НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Программирование

Лабораторная работа № 3

Вариант 4253

Выполнил студент: Маликов Глеб Игоревич

Группа № P3124

Преподаватель: Харитонова Анастасия Евгеньевна

г. Санкт-Петербург

2022

Оглавление

Задание

Описание предметной области, по которой должна быть построена объектная модель:
После господина Спрутса выступил мебельный фабрикант и владелец лесопильных заводов Дубе, который прославился тем, что у него была тяжелая, словно вытесанная из дубового чурбака, голова, туго вертевшаяся из стороны в сторону и с трудом наклонявшаяся, когда ему требовалось посмотреть вниз. Коротышек с подобного рода головами среди лунатиков принято называть дуботолками. Господин Дубе сказал, что у него имеются две очень способные и даже талантливые, в своем роде, личности (именно так господин Дубе и выразился), которые могут взяться за это дельце и в два счета уберут с дороги Мигу и Жулио, а заодно и Незнайку с Козликом. Господин Спрутс сказал, что господин Дубе, видимо, его не понял, так как, говоря о том, что дело надо убить в зародыше, он вовсе не подразумевал, что кого-либо следует убить в буквальном смысле этого слова.

Программа должна удовлетворять следующим требованиям:

1. Доработанная модель должна соответствовать принципам SOLID.
2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
3. В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().
4. Программа должна содержать как минимум один перечисляемый тип (enum).

Порядок выполнения работы:

1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

Исходный код

Main.java

```
public class Main {
    public static void main(String[] args) {

        Senior s = new Senior("Спрутс");
        System.out.println(s);
        Senior d = new Senior("Дубе");
        d.addComo(Characteristic.DUB);
        System.out.println(d);

        Lichnost a = new Lichnost("Agent 47");
        a.addComo(Characteristic.HAB);
        a.addComo(Characteristic.TAL);
        System.out.println(a);
        Lichnost b = new Lichnost("Agent 48");
        b.addComo(Characteristic.HAB);
        b.addComo(Characteristic.TAL);
        System.out.println(b);

        Storychar m = new Storychar("Миго");
        Storychar j = new Storychar("Жулио");
        Storychar n = new Storychar("Незнайка");
        Storychar k = new Storychar("Козлик");
        System.out.println(m);
        System.out.println(j);
        System.out.println(n);
        System.out.println(k);


        System.out.println("---");
        s.talk("Моё выступление окончено");
        d.talk("У меня имеются две личности: " + a.getName() + " он" + a.get-
ComoTitles() + " и " + b.getName() + " он" + b.getComoTitles());
        d.talk("Они могут взяться за это дельце и в два счета уберут с дороги
" + m.getName() + " и " + j.getName() + ", а заодно и " + n.getName() + " с "
+ k.getName());
        s.talk("Вы господин " + d.getName() + ", видимо меня не поняли, так
как, говоря о том, что дело надо убить в зародыше, я вовсе не подразумевал,
что кого-либо следует убить в буквальном смысле этого слова.");
        System.out.println("---");

        a.kill(m);
        a.kill(j);
        a.kill(n);
        b.kill(k);

        System.out.println(a);
        System.out.println(b);

        System.out.println(m);
        System.out.println(j);
        System.out.println(n);
        System.out.println(k);

    }
```

```
}
```

Abstract class Personaje

```
import java.util.ArrayList;

abstract class Personaje implements Die {
    protected String name;
    protected ArrayList<Characteristic> como;

    Personaje() {
        this.name = "Безымянный";
        this.como = new ArrayList<Characteristic>();
    }
    Personaje(String name) {
        this();
        this.name = name;
    }


    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void addComo(Characteristic c) {
        this.como.add(c);
    }

    public ArrayList getComo() {
        return como;
    }
    public String getComoTitles() {
        if (como == null || como.isEmpty()) {
            return "без характеристик";
        }
        String ret = "";
        for (Characteristic c : como) {
            ret = ret + " " + c.getTitle();
        }
        return ret;
    }

    @Override
    public void die() {
        this.addComo(Characteristic.DED);
        System.out.println(getName() + " Умер :(");
    }


}
```

Enum Characteristic

```
public enum Characteristic {
    DUB("Дуботолк"),
    HAB("Способный"),
```

```
    TAL("Талантливый"),
    DED("Мёртвый");

    private final String title;

    Characteristic(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }
}
```

Interface Die

```
public interface Die {
    void die();
}
```

Interface Killer

```
public interface Killer {
    void kill(Personaje p);
    int getKillcount();

}
```

