Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики.

Лабораторная работа №2

По дисциплине «Программирование»

Вариант № 32530

Выполнил:

Русинов Дмитрий Станиславович P3132

Преподаватель:

Горбунов Михаил Витальевич

Санкт-Петербург

2021

Оглавление

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.

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 и их атаки варианта:

Bouffalant:

- Stone Edge
- Leer
- Rock Tomb
- Facade

Cosmoem

- Swagger
- Double team

Munchlax

- Blizzard
- Body Slam
- Double Team

Nigilego

- Swagger
- Double team
- Aerial Ace
- Charge Beam

Snorlax

- Pound
- Blizzard

Body Slam

Double Team

Solgaleo

Aerial Ace

Swagger

Double team

Исходный код:

Main

```
import ru.ifmo.se.pokemon.*;
import Pokemons.*;

public class Main {

    public static void main(String[] args) {

        Battle b = new Battle();

        Munchlax p1 = new Munchlax("1", 22);
        Bouffalant p2 = new Bouffalant("2", 30);
        Snorlax p3 = new Snorlax("3", 38);
        Cosmoem p4 = new Cosmoem("4", 43);
        Solgaleo p5 = new Solgaleo("5", 53);
        Nigilego p6 = new Nigilego("6", 61);

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

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

        b.go();
    }

}
```

Bouffalant

```
package Pokemons;

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

public class Bouffalant extends Pokemon {
    public Bouffalant(String name, int level) {
        super(name, level);
        setStats(95, 110, 95, 30, 95, 55);
        setType(Type.NORMAL);

        addMove(new Stone_Edge());
        addMove(new Leer());
        addMove(new Rock_Tomb());
        addMove(new Facade());
    }
}
```

Cosmoem

```
package Pokemons;

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

public class Cosmoem extends Pokemon{
    public Cosmoem (String name, int level) {
        super(name, level);
        setStats(43, 29, 131, 29, 131, 37);
        setType(Type.PSYCHIC);
        addMove(new Swagger());
        addMove(new Double_Team());
    }

}
```

Munchlax

```
package Pokemons;

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

public class Munchlax extends Pokemon{
    public Munchlax(String name, int level) {
        super(name, level);
        setStats(135, 85, 40, 40, 85, 5);
        setType(Type.NORMAL);

        addMove(new Blizzard());
        addMove(new Body_Slam());
        addMove(new Double_Team());
    }

}
```

Nigilego

```
package Pokemons;

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

public class Nigilego extends Pokemon {
    public Nigilego(String name, int level) {
        super(name, level);
        setStats(109, 53, 47, 127, 131, 103);
        setType(Type.ROCK, Type.POISON);

        addMove(new Swagger());
        addMove(new Double_Team());
        addMove((new Aerial_Ace()));
        addMove((new Charge_Beam()));;
    }
}
```

Snorlax

```
package Pokemons;

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

public class Snorlax extends Munchlax{
    public Snorlax(String name, int level) {
        super(name, level);
        setStats(160, 110, 65, 65, 110, 30);
        setType(Type.NORMAL);

        addMove(new Pound());
    }
}
```

Solgaleo

```
package Pokemons;

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

public class Solgaleo extends Cosmoem {
    public Solgaleo(String name, int level) {
        super(name, level);
        setStats(137, 137, 107, 113, 89, 97);
        setType(Type.PSYCHIC, Type.STEEL);

        addMove(new Aerial_Ace());
    }
}
```

Aerial_Ace

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Aerial_Ace extends PhysicalMove {
    public Aerial_Ace() { super(Type.FLYING, 60, 0); } // has 100% chance to
hit

    @Override
    protected boolean checkAccuracy(Pokemon pokemon, Pokemon pokemon1) {
        return true;
    }

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

Blizzard

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Blizzard extends SpecialMove {

    public Blizzard() { super (Type.ICE, 110, 70); }

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

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

Body_Slam

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Body_Slam extends PhysicalMove {

    public Body_Slam() { super(Type.NORMAL, 85, 100); }

    protected void applyOppEffects(Pokemon p) { // 30% chance of paralyzing
the target
        if (Math.random() < 0.3) {
            Effect.paralyze(p);
        }
    }

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

Charge_Beam

```
package Moves;

import ru.ifmo.se.pokemon.*;

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

    @Override
    protected void applySelfEffects(Pokemon p) { // has 70% chance of raising
the user's attack by one stage
        if (Math.random() <= 0.7) {
            p.setMod(Stat.ATTACK, 1);
        }
    }

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

Double_Team

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Double_Team extends StatusMove{

    public Double_Team() { super(Type.NORMAL, 0, 0); }

    @Override
    protected void applySelfEffects(Pokemon p) { // raises user's Evasiveness
by one stage
        p.setMod(Stat.EVASION, 1);
    }

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

Facade

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Facade extends PhysicalMove {

    public Facade() {
        super(Type.NORMAL, 70, 100);
    }

    @Override
    protected double calcBaseDamage(Pokemon var1, Pokemon var2) { // if the
user is burned, poisoned or paralyzed deal damage with double power
        Status c = var1.getCondition();
        int d = 1; // power multiplier
        if (c.equals(Status.BURN) || c.equals(Status.POISON) ||
c.equals(Status.PARALYZE)) {
            d = 2;
        }
        return (0.4D * (double) var1.getLevel() + 2.0D) * this.power * d /
150.0D;
    }

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

Leer

