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

Факультет программной инженерии и компьютерной техники.

Дисциплина: Программирование

Лабораторная работа №2

Вариант № 11122

Выполнил: Машкин Григорий Андреевич
Преподаватель: Гиря Максим Дмитриевич
Группа: P3130

2023г.
Санкт-Петербург

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.

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.

Покемоны и атаки:

Hippodwon: Focus Energy, Substitute

Servine: Thunder Shock, Frustration, Petal Dance, Rock Tomb

Cyndaquil: Hydro Pump, Taunt, Thunder Shock

Skorupi: Rage, Focus Energy, Substitute

Drapion: Rage, Focus Energy, Substitute, Confusion

Oddish: Hydro Pump, Taunt, Thunder Shock, Vine Whip

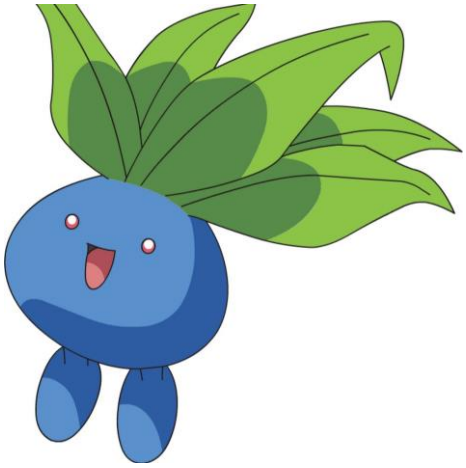
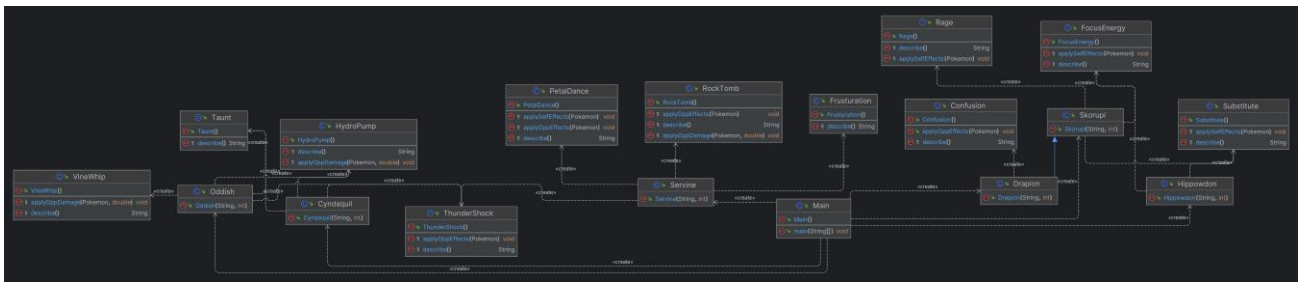
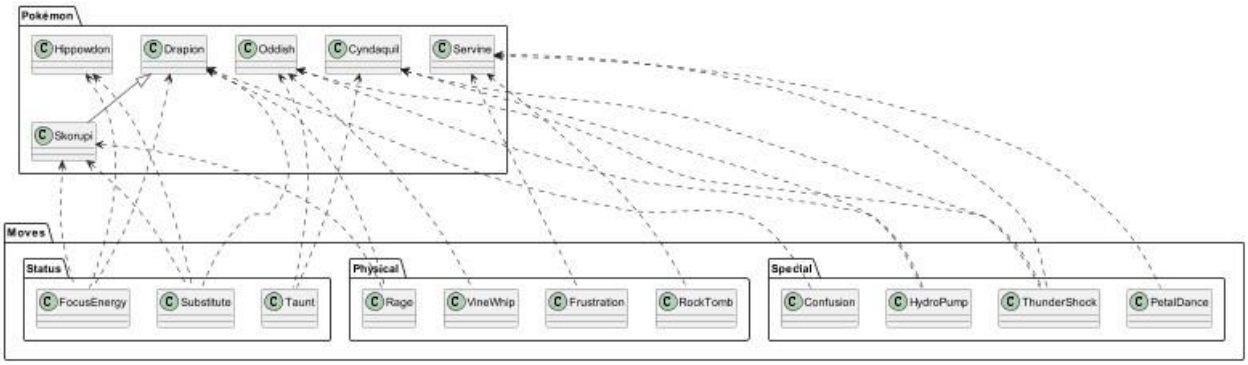


