Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет ИТМО"
Факультет Программной инженерии и компьютерной техники

Лабораторная работа №2 по курсу "Программирование"

Группа: P3131
Студент: Друян Эдуард Андреевич
Вариант: 31082

Санкт-Петербург
2021 г.

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.

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.

Отчёт по работе должен содержать:

1. Текст задания.
2. Диаграмма классов реализованной объектной модели.
3. Исходный код программы.
4. Результат работы программы.
5. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Объектно-ориентированное программирование. Основные понятия: объекты, наследование, полиморфизм, инкапсуляция.
2. Понятие класса. Классы и объекты в Java.
3. Члены класса. Модификаторы доступа.
4. Создание и инициализация объектов. Вызов методов.
5. Области видимости переменных.
6. Модификаторы `final` и `static`.
7. Пакеты, инструкция `import`.

Покемоны варианта 31082 (на момент 2021/10/17):

Исходный код программы

```
1.  // Lab2.java
2.  package lib;
3.
4.  import ru.ifmo.se.pokemon.*;
5.  import lib.myMoves.*;
6.  import lib.myPokemons.*;
7.
8.  public class Lab2 {
9.      public static void main(String[] args) {
10.         Battle battle = new Battle();
11.
12.         int lvl = 5;
13.         battle.addAlly(new Pansage("Forelock", lvl));
14.         battle.addAlly(new Gabite("TorpedoEars", lvl));
15.         battle.addAlly(new Manectric("Pony", lvl));
16.
17.         battle.addFoe(new Tropius("NotTRex", lvl));
18.         battle.addFoe(new Snivy("Arrogant", lvl));
19.         battle.addFoe(new Chimecho("Rag", lvl));
20.
21.         battle.go();
22.     }
23. }
```

MyPokemons

```java
// Pansage.java
package lib.myPokemons;

import ru.ifmo.se.pokemon.*;
import lib.myMoves.*;

public class Pansage extends Pokemon {
    public Pansage(String myName, int myLvl) {
        super(myName, myLvl);
        //           hp   att  def  spAtt spDef speed
        this.setStats(50D, 53D, 48D, 53D, 48D, 64D);
        this.setType(Type.GRASS);

        this.addMove(new DynamicPunch());
        this.addMove(new Bulldoze());
        this.addMove(new LeechLife());
        this.addMove(new StoneEdge());
    }
}
```

```java
// Gabite.java
package lib.myPokemons;

import ru.ifmo.se.pokemon.*;
import lib.myMoves.*;

public class Gabite extends Pokemon {
    public Gabite(String myName, int myLvl) {
        super(myName, myLvl);
        //           hp   att  def  spAtt spDef speed
        this.setStats(68D, 90D, 65D, 50D, 55D, 82D);
        this.setType(Type.DRAGON, Type.GROUND);

        this.addMove(new Confusion());
        this.addMove(new ChargeBeam());
        this.addMove(new Swagger());
    }
}
```

```java
// Manectric.java
package lib.myPokemons;

import ru.ifmo.se.pokemon.*;
import lib.myMoves.*;

public class Manectric extends Pokemon {
    public Manectric(String myName, int myLvl) {
        super(myName, myLvl);
        //           hp   att  def  spAtt spDef speed
        this.setStats(70D, 75D, 60D, 105D, 60D, 105D);
        this.setType(Type.ELECTRIC);

        this.addMove(new Confusion());
        this.addMove(new ChargeBeam());
        this.addMove(new Swagger());
        this.addMove(new ShadowBall());
    }
}
```

```java
// Tropius.java
package lib.myPokemons;

import ru.ifmo.se.pokemon.*;
import lib.myMoves.*;

public class Tropius extends Pokemon {
    public Tropius(String myName, int myLvl) {
        super(myName, myLvl);
        //           hp   att  def  spAtt spDef speed
        this.setStats(99D, 68D, 83D, 72D, 87D, 51D);
        this.setType(Type.GRASS, Type.FLYING);

        this.addMove(new DoubleTeam());
        this.addMove(new TeeterDance());
    }
}
```

```java
// Snivy.java
package lib.myPokemons;

import ru.ifmo.se.pokemon.*;
import lib.myMoves.*;

public class Snivy extends Pokemon {
    public Snivy(String myName, int myLvl) {
        super(myName, myLvl);
        //           hp   att  def  spAtt spDef speed
        this.setStats(45D, 45D, 55D, 45D, 55D, 63D);
        this.setType(Type.GRASS);

        this.addMove(new DoubleTeam());
        this.addMove(new TeeterDance());
        this.addMove(new PlayNice());
    }
}
```

```java
// Chimecho.java
package lib.myPokemons;

import ru.ifmo.se.pokemon.*;
import lib.myMoves.*;

public class Chimecho extends Pokemon {
    public Chimecho(String myName, int myLvl) {
        super(myName, myLvl);
        //           hp   att  def  spAtt spDef speed
        this.setStats(75D, 50D, 80D, 95D, 90D, 65D);
        this.setType(Type.PSYCHIC);

        this.addMove(new DoubleTeam());
        this.addMove(new TeeterDance());
        this.addMove(new PlayNice());
        this.addMove(new SweetScent());
    }
}
```

