Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет программной инженерии и компьютерной техники (ПИКТ)

Направление подготовки (специальность) – 09.03.04 (Нейротехнологии и программная инженерия)

Программирование

Лабораторная работа № 2

Вариант: 78729

Выполнил

студент

Немыкин Ярослав Алексеевич

Группа № 3122

Преподаватель: Данилов Павел Юрьевич

г. Санкт-Петербург

2024 г.

Оглавление

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. 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.

Покемоны:

Рис. 1. Покемоны

Исходный код программы:

Main

```
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 Carbink( name: "карабинчик", lvl: 55);
        Pokemon p2 = new Gligar( name: "гилгарик", lvl: 80);
        Pokemon p3 = new Gliscor( name: "глиссорчик", lvl: 80);
        Pokemon p4 = new Bellsprout( name: "беллспортик", lvl: 29);
        Pokemon p5 = new Weepinbell( name: "випинбеллчик", lvl: 29);
        Pokemon p6 = new Victreebel( name: "виктребельчик", lvl: 29);
        b.addAlly(p1);
        b.addFoe(p2);
        b.addAlly(p3);
        b.addFoe(p4);
        b.addAlly(p5);
        b.addFoe(p6);
        b.go();
    }
}
```

Рис. 2. Main

Pokemons

Bellsprout

```
package pokemons;

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

public class Bellsprout extends Pokemon {  2 usages  2 inheritors
    public Bellsprout(String name, int lvl){  2 usages
        super(name, lvl);
        super.setType(Type.GRASS, Type.POISON);
        super.setStats( v: 50,  v1: 75,  v2: 35,  v3: 70,  v4: 30,  v5: 40);
        super.setMove(new SweetScent(), new EnergyBall());
    }
}
```

Рис. 3. Bellsprout

Carbink

```
package pokemons;

import movement.*;
import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;

public class Carbink extends Pokemon {  1 usage
    public Carbink(String name, int lvl) {  1 usage
        super(name, lvl);
        super.setType(Type.FAIRY, Type.ROCK);
        super.setStats( v: 50,  v1: 50,  v2: 150,  v3: 50,  v4: 150,  v5: 50);
        super.setMove(new RockTomb(), new Moonblast(),
                new DazzlingGleam(), new PowerGem());
    }
}
```

Рис. 4. Carbink

Gligar

```
package pokemons;

import ru.ifmo.se.pokemon.Pokemon;
import ru.ifmo.se.pokemon.Type;
import movement.*;

public class Gligar extends Pokemon {  2 usages  1 inheritor
    public Gligar(String name, int lvl){  2 usages
        super(name, lvl);
        super.setType(Type.GROUND, Type.FLYING);
        super.setStats( v: 65,  v1: 75,  v2: 105,  v3: 35,  v4: 65,  v5: 85);
        super.setMove(new AerialAce(), new Venoshock(), new Roost());
    }
}
```

Рис. 5. Gligar

Gliscor

```
package pokemons;

import movement.*;
import ru.ifmo.se.pokemon.Type;

public class Gliscor extends Gligar {  1 usage
    public Gliscor(String name, int lvl) {  1 usage
        super(name, lvl);
        super.setType(Type.GROUND, Type.FLYING);
        super.setStats( v: 75,  v1: 95,  v2: 125,  v3: 45,  v4: 75,  v5: 95);
        super.setMove(new AerialAce(), new Venoshock(), new Roost(), new ThunderFang());
    }
}
```

Рис. 6. Gliscor

Victreebel

```java
package pokemons;

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

public class Victreebel extends Weepinbell{  1 usage
    public Victreebel(String name, int lvl) {  1 usage
        super(name, lvl);
        super.setType(Type.GRASS, Type.POISON);
        super.setStats( v: 65,  v1: 90,  v2: 50,  v3: 85,  v4: 45,  v5: 55);
        super.setMove(new SweetScent(), new EnergyBall(), new Acid(), new Swagger());
    }
}
```