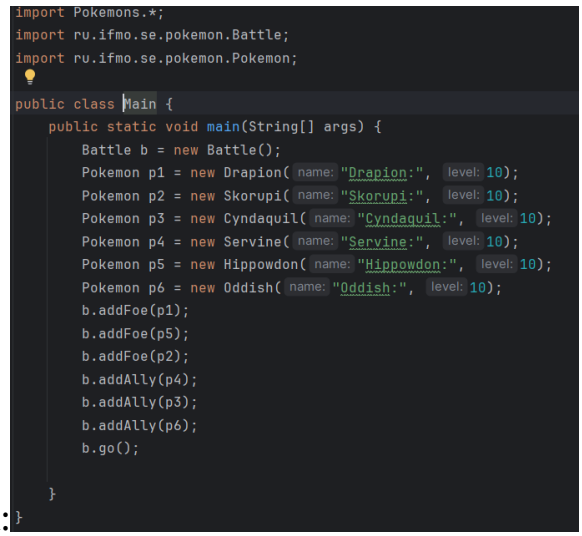
Диаграмма классов





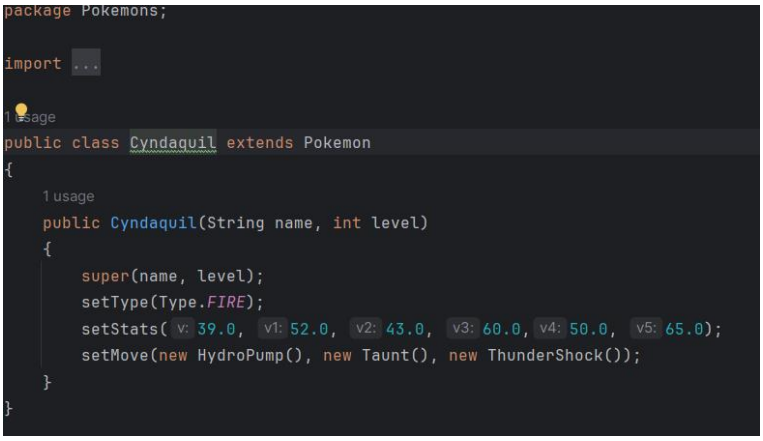
Код программы

1. Main.java:



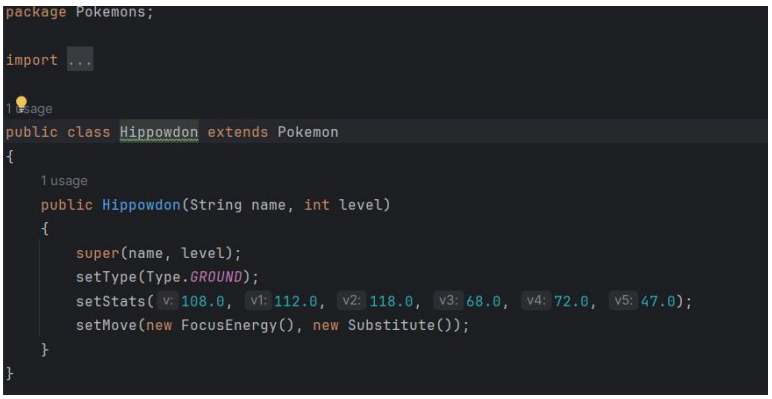
```java
import Pokemons.*;
import ru.ifmo.se.pokemon.Battle;
import ru.ifmo.se.pokemon.Pokemon;

public class Main {
    public static void main(String[] args) {
        Battle b = new Battle();
        Pokemon p1 = new Drapion( name: "Drapion:", level: 10);
        Pokemon p2 = new Skorupi( name: "Skorupi:", level: 10);
        Pokemon p3 = new Cyndaquil( name: "Cyndaquil:", level: 10);
        Pokemon p4 = new Servine( name: "Servine:", level: 10);
        Pokemon p5 = new Hippowdon( name: "Hippowdon:", level: 10);
        Pokemon p6 = new Oddish( name: "Oddish:", level: 10);
        b.addFoe(p1);
        b.addFoe(p5);
        b.addFoe(p2);
        b.addAlly(p4);
        b.addAlly(p3);
        b.addAlly(p6);
        b.go();

    }
}
```