MyMoves

```java
// Bulldoze.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class Bulldoze extends PhysicalMove {
    public Bulldoze() {
        super(Type.GROUND, 60D, 100D);
    }

    @Override
    protected String describe() {
        return("used PhysicalMove \"Bulldoze\"; defender's speed is lowered");
    }

    // Bulldoze deals damage and lowers the target's Speed by one stage.

    @Override
    protected void applyOppEffects(Pokemon def) {
        def.setMod(Stat.SPEED, -1);
    }
}
```

```java
// StoneEdge.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

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

    @Override
    protected String describe() {
        if (this.luckyChance) {
            return("used PhysicalMove \"StoneEdge\"; critical hit applied");
        }
        else { return("used PhysicalMove \"StoneEdge\""); }
    }

    // Stone Edge deals damage
    // and has an increased critical hit ratio (1/8 instead of 1/24).
    // Means x3 increased critical ratio.

    private boolean luckyChance = false;

    @Override
    protected double calcCriticalHit(Pokemon att, Pokemon def) {
        double critHitRatio = 3D * att.getStat(Stat.SPEED) / 512D;
        if (critHitRatio > Math.random()) {
            this.luckyChance = true;
            return 2D;
        }
        else { return 1D; }
    }
}
```

```java
// LeechLife.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class LeechLife extends PhysicalMove {
    public LeechLife() {
        super(Type.BUG, 80D, 100D);
    }

    @Override
    protected String describe() {
        return("used PhysicalMove \"LeechLife\"; attacker recovered HP");
    }

    // Leech Life deals damage and the user will recover 50% of the HP drained.

    @Override
    protected void applySelfDamage(Pokemon att, double damage) {
        att.setMod(Stat.HP, (int) -Math.round(damage * 0.5));
    }
}
```

```java
// DynamicPunch.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class DynamicPunch extends PhysicalMove {
    public DynamicPunch() {
        super(Type.FIGHTING, 100D, 50D);
    }

    @Override
    protected String describe() {
        return("used PhysicalMove \"DynamicPunch\"; defender is confused");
    }

    // Dynamic Punch deals damage and confuses the target, if it hits
    // (no verification required because it has already been done).

    @Override
    protected void applyOppEffects(Pokemon def) {
        Effect.confuse(def);
        // def.confuse();
    }
}
```

```java
// ShadowBall.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class ShadowBall extends SpecialMove {
    public ShadowBall() {
        super(Type.GHOST, 80D, 100D);
    }

    @Override
    protected String describe() {
        if (this.luckyChance) {
            return("used SpecialMove \"ShadowBall\";\n" +
                    "\tDefender's Special Defence is lowered");
        }
        else { return("used SpecialMove \"ShadowBall\""); }
    }

    // Shadow Ball deals damage and has a 20% chance
    // of lowering the target's Special Defense by one stage.

    private boolean luckyChance = false;

    @Override
    protected void applyOppEffects(Pokemon def) {
        if (0.2 > Math.random()) {
            this.luckyChance = true;
            def.setMod(Stat.SPECIAL_DEFENSE, -1);
        }
    }
}
```

```java
// ChargeBeam.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class ChargeBeam extends SpecialMove {
    public ChargeBeam() {
        super(Type.ELECTRIC, 50D, 90D);
    }

    @Override
    protected String describe() {
        if (this.luckyChance) {
            return("used SpecialMove \"ChargeBeam\";\n" +
                    "\tAtteker's Special Attack is raised");
        }
        else { return("used SpecialMove \"ChargeBeam\""); }
    }

    // Charge Beam deals damage and has a 70% chance of
    // raising the user's Special Attack by one stage.

    private boolean luckyChance = false;

    @Override
    protected void applySelfEffects(Pokemon att) {
        if (0.7 > Math.random()) {
            this.luckyChance = true;
            att.setMod(Stat.SPECIAL_ATTACK, +1);
        }
    }
}
```

```java
// Confusion.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class Confusion extends SpecialMove {
    public Confusion() {
        super(Type.PSYCHIC, 50D, 100D);
    }

    @Override
    protected String describe() {
        if (this.luckyChance) {
            return("used SpecialMove \"Confusion\"; defender is confused");
        }
        return("used SpecialMove \"Confusion\"");
    }

    // Confusion deals damage and has a 10% chance of confusing the target.

    private boolean luckyChance = false;

    @Override
    protected void applyOppEffects(Pokemon def) {
        if (0.1 > Math.random()) {
            this.luckyChance = true;
            Effect.confuse(def);
            // def.confuse();
        }
    }
}
```

```java
// DoubleTeam.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class DoubleTeam extends StatusMove {
    public DoubleTeam() {
        super(Type.NORMAL, 0D, 0D);
    }

    @Override
    protected String describe() {
        return("used StatusMove \"DoubleTeam\"; attaker's Evasion is raised");
    }

    // Double Team raises the user's Evasiveness by one stage,
    // thus making the user more difficult to hit.

    @Override
    protected void applySelfEffects(Pokemon att) {
        att.setMod(Stat.EVASION, +1);
    }
}