Рис. 7. Victreebel

Weepinbell

```java
package pokemons;

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

public class Weepinbell extends Bellsprout {  2 usages  1 inheritor
    public Weepinbell(String name, int lvl) {  2 usages
        super(name, lvl);
        super.setType(Type.GRASS, Type.POISON);
        super.setStats( v: 65,  v1: 90,  v2: 50,  v3: 85,  v4: 45,  v5: 55);
        super.setMove(new SweetScent(), new EnergyBall(), new Acid());
    }
}
```

Рис. 8. Weepinbell

Movement

Acid

```java
package pokemons;
import movement.*;
import ru.ifmo.se.pokemon.*;

public class Bellsprout extends Pokemon {  2 usages  2 inheritors
    public Bellsprout(String name, int lvl){  2 usages
        super(name, lvl);
        super.setType(Type.GRASS, Type.POISON);
        super.setStats( v: 50,  v1: 75,  v2: 35,  v3: 70,  v4: 30,  v5: 40);
        super.setMove(new SweetScent(), new EnergyBall());
    }
}
```

Рис. 9. Acid

AerialAce

```java
package movement;

import ru.ifmo.se.pokemon.*;

public class AerialAce extends PhysicalMove {  2 usages
    public AerialAce(){  2 usages
        super.type = Type.FLYING;
        super.power = 60;
    }

    @Override  no usages
    protected boolean checkAccuracy(Pokemon pokemon, Pokemon pokemon1) {
        return true;
    }

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

Рис. 10. AerialAce

DazzlingGleam

```
package movement;

import ru.ifmo.se.pokemon.*;

public class DazzlingGleam extends SpecialMove {  1 usage
    public DazzlingGleam() { super(Type.FAIRY,  v: 80,  v1: 100); }

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

Рис. 11. DazzlingGleam

EnergyBall

```
package movement;

import ru.ifmo.se.pokemon.*;

public class EnergyBall extends SpecialMove {  3 usages
    public EnergyBall() { super(Type.GRASS,  v: 90,  v1: 100); }
    @Override  7 usages
    protected void applyOppEffects(Pokemon p){
        super.applyOppEffects(p);
        Effect energyBall = new Effect().chance( v: 0.1).turns( i: -1).stat(Stat.DEFENSE,  i: -1);
        p.addEffect(energyBall);
    }
    @Override
    protected String describe() {
        return "использует Энергетический шар";
    }
}
```

Рис. 12. EnergyBall

Moonblast

```java
package movement;

import ru.ifmo.se.pokemon.*;

public class Moonblast extends SpecialMove {  1 usage
    public Moonblast() {  1 usage
        super(Type.FAIRY,  v: 95,  v1: 100);
    }

    @Override  7 usages
    protected void applyOppEffects(Pokemon p){
        super.applyOppEffects(p);
        Effect moonblastEffect = new Effect().chance( v: 0.3).turns( i: -1).stat(Stat.ATTACK,  i: -1);
        p.addEffect(moonblastEffect);
    }

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

Рис. 13. Moonblast

PowerGem

```java
package movement;

import ru.ifmo.se.pokemon.*;

public class PowerGem extends SpecialMove {  1 usage
    public PowerGem() { super(Type.ROCK,  v: 80,  v1: 100); }

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

Рис. 14. PowerGem

RockTomb

```
package movement;

import ru.ifmo.se.pokemon.*;

public class RockTomb extends  PhysicalMove{  1 usage
    public RockTomb() {  1 usage
        super(Type.ROCK,  v: 60,  v1: 95);
    }
    @Override  7 usages
    protected void applyOppEffects(Pokemon p){
        super.applyOppEffects(p);
        Effect rockTombEffect = new Effect().turns( i: -1).stat(Stat.SPEED,  i: -1);
        p.addEffect(rockTombEffect);
    }
    @Override
    protected String describe() {
        return "использует Каменную гробницу";
    }
}
```

