco Meteor deals damage but lowers the user's Special
        //Attack by two stages after attacking.
        Effect e = new Effect().stat(Stat.SPECIAL_ATTACK,-2);
        p.addEffect(e);
        System.out.println(p.toString() + " Special Attack -2");
    }

    @Override // overridden original one
    protected String describe(){
        // class.pokemon.SimleMove
    }
}

```

```

        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1];
        // does SimpleMove

    }
}

```

DragonRage.java:

```

package myMoves.specialMoves;

import ru.ifmo.se.pokemon.*;

public class DragonRage extends SpecialMove{

    public DragonRage(double pow, double acc){
        super(Type.DRAGON,pow, acc);
    }

    @Override
    protected void applyOppDamage(Pokemon def, double damage) {
        //super.applyOppDamage(def, damage);
        def.setMod(Stat.HP, 40);

    }

    @Override // overridden original one
    protected String describe(){
        // class.pokemon.SimleMove
        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1];
        // does SimpleMove

    }
}

```

DreamEater.java:

```

package myMoves.specialMoves;

import ru.ifmo.se.pokemon.*;

public class DreamEater extends SpecialMove{
    public DreamEater(double pow, double acc) {
        super(Type.PSYCHIC, pow, acc);
    }
}

```

```

    }

    @Override
    protected void applyOppDamage(Pokemon def, double pow) {
        if(def.getCondition() == Status.SLEEP) {
            super.applyOppDamage(def, pow);
        }
    }
    @Override
    protected void applySelfDamage(Pokemon att, double pow) {
        att.setMod(Stat.HP, (int) (-(power / 2)));
    }

    @Override // overridden original one
    protected String describe(){
        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1];
    }
}

```

[IceBeam.java](#):

```

package myMoves.specialMoves;

import ru.ifmo.se.pokemon.*;

public class IceBeam extends SpecialMove {

    public IceBeam(double pow, double acc){
        super(Type.ICE,pow, acc);
    }

    //Ice Beam deals damage and has a 10% chance of freezing the target.
    @Override
    protected void applyOppEffects(Pokemon p) {
        if(Math.random() <= 0.1) {
            Effect.freeze(p);
        }
    }

    @Override // overridden original one
    protected String describe(){
        // class.pokemon.SimleMove
        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1];
    }
}

```

```

        // does SimpleMove

    }

}

package myMoves.Happiny;

import ru.ifmo.se.pokemon.*;

public class DreamEater extends SpecialMove{
    public DreamEater(double pow, double acc) {
        super(Type.PSYCHIC, pow, acc);
    }

    @Override
    protected void applyOppDamage(Pokemon def, double pow) {
        if(def.getCondition() == Status.SLEEP) {
            super.applyOppDamage(def, pow);
        }
    }

    @Override
    protected void applySelfDamage(Pokemon att, double pow) {
        att.setMod(Stat.HP, (int) (-(power / 2)));
    }

    @Override // overridden original one
    protected String describe(){
        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1];
    }
}

```

myMoves.statusMoves:

Confide.java:

```

package myMoves.statusMoves;

import ru.ifmo.se.pokemon.*;

public class Confide extends StatusMove {
    public Confide(double pow, double acc){
        super(Type.NORMAL,pow, acc);
    }
}

```

```

@Override
protected void applyOppEffects(Pokemon p) {
    Effect e = new Effect().stat(Stat.SPECIAL_ATTACK, -1);
    p.addEffect(e);
    System.out.println(p.toString() + " Enemy Special Attack decreased by 1 stage");

}

@Override // overridden original one
protected String describe(){
    // class.pokemon.SimleMove
    String info[] = this.getClass().toString().split("\\.");
    return "Using " + info[info.length-1];
    // does SimpleMove

}
}

```

DoubleTeam.java:

```

package myMoves.statusMoves;

import ru.ifmo.se.pokemon.*;

public class DoubleTeam extends StatusMove{
    public DoubleTeam(double pow, double acc){
        super(Type.NORMAL,pow, acc);
    }