2. Покемоны(сверху вниз: Cyndaquil, Drapion, Hippowdon, Oddish, Servine, Skorupi):

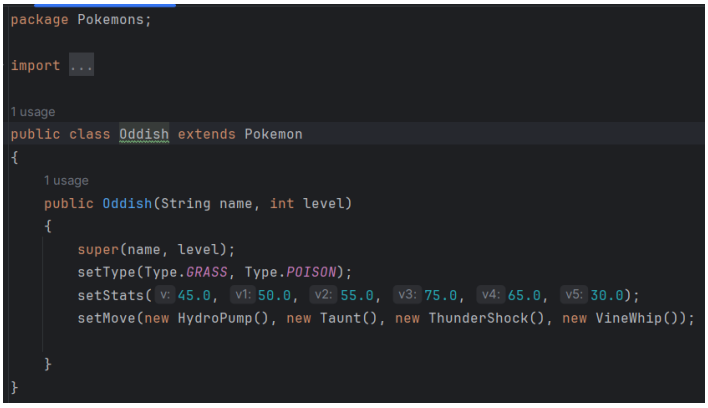


```java
package Pokemons;

import ...

1 usage
public class Cyndaquil extends Pokemon
{
    1 usage
    public Cyndaquil(String name, int level)
    {
        super(name, level);
        setType(Type.FIRE);
        setStats( v: 39.0,  v1: 52.0,  v2: 43.0,  v3: 60.0, v4: 50.0,  v5: 65.0);
        setMove(new HydroPump(), new Taunt(), new ThunderShock());
    }
}
```

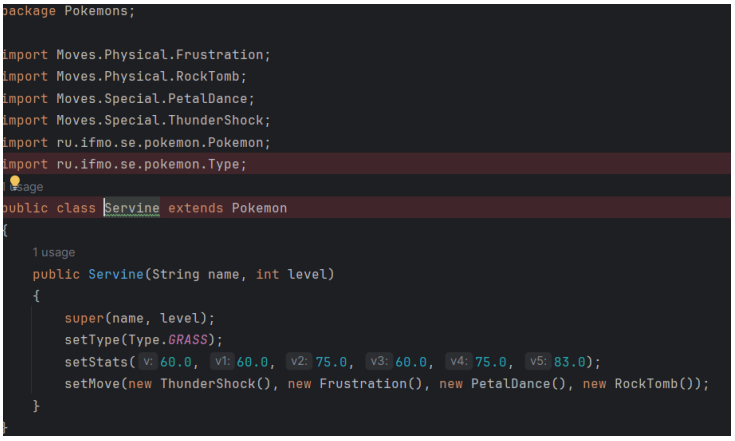
```java
package Pokemons;

import ...

1 usage
public class Drapion extends Skorupi
{

    1 usage
    public Drapion(String name, int level)
    {
        super(name, level);
        setType(Type.POISON, Type.DARK);
        setStats( v: 70.0,  v1: 90.0,  v2: 110.0,  v3: 60.0,  v4: 75.0,  v5: 95.0);
        addMove(new Confusion());
    }
}
```