```

```java
// PlayNice.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class PlayNice extends StatusMove {
    public PlayNice() {
        super(Type.NORMAL, 0D, 0D);
    }

    @Override
    protected String describe() {
        return("used StatusMove \"PlayNice\"; defender's Attack is lowered");
    }

    // Play Nice lowers the target's Attack by one stage.

    @Override
    protected void applyOppEffects(Pokemon def) {
        def.setMod(Stat.ATTACK, -1);
    }
}
```

```java
// Swagger.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

public class Swagger extends StatusMove {
    public Swagger() {
        super(Type.NORMAL, 0D, 85D);
    }

    @Override
    protected String describe() {
        return("used StatusMove \"Swagger\";\n" +
                "\tDefender is confused and his Attack is raised");
    }

    // Swagger confuses the target and raises its Attack by two stages.

    @Override
    protected void applyOppEffects(Pokemon def) {
        Effect.confuse(def);
        def.setMod(Stat.ATTACK, +2);
    }
}
```

```java
// SweetScent.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

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

    @Override
    protected String describe() {
        return("used StatusMove \"SweetScent\"; defender's Evasion is lowered");
    }

    // Sweet Scent lowers the target's Evasion by one stage.

    @Override
    protected void applyOppEffects(Pokemon def) {
        def.setMod(Stat.EVASION, -1);
    }
}
```