```
package Moves;
import ru.ifmo.se.pokemon.*;

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

    @Override
    protected void applyOppEffects(Pokemon p) { // lowers the target's
defence by 1 stage
        p.setMod(Stat.DEFENSE, -1);
    }

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

Pound

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Pound extends PhysicalMove {
    public Pound() {
        super (Type.NORMAL, 40, 100);
    }

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

Rock_Tomb

```
package Moves;

import ru.ifmo.se.pokemon.*;

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

    @Override
    protected void applyOppEffects(Pokemon p) { // lower target's speed by 1
stage
        p.setMod(Stat.SPEED, -1);
    }

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

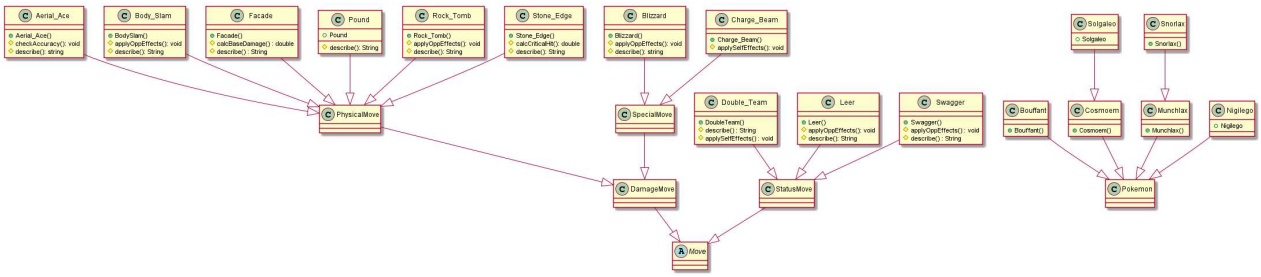
Stone_Edge

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Blizzard extends SpecialMove {

    public Blizzard() { super (Type.ICE, 110, 70); }

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

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

Swagger

```
package Moves;

import ru.ifmo.se.pokemon.*;

public class Swagger extends StatusMove {

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

    @Override
    protected void applyOppEffects (Pokemon p) { // confuses the target and
raises theirs attack by 2 stages
        p.setMod(Stat.ATTACK, 2);
        Effect.confuse(p);
    }

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

UML


Aerial_Ace
Aerial_Ace()
checkAccuracy(): void
describe(): string
Body_Slam
BodySlam()
applyOppEffects(): void
describe(): String
Facade
Facade()
calcBaseDamage(): double
describe(): String
Pound
Pound
describe(): String
Rock_Tomb
Rock_Tomb()
applyOppEffects(): void
describe(): String
Stone_Edge
Stone_Edge()
calcCriticalHit(): double
describe(): String
Blizzard
Blizzard()
applyOppEffects(): void
describe(): string
Charge_Beam
Charge_Beam()
applySelfEffects(): void
Solgaleo
Solgaleo
Snorlax
Snorlax()
PhysicalMove
SpecialMove
Double_Team
DoubleTeam()
describe(): String
applySelfEffects(): void
Leer
Leer()
applyOppEffects(): void
describe(): String
Swagger
Swagger()
applyOppEffects(): void
describe(): String
Bouffant
Bouffant()
Cosmoem
Cosmoem()
Munchlax
Munchlax()
Nigilego
Nigilego
DamageMove
StatusMove
Pokemon
Move

Пример выходных данных программы:

Munchlax 1 из команды синих вступает в бой!

Cosmoem 4 из команды желтых вступает в бой!

Cosmoem 4 борется с соперником.

Munchlax 1 теряет 19 здоровья.

Cosmoem 4 теряет 5 здоровья.

Munchlax 1 борется с соперником.

Cosmoem 4 теряет 7 здоровья.

Munchlax 1 теряет 2 здоровья.

Cosmoem 4 промахивается

Munchlax 1 промахивается

Cosmoem 4 промахивается

Munchlax 1 промахивается

Cosmoem 4 борется с соперником.

Munchlax 1 теряет 14 здоровья.

Cosmoem 4 теряет 4 здоровья.

Munchlax 1 использует Body Slam.

Cosmoem 4 теряет 7 здоровья.

Cosmoem 4 промахивается

Munchlax 1 использует Body Slam.