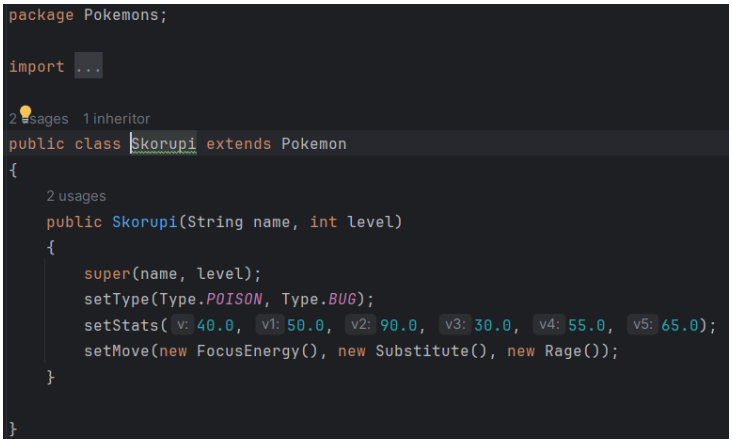
```java
package Pokemons;

import ...

1 usage
public class Hippowdon extends Pokemon
{
    1 usage
    public Hippowdon(String name, int level)
    {
        super(name, level);
        setType(Type.GROUND);
        setStats( v: 108.0,  v1: 112.0,  v2: 118.0,  v3: 68.0,  v4: 72.0,  v5: 47.0);
        setMove(new FocusEnergy(), new Substitute());
    }
}
```



```java
package Pokemons;

import ...

1 usage
public class Oddish extends Pokemon
{
    1 usage
    public Oddish(String name, int level)
    {
        super(name, level);
        setType(Type.GRASS, Type.POISON);
        setStats( v: 45.0,  v1: 50.0,  v2: 55.0,  v3: 75.0,  v4: 65.0,  v5: 30.0);
        setMove(new HydroPump(), new Taunt(), new ThunderShock(), new VineWhip());

    }
}
```



```java
package Pokemons;

import Moves.Physical.Frustration;
import Moves.Physical.RockTomb;
import Moves.Special.PetalDance;
import Moves.Special.ThunderShock;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;
1 usage
public class Servine extends Pokemon
{
    1 usage
    public Servine(String name, int level)
    {
        super(name, level);
        setType(Type.GRASS);
        setStats( v: 60.0,  v1: 60.0,  v2: 75.0,  v3: 60.0,  v4: 75.0,  v5: 83.0);
        setMove(new ThunderShock(), new Frustration(), new PetalDance(), new RockTomb());
    }
}
```



```java
package Pokemons;

import ...

2 usages  1 inheritor
public class Skorupi extends Pokemon
{
    2 usages
    public Skorupi(String name, int level)
    {
        super(name, level);
        setType(Type.POISON, Type.BUG);
        setStats( v: 40.0,  v1: 50.0,  v2: 90.0,  v3: 30.0,  v4: 55.0,  v5: 65.0);
        setMove(new FocusEnergy(), new Substitute(), new Rage());
    }

}
```

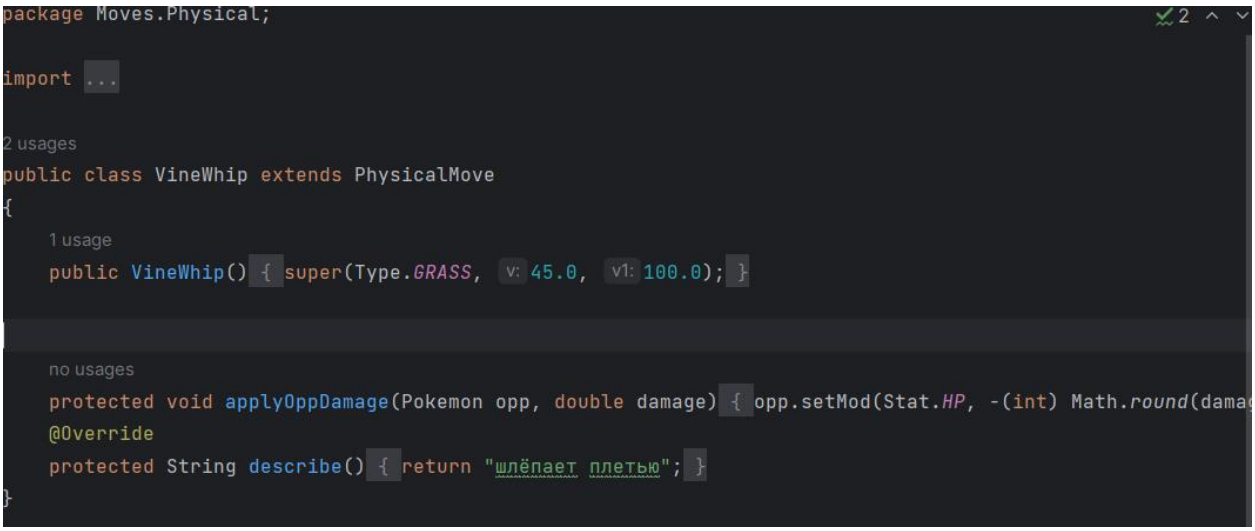
3. Движения:

3.1 Physical(Frustration, Rage, Rock Tomb, Vine Whip):

```java
package Moves.Physical;

import ...

2 usages
public class Frustration extends PhysicalMove
{
    1 usage
    public Frustration() { super(Type.NORMAL, v: 0.0, v1: 100.0); }
    @Override
    protected String describe()
    {
        return "заставляет другого покемона разочароваться в своём хозяине. Снижает показатель дружбы, на

    }
}
```

```java
package Moves.Physical;

import ru.ifmo.se.pokemon.*;

2 usages
public class Rage extends PhysicalMove
{

    1 usage
    public Rage() { super(Type.NORMAL, v: 20, v1: 100); }




    no usages
    @Override
    protected void applySelfEffects(Pokemon p)
    {
        Pokemon opp = new Pokemon();
        if (checkAccuracy(opp, p))
        {
            p.setMod(Stat.ATTACK, i: 1);
        }
    }
    @Override
    protected String describe() { return "взбесился"; }

}
```

```java
package Moves.Physical;

import ...

2 usages
public class RockTomb extends PhysicalMove
{
    1 usage
    public RockTomb() { super(Type.ROCK, v: 60.0, v1: 95.0); }
    no usages
    @Override
    protected void applyOppEffects(Pokemon opp) { opp.setMod(Stat.SPEED, i: -1); }

    no usages
    protected void applyOppDamage(Pokemon opp, double damage) { opp.setMod(Stat.HP, -(int)Math.round(damage

    @Override
    protected String describe() { return "кидает надгробный камень"; }
}
```



```
package Moves.Physical;

import ...

2 usages
public class VineWhip extends PhysicalMove
{
    1 usage
    public VineWhip() { super(Type.GRASS, v: 45.0, v1: 100.0); }

    no usages
    protected void applyOppDamage(Pokemon opp, double damage) { opp.setMod(Stat.HP, -(int) Math.round(dama
    @Override
    protected String describe() { return "шлёпает плетью"; }
}
```

3.2. Special(Confusion, Hydro Pump, Petal Dance, Thunder Shock):

```java
package Moves.Special;

import ...

2 usages
public class Confusion extends SpecialMove
{
    1 usage
    public Confusion() { super(Type.PSYCHIC, v: 50.0, v1: 100.0); }
    no usages
    @Override
    public void applyOppEffects(Pokemon p)
    {
        if (Math.random() <= 0.1)
        {
            Effect.confuse(p);
        }
    }

    protected String describe()
    {
        return "сбивает с толку";

    }
}
```

```java
package Moves.Special;

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

4 usages
public class HydroPump extends SpecialMove
{
    2 usages
    public HydroPump() { super(Type.WATER, v: 110.0, v1: 80.0); }

    no usages
    protected void applyOppDamage(Pokemon opp, double damage) { opp.setMod(Stat.HP, -(int)Math.round(damage
    @Override
    protected String describe() { return "поливает водой"; }
}
```