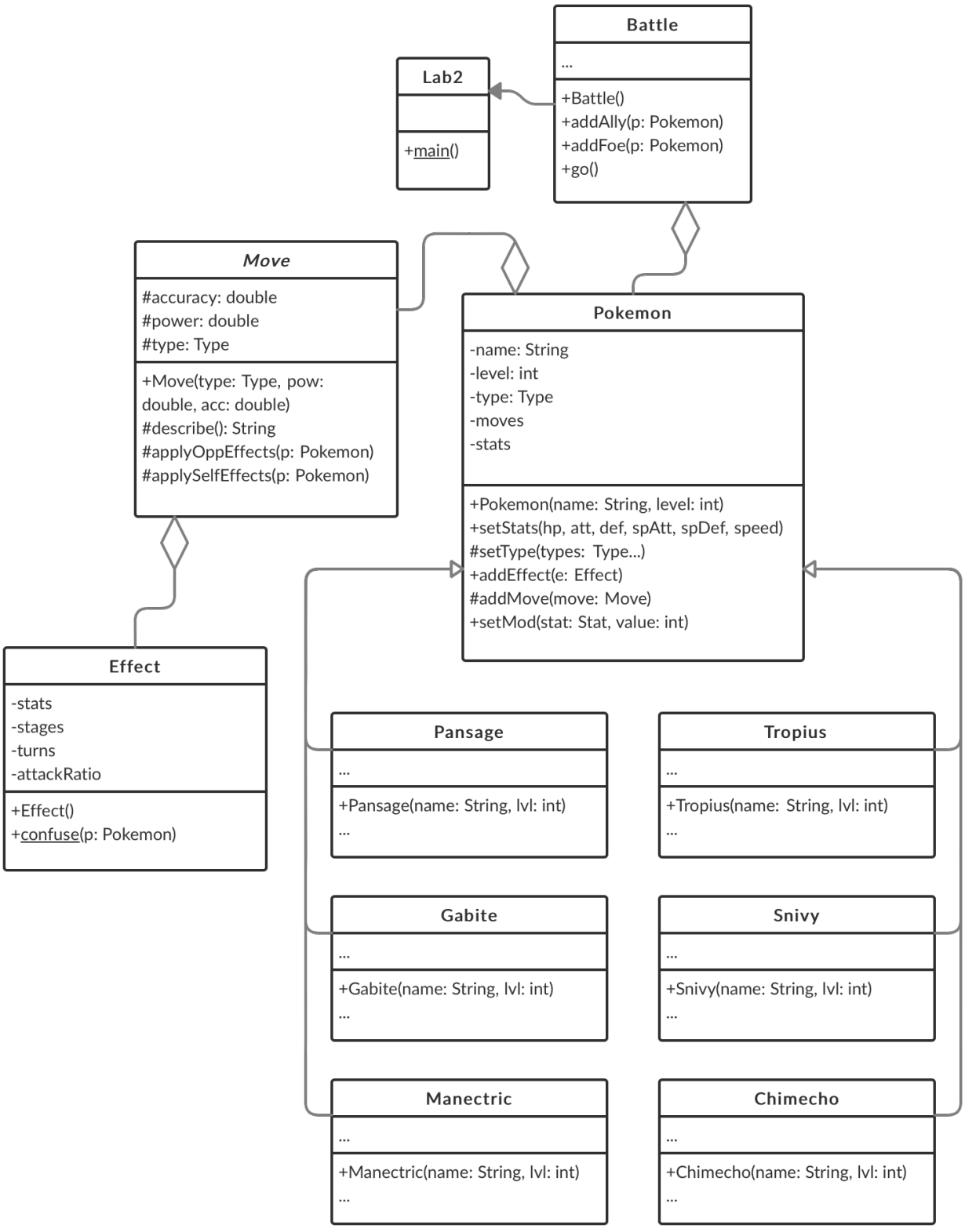
```java
// TeeterDance.java
package lib.myMoves;

import ru.ifmo.se.pokemon.*;

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

    @Override
    protected String describe() {