@Override
protected void applySelfEffects(Pokemon p) {
    Effect e = new Effect().stat(Stat.EVASION, 1);
    p.addEffect(e);
    System.out.println(p.toString() + " Evasion increased by 1 stage");

}

@Override // overridden original one
protected String describe(){
    // class.pokemon.SimleMove
    String info[] = this.getClass().toString().split("\\.");
    return "Using " + info[info.length-1];
    // does SimpleMove
}

```

```
    }
}
```

Swagger.java:

```
package myMoves.statusMoves;

import ru.ifmo.se.pokemon.*;

public class Swagger extends StatusMove{

    public Swagger(double pow, double acc) {
        super(Type.NORMAL, pow, acc);
    }

    @Override
    protected void applyOppEffects(Pokemon p) {
        Effect.confuse(p);
    }

    @Override
    protected void applySelfEffects(Pokemon p) {
        Effect e = new Effect().stat(Stat.ATTACK, 2);
        p.addEffect(e);
        System.out.println(p.toString() + " attack increased 2 points");
    }

    @Override // overridden original one
    protected String describe(){
        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1];
    }
}
```

Rest.java:

```
package myMoves.statusMoves;

import ru.ifmo.se.pokemon.*;

public class Rest extends StatusMove {
```

```

    public Rest(double pow, double acc){
        super(Type.PSYCHIC, pow, acc);
    }

    @Override
    protected void applySelfEffects(Pokemon p) {
        // Put user to sleep for 2 turns
        Effect.sleep(p);
        Effect.sleep(p);
        // Calculate and apply full healing (negative value for healing)
        double maxHP = p.getStat(Stat.HP);
        double currentHP = p.getHP();
        double healAmount = maxHP - currentHP;
        p.setMod(Stat.HP, -(int)healAmount);
    }

    @Override // overridden original one
    protected String describe(){
        String info[] = this.getClass().toString().split("\\.");
        return "Using " + info[info.length-1];
    }
}

```

Package-myPokemons:

Blissey.java:

```
package myPokemons;
```

```

import myMoves.Blissey.*;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Blissey extends Pokemon{
    public Blissey(String name, int level) {
        super(name, level);

        super.setType(Type.NORMAL);
        super.setStats(255, 10, 10, 75, 135, 55);

        DreamEater dreamEater = new DreamEater(100,100);
        Present present = new Present(0,90);
        Swagger swagger = new Swagger(0,85);
        DoubleSlap doubleSlap = new DoubleSlap(15,85);
    }
}

```

```
        super.setMove(dreamEater,present,swagger,doubleSlap);
    }
}
```

Chansey.java:

```
package myPokemons;

import myMoves.Chansey.*;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Chansey extends Pokemon {
    public Chansey(String name, int level) {
        super(name, level);

        super.setType(Type.NORMAL);
        super.setStats(250, 5, 5, 35, 105, 50);

        DreamEater dreamEater = new DreamEater(100,100);
        Present present = new Present(0,90);
        Swagger swagger = new Swagger(0,85);

        super.setMove(dreamEater,present,swagger);
    }
}
```

Happiny.java:

```
package myPokemons;

import myMoves.Happiny.*;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Happiny extends Pokemon {
    public Happiny(String name, int level) {
        super(name, level);

        super.setType(Type.NORMAL);
        super.setStats(100, 5, 5, 15, 65, 30);

        DreamEater dreamEater = new DreamEater(100,100);
        Swagger swagger = new Swagger(0,85);

        super.setMove(dreamEater,swagger);
    }
}
```

```
    }
}
```

Kyurem.java:

```
package myPokemons;

import myMoves.Kyurem.*;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Kyurem extends Pokemon {

    public Kyurem(String name, int level) {
        super(name,level);

        super.setType(Type.DRAGON,Type.ICE); //iki tane tipi var
        super.setStats(125, 130, 90, 130, 90, 95);

        DracoMeteor dracoMeteor = new DracoMeteor(130,90);
        IceBeam iceBeam = new IceBeam(90,100);
        DragonRage dragonRage = new DragonRage(0,100);
        Swagger swagger = new Swagger(0,85);

        super.setMove(dracoMeteor, iceBeam, dragonRage, swagger);