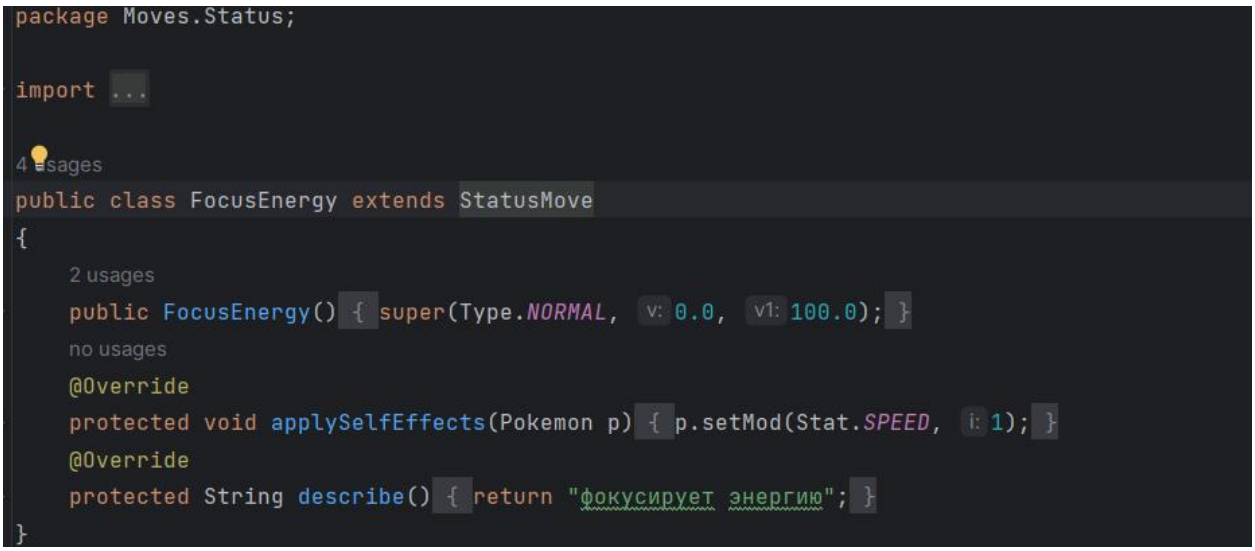
```java
package Moves.Special;

import ru.ifmo.se.pokemon.*;

2 usages
public class PetalDance extends SpecialMove
{
    1 usage
    public PetalDance()
    {
        super(Type.GRASS, v: 120.0, v1: 100.0);
        hits = 3;
    }
    no usages
    @Override
    protected void applyOppEffects(Pokemon opp)
    {
        Effect e = new Effect().chance( v: 1.0).turns( i: 3).condition(Status.FREEZE);
        opp.setCondition(e);
    }
    no usages
    @Override
    protected void applySelfEffects(Pokemon p) { Effect.confuse(p); }
    @Override
    protected String describe() { return "танцует на костях и сбивает себя с толку"; }
}
```

```java
package Moves.Special;

import ...

6 usages
public class ThunderShock extends SpecialMove
{
    3 usages
    public ThunderShock() { super(Type.ELECTRIC, v: 40.0, v1: 100.0); }

    no usages
    @Override
    protected void applyOppEffects(Pokemon p)
    {
        if (Math.random() <= 0.1)
        {
            Effect.paralyze(p);
        }
    }
    @Override
    protected String describe() { return "ударил током"; }
}
```

3.3 Status(Focus Energy, Substitute, Taunt):

```java
package Moves.Status;

import ...

4 usages
public class FocusEnergy extends StatusMove
{
    2 usages
    public FocusEnergy() { super(Type.NORMAL, v: 0.0, v1: 100.0); }
    no usages
    @Override
    protected void applySelfEffects(Pokemon p) { p.setMod(Stat.SPEED, i: 1); }
    @Override
    protected String describe() { return "фокусирует энергию"; }
}
```

```java
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.StatusMove;
import ru.ifmo.se.pokemon.Type;

import static ru.ifmo.se.pokemon.Stat.HP;

4 usages
public class Substitute extends StatusMove
{

    2 usages
    public Substitute() { super(Type.NORMAL, v: 0.0, v1: 100.0); }

    no usages
    @Override
    protected void applySelfEffects(Pokemon p) { p.setMod(HP, (int)p.getStat(HP)); }
    @Override
    protected String describe() { return "кидает приманку"; }
```



```java
package Moves.Status;

import ...

4 usages
public class FocusEnergy extends StatusMove
{
    2 usages
    public FocusEnergy() { super(Type.NORMAL, v: 0.0, v1: 100.0); }
    no usages
    @Override
    protected void applySelfEffects(Pokemon p) { p.setMod(Stat.SPEED, i: 1); }
    @Override
    protected String describe() { return "фокусирует энергию"; }
}
```

Вывод

Выполнив данную лабораторную работу, я освоил азы ООП, усвоил основные принципы ООП(инкапсуляция, полиморфизм, наследование), научился различать модификаторы доступа к классам, узнал об особенностях ООП в Java. Знания и опыт, полученные при выполнении данной лабораторной работы помогут мне в дальнейшем изучении языка Java и других объектно-ориентированных языков программирования(C#, C++). Кроме этого, основы ООП помогут при разработке трудоёмких проектов и написании приложений.