Interface Talker

```
public interface Talker {
    void talk(String s);
}
```

Class Lichnost

```
public class Lichnost extends Personaje implements Killer{

    private int killcount = 0;

    Lichnost(){
        super();
    }
    Lichnost(String name){
        super(name);
    }

    @Override
    public int getKillcount() {
        return killcount;
    }

    @Override
    public void kill(Personaje p) {
        p.die();
        this.killcount++;
        System.out.println(getName() + " Убил " + p.getName());
    }

    @Override
    public String toString() {
        return this.getClass() + " {" + "name = '" + name + '\'' + ", como =
" + this.getComoTitles() + ", killcount = " + killcount +'}';
    }
```

```
    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (obj == null || obj.getClass() != this.getClass()) {
            return false;
        }
        Lichnost comp = (Lichnost) obj;
        return this.killcount == ((Lichnost) obj).killcount &&
name.equals(comp.name) && como.equals(comp.como);

    }

    @Override
    public int hashCode() {
        int result = 7;
        if (name != null) {
            result = 13 * result + name.hashCode();
        }
        if (como != null) {
            result = 13 * result + como.hashCode();
        }
        result += killcount;
        return result;
    }
}
```

Class Storychar

```
public class Storychar extends Personaje{

    String info = "Нет информации";

    Storychar(){
        super();
    }
    Storychar(String name){
        super(name);
    }

    Storychar(String name, String info){
        super(name);
        this.info = info;
    }

    @Override
    public String toString() {
        return this.getClass() + " {" + "name = '" + name + '\'' + ", como =
" + this.getComoTitles() + ", info = " + info + '}';
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (obj == null || obj.getClass() != this.getClass()) {
```

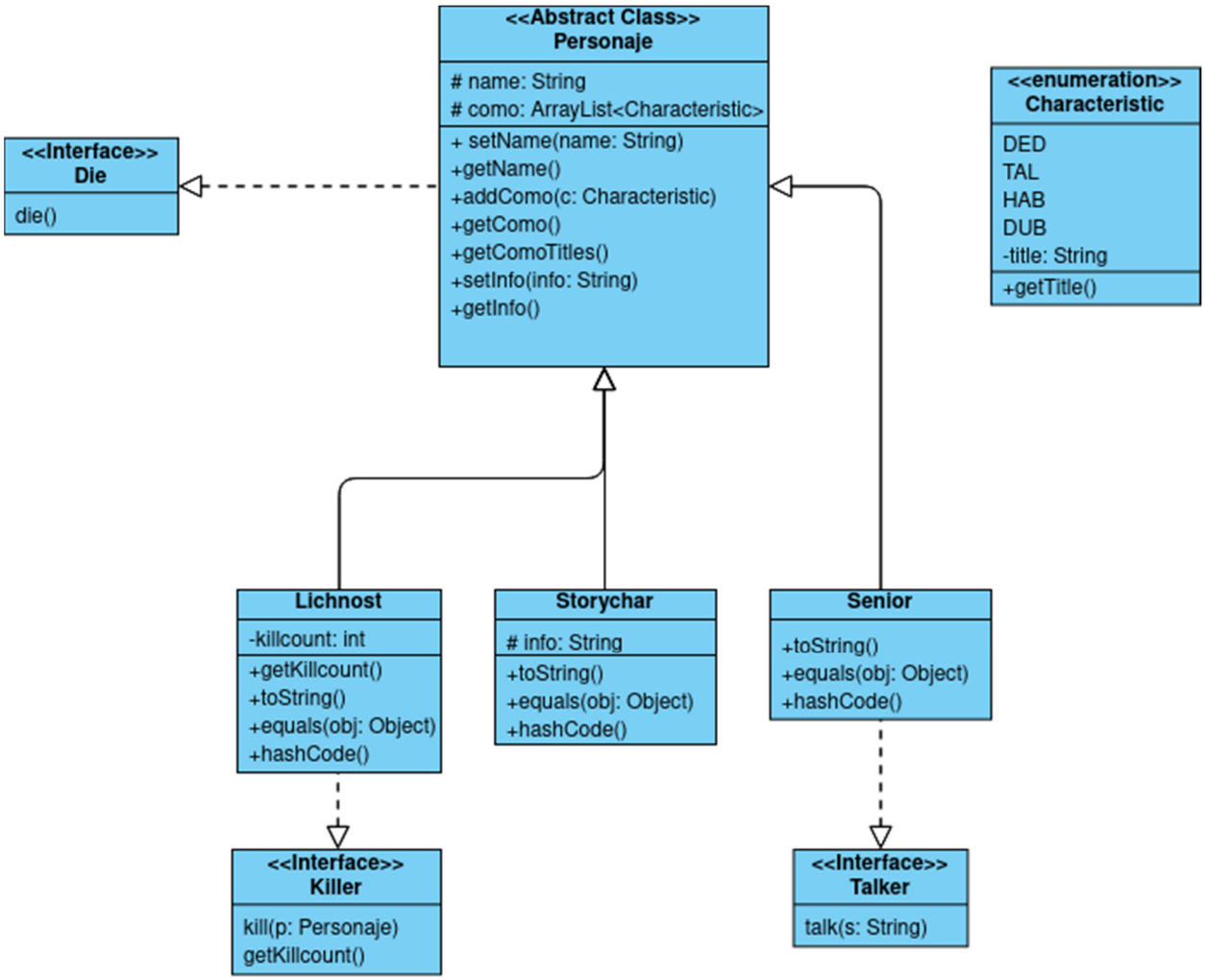
```
            return false;
        }
        Storychar comp = (Storychar) obj;
        return name.equals(comp.name) && como.equals(comp.como) &&
info.equals(comp.info);

    }

    @Override
    public int hashCode() {
        int result = 7;
        if (name != null) {
            result = 13 * result + name.hashCode();
        }
        if (como != null) {
            result = 13 * result + como.hashCode();
        }
        if (info != null) {
            result = 13 * result + info.hashCode();
        }
        return result;
    }

}
```