    }
}
```

Lilligant.java:

```
package myPokemons;

import myMoves.Lilligant.*;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Lilligant extends Pokemon {
    public Lilligant(String name, int level) {
        super(name,level);

        super.setType(Type.GRASS);
        super.setStats(70,60,75,110,75,90);

        Confide confide = new Confide(0,0);
        DoubleTeam doubleTeam = new DoubleTeam(0,0);
        Rest rest = new Rest(0,0);
        PetalBlizzard petalBlizzard = new PetalBlizzard(90,100);
```

```
        super.setMove(rest,doubleTeam,confide,petalBlizzard);
    }
}
```

Petilil.java:

```
package myPokemons;

import myMoves.Petilil.*;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Petilil extends Pokemon {

    public Petilil(String name, int level) {
        super(name,level);

        super.setType(Type.GRASS); //one type. = grass
        super.setStats(45, 35, 50, 70, 50, 30);

        Confide confide = new Confide(0,0);
        DoubleTeam doubleTeam = new DoubleTeam(0,0);
        Rest rest = new Rest(0,0);

        super.setMove(rest,doubleTeam,confide);
    }

}
```

Package-prog_lab2:

Program.java:

```
package prog_lab2;

import myPokemons.*;
import ru.ifmo.se.pokemon.Battle;

public class Program {

    public static void main(String[] args) {
        Battle b = new Battle();

        // Create Pokémon instances
```

```

Petilil petilil = new Petilil("Legend", 1);
Kyurem kyurem = new Kyurem("Precious", 1);
Blissey blissey = new Blissey("Useless", 1);
Chansey chansey = new Chansey("My Egg", 1);
Happiny happiny = new Happiny("Happiness", 1);
Lilligant lilligant = new Lilligant("Green Coconut", 1);

// Add Pokémon to teams and count them
int allyCount = 0;
int foeCount = 0;

b.addAlly(kyurem);
allyCount++;

b.addAlly(blissey);
allyCount++;

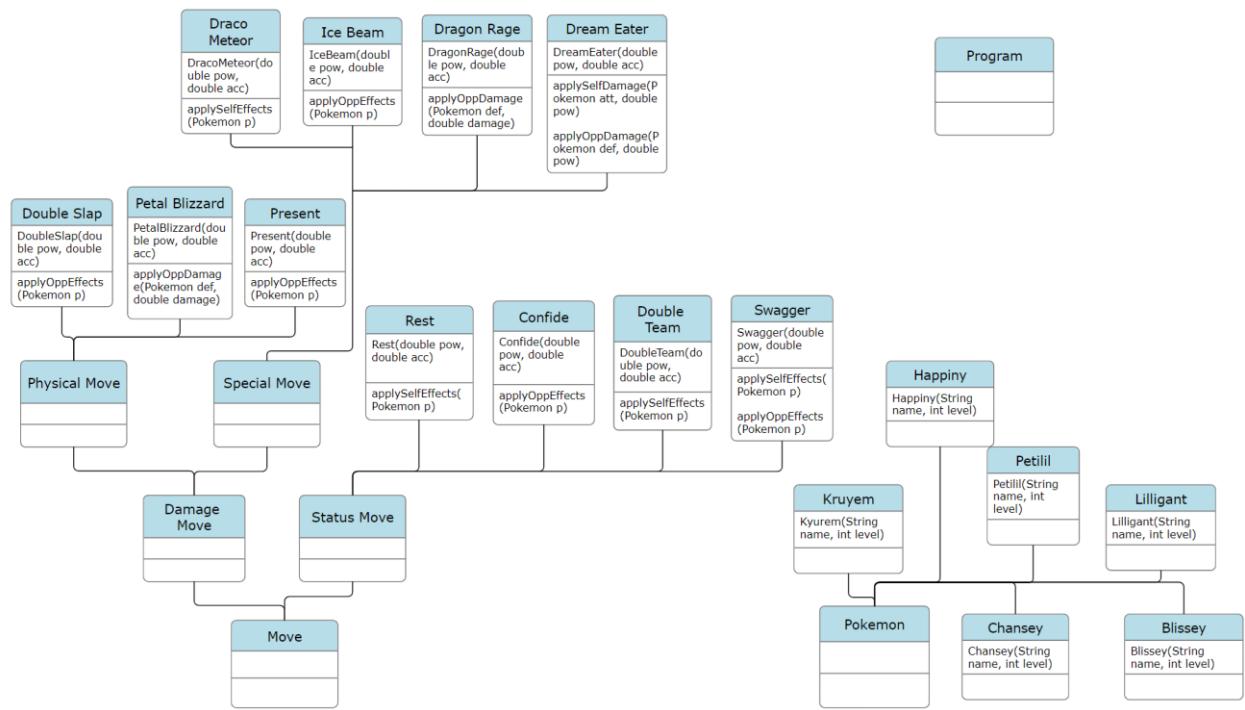
b.addAlly(happiny);
allyCount++;