        return("used StatusMove \"TeeterDance\"; defender is confused");
    }

    // Teeter Dance causes opponent to become confused.

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

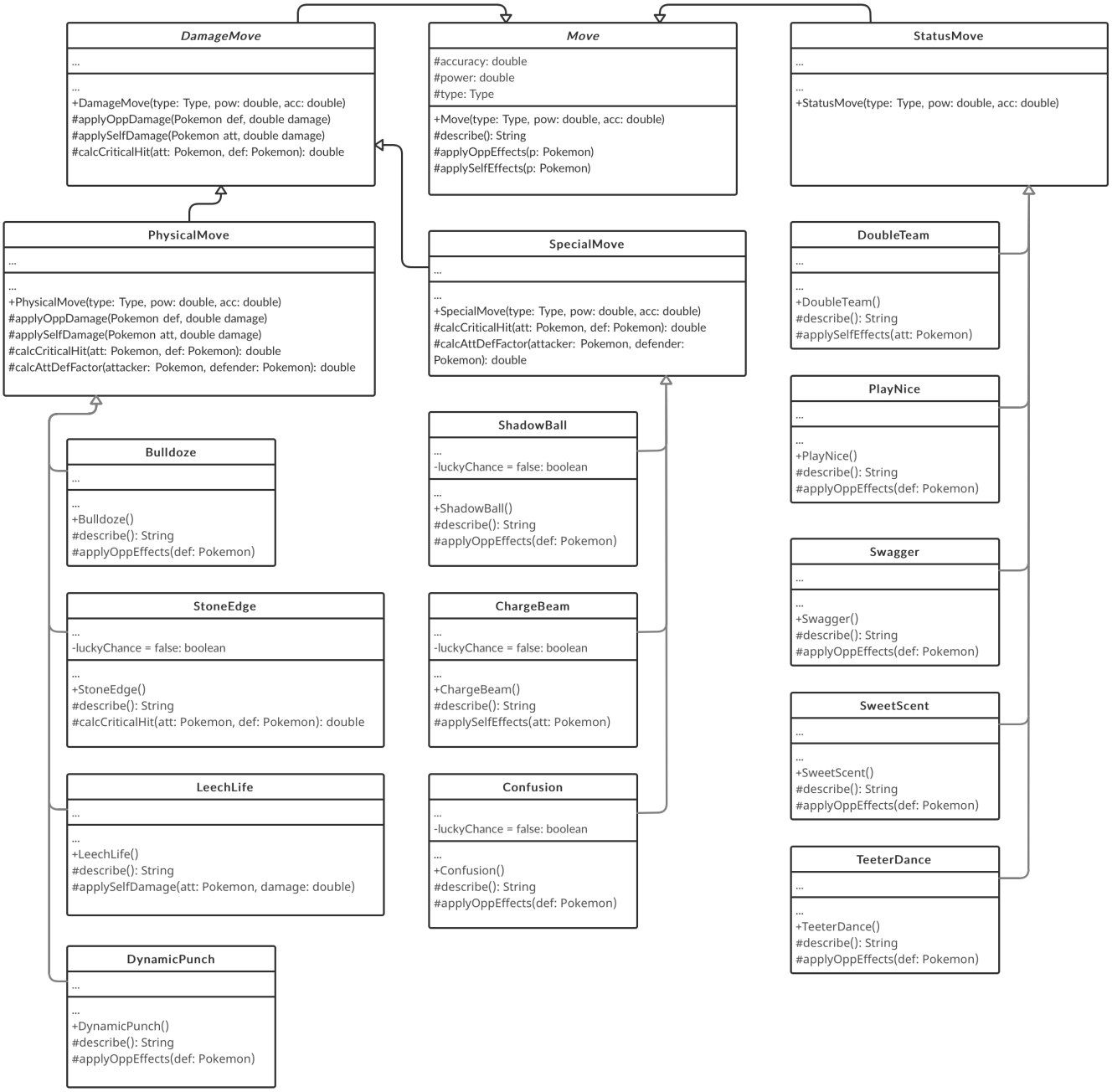
Результаты работы программы