Рис. 15. RockTomb

Roost

```
package movement;

import ru.ifmo.se.pokemon.*;

public class Roost extends StatusMove {  2 usages
    public Roost() { super.type = Type.FLYING; }
    @Override  1 usage
    protected void applySelfEffects(Pokemon p){
        super.applySelfEffects(p);
        int hp = (int) (p.getStat(Stat.HP)*0.5 > p.getHP() ? 0.5*p.getStat(Stat.HP) : p.getStat(Stat.HP) - p.getHP());
        Effect RoostEffect = new Effect().chance( v: 1).turns( i: 0).stat(Stat.HP, hp);
        p.addEffect(RoostEffect);
    }
    @Override
    protected String describe() { return "использует Насест"; }
}
```

Рис. 16. Roost

Swagger

```
package movement;

import ru.ifmo.se.pokemon.*;

public class Swagger extends StatusMove {  1 usage
    public Swagger() { super(Type.NORMAL,  v: 0,  v1: 85); }
    @Override  7 usages
    protected void applyOppEffects(Pokemon p){
        super.applyOppEffects(p);
        p.confuse();
        Effect swaggerEffect = new Effect().turns( i: -1).stat(Stat.ATTACK,  i: 2);
        p.addEffect(swaggerEffect);
    }
    @Override
    protected String describe() { return "использует Развязность"; }
}
```

Рис. 17. Swagger

SweetScent

```
package movement;

import ru.ifmo.se.pokemon.*;

public class SweetScent extends StatusMove {  3 usages
    public SweetScent() { super(Type.NORMAL,  v: 0,  v1: 100); }
    @Override  7 usages
    protected void applyOppEffects(Pokemon p){
        super.applyOppEffects(p);
        Effect sweetScentEffect = new Effect().turns( i: -1).stat(Stat.EVASION,  i: -1);
        p.addEffect(sweetScentEffect);
    }
    @Override
    protected String describe() { return "использует Сладкий аромат"; }
}
```

Рис. 18. SweetScent

ThunderFang

```
package movement;

import ru.ifmo.se.pokemon.*;

public class ThunderFang extends PhysicalMove{  1 usage
    public ThunderFang(){  1 usage
        super(Type.ELECTRIC,  v: 65,  v1: 95);
    }

    @Override  7 usages
    protected void applyOppEffects(Pokemon p){
        super.applyOppEffects(p);
        Effect.flinch(p);
        Effect thunderFangEffect = new Effect().chance( v: 0.1).condition(Status.PARALYZE);
        p.addEffect(thunderFangEffect);
    }
}
```

Рис. 19. ThunderFang

Venoshock

```
package movement;

import ru.ifmo.se.pokemon.*;

public class Venoshock extends SpecialMove {  2 usages
    public Venoshock(){  2 usages
        super(Type.POISON,  v: 65,  v1: 100);
    }
    @Override
    protected String describe() {
        return "использует Веношок";
    }

    @Override  1 usage
    protected void applyOppDamage(Pokemon pokemon, double damage) {
        int multiplier = pokemon.getCondition() == Status.POISON ? 2 : 1;
        super.applyOppDamage(pokemon,  v: damage*multiplier);
    }
}
```

Рис. 20. Venoshock

UML:

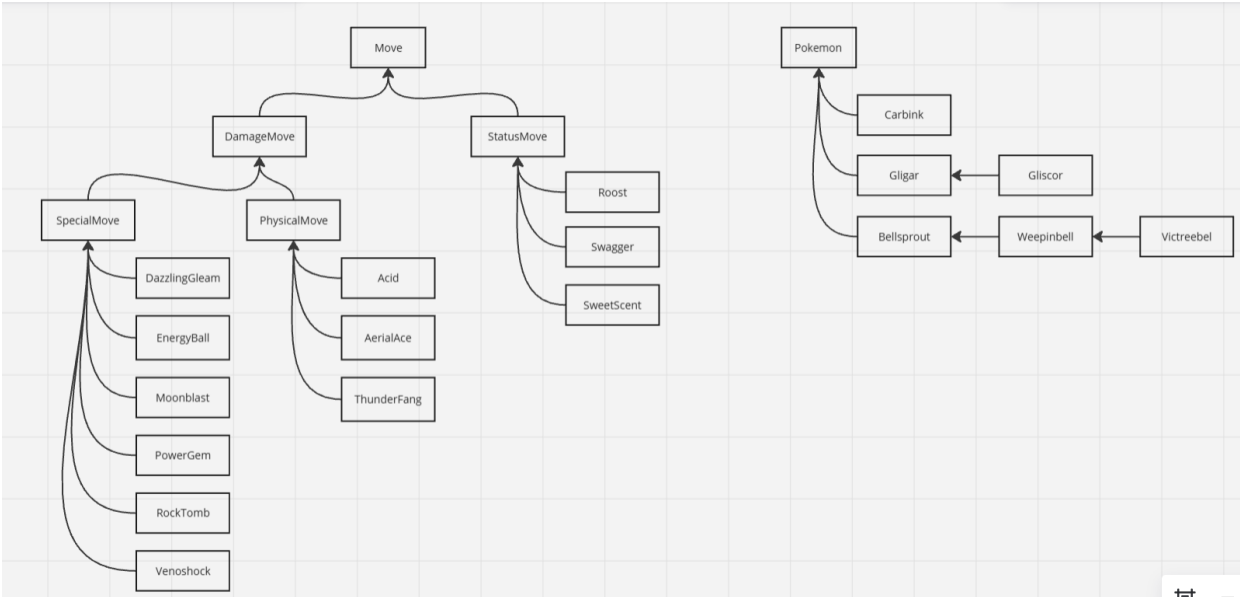

Рис. 2.

Пример результата работы программы:

Carbink карабинчик из команды белых вступает в бой!
Gligar гилгарик из команды полосатых вступает в бой!
Gligar гилгарик использует Веношок.
Carbink карабинчик теряет 12 здоровья.

Carbink карабинчик использует Силу самоцвета.
Критический удар!
Gligar гилгарик теряет 45 здоровья.

Gligar гилгарик использует Насест.

Carbink карабинчик использует Силу самоцвета.
Gligar гилгарик теряет 17 здоровья.

Gligar гилгарик теряет 45 здоровья.
Gligar гилгарик использует Веношок.
Carbink карабинчик теряет 12 здоровья.

Carbink карабинчик использует Силу самоцвета.
Gligar гилгарик теряет 24 здоровья.

Gligar гилгарик теряет 45 здоровья.
Gligar гилгарик использует Веношок.
Критический удар!
Carbink карабинчик теряет 26 здоровья.

Carbink карабинчик использует Лунный взрыв.
Gligar гилгарик теряет 24 здоровья.

Gligar гилгарик теряет 45 здоровья.
Bellsprout беллспортик из команды полосатых вступает в бой!
Carbink карабинчик использует Ослепительный блеск.
Bellsprout беллспортик теряет 36 здоровья.

Bellsprout беллспортик использует Сладкий аромат.

Carbink карабинчик использует Ослепительный блеск.
Bellsprout беллспортик теряет 34 здоровья.

Bellsprout беллспортик использует Энергетический шар.
Carbink карабинчик теряет 14 здоровья.

Carbink карабинчик использует Каменную гробницу.
Bellsprout беллспортик теряет 67 здоровья.
Bellsprout беллспортик теряет сознание.
Victreebel виктребельчик из команды полосатых вступает в бой!
Carbink карабинчик использует Ослепительный блеск.
Victreebel виктребельчик теряет 28 здоровья.

Victreebel виктребельчик использует Развязность.

Carbink карабинчик растерянно попадает по себе.
Carbink карабинчик теряет 25 здоровья.

Victreebel виктребельчик использует Кислоту.
Carbink карабинчик теряет 9 здоровья.

Carbink карабинчик использует Лунный взрыв.
Victreebel виктребельчик теряет 33 здоровья.

Victreebel виктребельчик использует Кислоту.

Carbink карабинчик теряет 5 здоровья.

Carbink карабинчик использует Каменную гробницу.
Victreebel виктребельчик теряет 65 здоровья.
Victreebel виктребельчик теряет сознание.
В команде полосатых не осталось покемонов.
Команда белых побеждает в этом бою!

Вывод:

Во время выполнения лабораторной работы я познакомился с основными принципами ООП, понятиями объектов, классов, наследования, инкапсуляции и научился переопределять методы