// Check if both teams have at least one Pokémon
if (allyCount == 0 || foeCount == 0) {
    System.err.println("Error: Both teams must have at least one Pokémon to start the
battle.");
    return; // Stop execution if teams are not properly set up
}

// Start the battle
b.go();
}
}

```

Диаграмма классов реализованной объектной модели
(Unified Modeling Language UML diagram)



Результат Работы Программы:

```
[s408078@helios ~/lab2/build]$ java -jar MyJar.jar
Picked up _JAVA_OPTIONS: -XX:MaxHeapSize=1G -XX:MaxMetaspaceSize=128m
Kyurem Precious из команды полосатых вступает в бой!
Chansey My Egg из команды синих вступает в бой!
Kyurem Precious Using Swagger.
Kyurem Precious attack increased by 2 stage

Chansey My Egg Using Present.
Kyurem Precious теряет 5 здоровья.

Kyurem Precious Using DragonRage.
Chansey My Egg теряет 40 здоровья.
Chansey My Egg теряет сознание.
Petilil Legend из команды синих вступает в бой!
Kyurem Precious Using DracoMeteor.
Petilil Legend теряет 10 здоровья.
Kyurem Precious Special Attack -2

Petilil Legend промахивается

Kyurem Precious Using Swagger.
Kyurem Precious attack increased by 2 stage

Petilil Legend промахивается

Kyurem Precious Using DracoMeteor.
Критический удар!
Petilil Legend восстанавливает 1 здоровья.
Kyurem Precious Special Attack -2

Petilil Legend промахивается

Kyurem Precious Using Swagger.
Kyurem Precious attack increased by 2 stage

Petilil Legend промахивается

Kyurem Precious Using IceBeam.
Petilil Legend теряет 11 здоровья.
Petilil Legend теряет сознание.
Lilligant Green Coconat из команды синих вступает в бой!
Kyurem Precious Using DracoMeteor.
Lilligant Green Coconat теряет 5 здоровья.
Kyurem Precious Special Attack -2
```

Kyurem Precious Special Attack -2

Petilil Legend промахивается

Kyurem Precious Using Swagger.

Kyurem Precious attack increased by 2 stage

Petilil Legend промахивается

Kyurem Precious Using IceBeam.

Petilil Legend теряет 11 здоровья.

Petilil Legend теряет сознание.

Lilligant Green Coconat из команды синих вступает в бой!

Kyurem Precious Using DracoMeteor.

Lilligant Green Coconat теряет 5 здоровья.

Kyurem Precious Special Attack -2

Lilligant Green Coconat промахивается

Kyurem Precious Using DragonRage.

Lilligant Green Coconat теряет 40 здоровья.

Lilligant Green Coconat теряет сознание.

В команде синих не осталось покемонов.

Команда полосатых побеждает в этом бою!

[s408078@helios ~/lab2/build]\$ |

Рисунок 2&3&4 Битва покемонов

Выводы по Работе

В результате этого проекта я научился использовать внешние jar-файлы в своем исходном коде. В этой лабораторной работе я создал простую битву для своих покемонов, научился писать методы и метод consturcter. Благодаря этому интерактивному примеру я познакомился с основами ООП (объектно-ориентированного программирования). С помощью диаграммы UML я могу увидеть связи между моими классами и суперклассами.