```
Pansage Forelock from the team Red enters the battle!
Tropius NotTRex from the team Purple enters the battle!
Pansage Forelock used PhysicalMove "Bulldoze"; defender's speed is lowered.
Tropius NotTRex loses 1 hit points.
Tropius NotTRex isn't affected by GROUND

Tropius NotTRex used StatusMove "TeeterDance"; defender is confused.

Pansage Forelock used PhysicalMove "DynamicPunch"; defender is confused.
Tropius NotTRex loses 3 hit points.

Tropius NotTRex misses

Pansage Forelock hits himself in confusion.
Pansage Forelock loses 4 hit points.

Tropius NotTRex struggles.
Pansage Forelock loses 6 hit points.
Tropius NotTRex loses 2 hit points.

Pansage Forelock hits himself in confusion.
Pansage Forelock loses 5 hit points.

Tropius NotTRex misses

Pansage Forelock used PhysicalMove "LeechLife"; attacker recovered HP.
Tropius NotTRex loses 7 hit points.
Pansage Forelock restores 4 hit points.

Tropius NotTRex hits himself in confusion.
Tropius NotTRex loses 4 hit points.

Pansage Forelock used PhysicalMove "Bulldoze"; defender's speed is lowered.
Tropius NotTRex loses 1 hit points.
Tropius NotTRex isn't affected by GROUND

Tropius NotTRex used StatusMove "TeeterDance"; defender is confused.

Pansage Forelock used PhysicalMove "Bulldoze"; defender's speed is lowered.
Tropius NotTRex loses 1 hit points.
Tropius NotTRex isn't affected by GROUND

Tropius NotTRex struggles.
Pansage Forelock loses 6 hit points.
Tropius NotTRex loses 2 hit points.

Pansage Forelock hits himself in confusion.
Pansage Forelock loses 6 hit points.
Pansage Forelock faints.
Gabite TorpedoEars from the team Red enters the battle!
Gabite TorpedoEars used StatusMove "Swagger";
     Defender is confused and his Attack is raised.
Tropius NotTRex increases attack.

Tropius NotTRex misses

Gabite TorpedoEars used StatusMove "Swagger";
     Defender is confused and his Attack is raised.
Tropius NotTRex increases attack.

Tropius NotTRex used StatusMove "TeeterDance"; defender is confused.

Gabite TorpedoEars used SpecialMove "ChargeBeam".
Tropius NotTRex loses 4 hit points.

Tropius NotTRex used StatusMove "TeeterDance"; defender is confused.

Gabite TorpedoEars used StatusMove "Swagger";
     Defender is confused and his Attack is raised.
Tropius NotTRex increases attack.
```

```
Tropius NotTRex hits himself in confusion.
Tropius NotTRex loses 5 hit points.
Tropius NotTRex faints.
Snivy Arrogant from the team Purple enters the battle!
Gabite TorpedoEars used SpecialMove "Confusion".
Critical hit!
Snivy Arrogant loses 9 hit points.

Snivy Arrogant used StatusMove "TeeterDance"; defender is confused.

Gabite TorpedoEars used SpecialMove "ChargeBeam".
Snivy Arrogant loses 2 hit points.

Snivy Arrogant struggles.
Gabite TorpedoEars loses 4 hit points.
Snivy Arrogant loses 1 hit points.

Gabite TorpedoEars used StatusMove "Swagger";
     Defender is confused and his Attack is raised.
Snivy Arrogant increases attack.

Snivy Arrogant misses

Gabite TorpedoEars struggles.
Snivy Arrogant loses 7 hit points.
Gabite TorpedoEars loses 2 hit points.

Snivy Arrogant hits himself in confusion.
Snivy Arrogant loses 5 hit points.
Snivy Arrogant faints.
Chimecho Rag from the team Purple enters the battle!
Gabite TorpedoEars struggles.
Chimecho Rag loses 4 hit points.
Gabite TorpedoEars loses 1 hit points.

Chimecho Rag used StatusMove "SweetScent"; defender's Evasion is lowered.
Gabite TorpedoEars decreases evasion.

Gabite TorpedoEars used SpecialMove "ChargeBeam".
Critical hit!
Chimecho Rag loses 7 hit points.

Chimecho Rag used StatusMove "TeeterDance"; defender is confused.

Gabite TorpedoEars hits himself in confusion.
Gabite TorpedoEars loses 3 hit points.

Chimecho Rag misses

Gabite TorpedoEars hits himself in confusion.
Gabite TorpedoEars loses 5 hit points.

Chimecho Rag used StatusMove "SweetScent"; defender's Evasion is lowered.
Gabite TorpedoEars decreases evasion.

Gabite TorpedoEars used SpecialMove "ChargeBeam".
Chimecho Rag loses 4 hit points.

Chimecho Rag misses

Gabite TorpedoEars used SpecialMove "ChargeBeam".
Chimecho Rag loses 3 hit points.

Chimecho Rag misses

Gabite TorpedoEars struggles.
Chimecho Rag loses 6 hit points.
Gabite TorpedoEars loses 2 hit points.
Chimecho Rag faints.
Team Purple loses its last Pokemon.
The team Red wins the battle!
```

Диаграммы классов





Выводы

В "лабе" работал с документацией, пакетами, UML, компиляцией нескольких файлов и упаковкой нескольких классов в jar-файл, что, должно быть, весьма актуально.

На простом примере разобрался с основными концепциями ООП и научился использовать их в программах.