Cosmoem 4 теряет 11 здоровья.

Cosmoem 4 борется с соперником.

Munchlax 1 теряет 13 здоровья.

Cosmoem 4 теряет 3 здоровья.

Munchlax 1 промахивается

Cosmoem 4 борется с соперником.

Munchlax 1 теряет 19 здоровья.

Cosmoem 4 теряет 5 здоровья.

Munchlax 1 использует Body Slam.

Cosmoem 4 теряет 7 здоровья.

Cosmoem 4 парализован

Cosmoem 4 борется с соперником.

Критический удар!

Munchlax 1 теряет 30 здоровья.

Cosmoem 4 теряет 8 здоровья.

Munchlax 1 теряет сознание.

Bouffalant 2 из команды синих вступает в бой!

Cosmoem 4 использует Swagger.

Bouffalant 2 увеличивает атаку.

Bouffalant 2 использует Facade.

Cosmoem 4 теряет 25 здоровья.

Cosmoem 4 борется с соперником.

Критический удар!

Bouffalant 2 теряет 20 здоровья.

Cosmoem 4 теряет 5 здоровья.

Bouffalant 2 растерянно попадает по себе.

Bouffalant 2 теряет 10 здоровья.

Cosmoem 4 использует Swagger.

Bouffalant 2 увеличивает атаку.

Bouffalant 2 растерянно попадает по себе.

Bouffalant 2 теряет 15 здоровья.

Bouffalant 2 растерянно попадает по себе.

Bouffalant 2 теряет 16 здоровья.

Bouffalant 2 использует Rock Tomb.

Cosmoem 4 теряет 12 здоровья.

Cosmoem 4 уменьшает скорость.

Cosmoem 4 теряет сознание.

Solgaleo 5 из команды желтых вступает в бой!

Solgaleo 5 использует Swagger.

Bouffalant 2 увеличивает атаку.

Bouffalant 2 использует Rock Tomb.

Solgaleo 5 теряет 11 здоровья.

Solgaleo 5 уменьшает скорость.

Solgaleo 5 использует Swagger.

Bouffalant 2 увеличивает атаку.

Bouffalant 2 использует Facade.

Solgaleo 5 теряет 18 здоровья.

Solgaleo 5 промахивается

Bouffalant 2 использует Rock Tomb.

Solgaleo 5 теряет 12 здоровья.

Solgaleo 5 уменьшает скорость.

Solgaleo 5 использует Swagger.

Bouffalant 2 увеличивает атаку.

Bouffalant 2 растерянно попадает по себе.

Bouffalant 2 теряет 12 здоровья.

Solgaleo 5 использует Swagger.

Bouffalant 2 увеличивает атаку.

Bouffalant 2 использует Rock Tomb.

Solgaleo 5 теряет 11 здоровья.

Solgaleo 5 уменьшает скорость.

Bouffalant 2 использует Leer.

Solgaleo 5 уменьшает защиту.

Solgaleo 5 использует Aerial Ace.

Bouffalant 2 теряет 37 здоровья.

Bouffalant 2 теряет сознание.

Snorlax 3 из команды синих вступает в бой!

Snorlax 3 использует Body Slam.

Solgaleo 5 теряет 8 здоровья.

Solgaleo 5 борется с соперником.

Snorlax 3 теряет 34 здоровья.

Solgaleo 5 теряет 9 здоровья.

Snorlax 3 использует Body Slam.

Solgaleo 5 теряет 8 здоровья.

Solgaleo 5 борется с соперником.

Snorlax 3 теряет 25 здоровья.

Solgaleo 5 теряет 6 здоровья.

Snorlax 3 промахивается

Solgaleo 5 использует Aerial Ace.

Snorlax 3 теряет 39 здоровья.

Snorlax 3 использует Body Slam.

Solgaleo 5 теряет 6 здоровья.

Solgaleo 5 парализован

Solgaleo 5 использует Aerial Ace.

Snorlax 3 теряет 27 здоровья.

Snorlax 3 использует Pound.

Solgaleo 5 теряет 4 здоровья.

Solgaleo 5 использует Swagger.

Snorlax 3 увеличивает атаку.

Snorlax 3 использует Body Slam.

Solgaleo 5 теряет 11 здоровья.

Solgaleo 5 использует Aerial Ace.

Snorlax 3 теряет 45 здоровья.

Snorlax 3 использует Blizzard.

Solgaleo 5 теряет 7 здоровья.

Solgaleo 5 использует Aerial Ace.

Snorlax 3 теряет 40 здоровья.

Snorlax 3 теряет сознание.

В команде синих не осталось покемонов.

Команда желтых побеждает в этом бою!

Вывод

Для выполнения этой лабораторной работы нужно было ознакомиться с основными принципами ООП, понятиями объектов, классов, наследования, инкапсуляции, научиться переопределять методы.