Class Senior

```
public class Senior extends Personaje implements Talker {

    Senior(){
        super();
    }
    Senior(String name){
        super("Господин " + name);
    }

    public void talk(String text) {
        System.out.println(getName() + " сказал \"" + text + "\"");
    }


    @Override
    public String toString() {
        return this.getClass() + " {" + "name = '" + name + '\'' + ", como =
" + this.getComoTitles() + '}';
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (obj == null || obj.getClass() != this.getClass()) {
            return false;
        }
        Personaje comp = (Personaje) obj;
        return name.equals(comp.name) && como.equals(comp.como);

    }
```

```
    @Override
    public int hashCode() {
        int result = 7;
        if (name != null) {
            result = 13 * result + name.hashCode();
        }
        if (como != null) {
            result = 13 * result + como.hashCode();
        }
        return result;
    }
}
```

Диаграмма классов



Результат работы программы

```
[s372819@helios ~/lab3p]$ java -jar Lab3.jar
class Senior {name = 'Господин Спрутс', como = без характеристик}
class Senior {name = 'Господин Дубе', como =  Дуботолк}
class Lichnost {name = 'Agent 47', como =  Способный Талантливый, killcount =
0}
class Lichnost {name = 'Agent 48', como =  Способный Талантливый, killcount =
0}
class Storychar {name = 'Миго', como = без характеристик, info = Нет
информации}
class Storychar {name = 'Жулио', como = без характеристик, info = Нет
информации}
class Storychar {name = 'Незнайка', como = без характеристик, info = Нет
информации}
class Storychar {name = 'Козлик', como = без характеристик, info = Нет
информации}
---
Господин Спрутс сказал "Моё выступление окончено"
Господин Дубе сказал "У меня имеются две личности: Agent 47 он Способный
Талантливый и Agent 48 он Способный Талантливый"
Господин Дубе сказал "Они могут взяться за это дельце и в два счета уберут с
дороги Миго и Жулио, а заодно и Незнайка с Козлик"
Господин Спрутс сказал "Вы господин Господин Дубе, видимо меня не поняли, так
как, говоря о том, что дело надо убить в зародыше, я вовсе не подразумевал,
что кого-либо следует убить в буквальном смысле этого слова."
---
Миго Умер :(
Agent 47 Убил Миго
Жулио Умер :(
Agent 47 Убил Жулио
Незнайка Умер :(
Agent 47 Убил Незнайка
Козлик Умер :(
Agent 48 Убил Козлик
class Lichnost {name = 'Agent 47', como =  Способный Талантливый, killcount =
3}
class Lichnost {name = 'Agent 48', como =  Способный Талантливый, killcount =
1}
class Storychar {name = 'Миго', como =  Мёртвый, info = Нет информации}
class Storychar {name = 'Жулио', como =  Мёртвый, info = Нет информации}
class Storychar {name = 'Незнайка', como =  Мёртвый, info = Нет информации}
class Storychar {name = 'Козлик', como =  Мёртвый, info = Нет информации}
[s372819@helios ~/lab3p]$
```

Вывод

В ходе работы были использованы абстрактный класс, интерфейсы и перечисление enum для отображения текста в виде объектной модели. Были переопределены методы equals(), hashCode() и toString() для классов.

Список литературы

А.В. Гаврилов, С.В. Клименков, Ю.А. Королёва, А.Е. Харитонова, Е.А. Цопа (2019) ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ JAVA – СПб: Университет ИТМО.

Письмак А.Е. (2022) Программирование. Конспект лекций – СПБ: Университет ИТМО.