

Санкт-Петербургский Национальный Исследовательский
Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

По дисциплине

“Программирование”

Вариант 3117210

Выполнил:

Студент группы Р3117

Пономарёв М. И.

Преподаватель:

Письмак А. Е.



Оглавление

Текст задания	3
Основные этапы вычисления.....	5
Список источников	13
Вывод.....	13

Текст задания

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.

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:

Bouffalant  Атаки: <ul style="list-style-type: none">✓ Head Charge✓ Swagger✓ Horn Attack✓ Facade	Tirtouga  Атаки: <ul style="list-style-type: none">✓ Aqua Tail✓ Rest✓ Rock Tomb	Carracosta  Атаки: <ul style="list-style-type: none">✓ Aqua Tail✓ Rest✓ Rock Tomb✓ Focus Blast	Trapinch  Атаки: <ul style="list-style-type: none">✓ Swagger✓ Mud-Slap
Vibrava  Атаки: <ul style="list-style-type: none">✓ Swagger✓ Mud-Slap✓ Bug Buzz	Flygon  Атаки: <ul style="list-style-type: none">✓ Swagger✓ Mud-Slap✓ Bug Buzz✓ Feint Attack		

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`.

Основные этапы вычисления

UML Диаграмма

[Link to Image](#)

Исходный код программы

Attacks

```
public class AquaTail extends PhysicalMove {  
    public AquaTail() {  
        super(Type.WATER, 90, 90);  
    }  
}
```

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

```
public class BugBuzz extends SpecialMove {  
    public BugBuzz() {  
        super(Type.BUG, 90, 100);  
    }  
    private boolean isChance = false;  
    @Override  
    protected void applyOppEffects(Pokemon p) {  
        if(Math.random() < 0.1)  
        {  
            p.setMod(Stat.SPECIAL_DEFENSE, -1);  
            isChance = true;  
        }  
    }  
    @Override  
    protected String describe() {  
        return "Д.Ñ□Д¿Д³Д»ÑCED·ÑfДµÑ, BugBuzz " +((isChance ? "and lowered  
opponent SPECIAL DEFENSE by -1" : ""));  
    }  
}
```

```
public class Facade extends PhysicalMove {  
    public Facade() {  
        super(Type.NORMAL, 70, 100);  
    }  
}
```



```

        super(Type.NORMAL, 120, 100);
    }

    @Override
    protected void applySelfDamage(Pokemon att, double damage) {
        att.setMod(Stat.HP, (int) Math.round(damage / 4));
    }

    @Override
    protected String describe() {
        return "использует HeadCharge ";
    }
}

public class HornAttack extends PhysicalMove {
    public HornAttack() {
        super(Type.NORMAL, 65, 100);
    }

    @Override
    protected String describe() {
        return "использует HornAttack ";
    }
}

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

    @Override
    protected void applyOppEffects(Pokemon p) {
        p.setMod(Stat.ACCURACY, -1);
    }

    @Override
    protected String describe() {
        return "использует MudSlap ";
    }
}

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

    @Override
    protected void applySelfEffects(Pokemon p) {
        Effect e = new Effect();
        e = e.condition(Status.SLEEP);
        e = e.turns(2);
    }
}

```

```

        p.restore();
        p.addEffect(e);
    }

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

public class RockTomb extends PhysicalMove {
    public RockTomb() {
        super(Type.ROCK, 60, 95);
    }

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

    @Override
    protected String describe() {
        return "использует RockTomb ";
    }
}

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

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

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

```


Pokemons

```
public class Bouffalant extends Pokemon {
    public Bouffalant(String name, int level) {
        super(name, level);
        setStats(95, 110, 95, 40, 95, 55);
        setType(Type.NORMAL);
        setMove(new HeadCharge(), new Swagger(), new HornAttack(), new
Facade());
    }
}
```

```
public class Carracosta extends Pokemon {
    public Carracosta(String name, int level) {
        super(name, level);
        setStats(74, 108, 133, 83, 65, 32);
        setType(Type.WATER, Type.ROCK);
        setMove(new AquaTail(), new Rest(), new RockTomb(), new
FocusBlast());
    }
}
```

```
public class Flygon extends Pokemon {
    public Flygon(String name, int level) {
        super(name, level);
        setStats(80, 100, 80, 80, 80, 100);
        setType(Type.GROUND, Type.DRAGON);
        setMove(new Swagger(), new MudSlap(), new BugBuzz(), new
FeintAttack());
    }
}
```

```
public class Tirtouga extends Pokemon {
    public Tirtouga(String name, int level) {
        super(name, level);
        setStats(54, 78, 103, 53, 45, 22);
        setType(Type.WATER, Type.ROCK);
        setMove(new AquaTail(), new Rest(), new RockTomb());
    }
}
```

```

public class Trapinch extends Pokemon {
    public Trapinch(String name, int level) {
        super(name, level);
        setStats(45, 100, 45, 45, 45, 10);
        setType(Type.GROUND);
        setMove(new Swagger(), new MudSlap());
    }
}

public class Vibrava extends Pokemon {
    public Vibrava(String name, int level) {
        super(name, level);
        setStats(50, 70, 50, 50, 50, 70);
        setType(Type.GROUND, Type.DRAGON);
        setMove(new Swagger(), new MudSlap(), new BugBuzz());
    }
}

```

Lab2

```

package ponomaryov.mikhail;

import ru.ifmo.se.pokemon.*;
import ponomaryov.mikhail.attacks.*;
import ponomaryov.mikhail.pokemons.*;

public class Lab2 {
    public static void main(String[] args) {
        Battle b = new Battle();
        Bouffalant p1 = new Bouffalant("Бычок", 1);
        Tirtouga p2 = new Tirtouga("Черепашка", 1);
        Carracosta p3 = new Carracosta("Черепашка мутант", 1);
        Trapinch p4 = new Trapinch("Большеголовая черепашка", 3);
        Vibrava p5 = new Vibrava("Стрекоза", 3);
        Flygon p6 = new Flygon("Мини дракон", 3);

        b.addAlly(p1);
        b.addAlly(p2);
        b.addAlly(p3);

        b.addFoe(p4);
        b.addFoe(p5);
        b.addFoe(p6);
        b.go();
    }
}

```

Вывод программы

Bouffalant Бычок из команды фиолетовых вступает в бой!

Trapinch Большеголовая черепашка из команды синих вступает в бой!

Bouffalant Бычок использует HeadCharge .

Trapinch Большеголовая черепашка теряет 10 здоровья.

Bouffalant Бычок теряет 3 здоровья.

Trapinch Большеголовая черепашка использует Swagger .

Bouffalant Бычок увеличивает атаку.

Bouffalant Бычок использует Swagger .

Trapinch Большеголовая черепашка увеличивает атаку.

Trapinch Большеголовая черепашка использует MudSlap .

Bouffalant Бычок теряет 5 здоровья.

Bouffalant Бычок уменьшает точность.

Bouffalant Бычок использует HeadCharge .

Trapinch Большеголовая черепашка теряет 8 здоровья.

Bouffalant Бычок теряет 2 здоровья.

Trapinch Большеголовая черепашка теряет сознание.

Vibrava Стрекоза из команды синих вступает в бой!

Vibrava Стрекоза использует Swagger .

Bouffalant Бычок увеличивает атаку.

Bouffalant Бычок использует Facade .

Vibrava Стрекоза теряет 6 здоровья.

Vibrava Стрекоза использует Swagger .

Bouffalant Бычок увеличивает атаку.

Bouffalant Бычок использует Swagger .

Vibrava Стрекоза увеличивает атаку.

Vibrava Стрекоза использует BugBuzz .

Bouffalant Бычок теряет 5 здоровья.

Bouffalant Бычок теряет сознание.

Tirtouga Черепашка из команды фиолетовых вступает в бой!

Vibrava Стрекоза использует BugBuzz .

Tirtouga Черепашка теряет 9 здоровья.

Tirtouga Черепашка использует RockTomb .

Критический удар!

Vibrava Стрекоза теряет 6 здоровья.

Vibrava Стрекоза уменьшает скорость.

Vibrava Стрекоза использует BugBuzz .

Tirtouga Черепашка теряет 8 здоровья.

Tirtouga Черепашка теряет сознание.

Carracosta Черепашка мутант из команды фиолетовых вступает в бой!

Vibrava Стрекоза использует MudSlap .

Carracosta Черепашка мутант теряет 13 здоровья.

Carracosta Черепашка мутант уменьшает точность.

Carracosta Черепашка мутант теряет сознание.

В команде фиолетовых не осталось покемонов.

Команда синих побеждает в этом бою!

Список источников

- 1) <https://se.ifmo.ru/~tony/doc/>
- 2) <http://pokemondb.net>

Вывод

Во время лабораторной работы я использовал jar пакет для создания своих классов с помощью наследования, а также применил принципы ООП для того, чтобы переопределять методы и использовать их в своём java файле.