Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 3

По дисциплине

“ Программирование”

Вариант 3117550

Выполнил:

Студент группы P3117

Пономарёв М. И.

Преподаватель:

Письмак А. Е.

Санкт-Петербург
2022

Оглавление

Текст задания

Лабораторная работа #3

Введите вариант: 3117550

Описание предметной области, по которой должна быть построена объектная модель:

> И в то же мгновение на его рукопись шмякнулась слива и сделала большущее синее пятно. И он обернулся с решимостью как следует намять им холку. Но не тут-то было: его взгляд уперся в буйные заросли каких-то кустарников, обсыпанных желтыми ягодами. Муми-папа так и подскочил на месте, и тут уж на его письменный стол обрушился целый дождь синих слив. Весь потолок был наглухо заткан сплетением веток, они росли прямо на глазах и тянули свои зеленые руки к окну.

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.

Отчёт по работе должен содержать:

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

Основные этапы вычисления

UML диаграмма

Link to image

Action

```
public interface Action {
    void make_action(String action, Essence obj);
    void make_action(String action);
}
```

AnimateObject

```
public class AnimateObject extends Essence{
    public AnimateObject(String name){
        super(name);
    }
    public void wants_to_do(String action){
        System.out.println(this + " хочет " + action);
    }
    public void stare_at(Essence obj){
        System.out.println(this + " смотрит на " + obj);
    }
}
```

Berry

```
public class Berry extends InanimateObject{
    private Color color;
    public Berry(String name, Color color)
    {
        super(name);
        this.color = color;
    }
    public Berry(String name)
    {
        super(name);
        this.color = Color.NONE;
    }
    public Color getColor(){
        return this.color;
    }
    public void crush_into(Essence obj){
        System.out.println(this + " упал на " + obj);
        this.setState("разбилась");
        this.announce_state();
    }
    @Override
    public int hashCode(){
        return this.getName().hashCode() * this.color.hashCode();
    }
    @Override
    public String toString(){
        return this.getColor() + " " + this.getName();
    }
}
```

Bush

```
public class Bush extends InanimateObject{
    public Bush(String name){
        super(name);
    }
    public void grow(){
        System.out.println(this + " растут");
    }
    public void filledWith(Essence obj){
        System.out.println(this + " обсыпаны " + obj);
    }
}
```

Color

```
public enum Color {
    BLUE{
        @Override
        public String toString(){
            return "Синяя";
        }
    },
    YELLOW{
        @Override
        public String toString(){
            return "Жёлтая";
        }
    },
    NONE{
        @Override
        public String toString(){
            return "";
        }
    }
}
```

Essence

```
public abstract class Essence implements Action {
    private String name;
    public Essence(String name){
        this.name = name;
    }
    public String getName(){
        return this.name;
    }
    public void make_action(String action){
        System.out.println(this + " " + action);
    }
    public void make_action(String action, Essence obj){
        System.out.println(this + " " + action + " " + obj);
    }
```

```
    @Override
    public int hashCode(){
        return this.getName().hashCode();
    }
    @Override
    public boolean equals(Object obj){
        return obj.hashCode() == this.hashCode();
    }
    @Override
    public String toString(){
        return this.getName();
    }
}
```

InanimateObject

```
public class InanimateObject extends Essence implements State{
    private String state;
    public InanimateObject(String name){
        super(name);
        this.state = "None";
    }
    public void setState(String state){
        this.state = state;
    }
    public String getState(){
        return this.state;
    }
    public void announce_state(){
        System.out.println(this + " " + this.getState());
    }
}
```

Main

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

        Berry plum = new Berry("Слива", Color.BLUE);
        InanimateObject manuscript = new InanimateObject("Рукопись");
        InanimateObject table = new InanimateObject("Письменный стол");
        Bush bushes = new Bush("Кустарники");
        InanimateObject ceiling = new InanimateObject("Потолок");
        AnimateObject character = new AnimateObject("Муми-папа");
        Berry yellow_berries = new Berry("Ягоды", Color.YELLOW);

        plum.crush_into(manuscript);
        plum.make_action("оставила большое пятно на", manuscript);
        character.make_action("обернулся с решимостью");
        character.wants_to_do("намять как сделедует им холку");
        character.stare_at(bushes);
        bushes.filledWith(yellow_berries);
        character.make_action("подскочил на месте");

        Berry[] plums = new Berry[5];
```

```
        for (int i = 0; i < 5; i++) {
            plums[i] = new Berry("Слива", Color.BLUE);
            plums[i].crush_into(table);
        }

        ceiling.setState("заткан сплетением веток");
        ceiling.announce_state();
        bushes.grow();
        bushes.make_action("тянут свои зелёные руки к окну");

    }
}
```

State

```
public interface State {
    void setState(String state);
    String getState();
    void announce_state();
}
```

Вывод программы

Синяя Слива упал на Рукопись

Синяя Слива разбилась

Синяя Слива оставила большое пятно на Рукопись

Муми-папа обернулся с решимостью

Муми-папа хочет намять как сделедует им холку

Муми-папа смотрит на Кустарники

Кустарники обсыпаны Жёлтая Ягоды

Муми-папа подскочил на месте

Синяя Слива упал на Письменный стол

Синяя Слива разбилась

Синяя Слива упал на Письменный стол

Синяя Слива разбилась

Синяя Слива упал на Письменный стол

Синяя Слива разбилась

Синяя Слива упал на Письменный стол

Синяя Слива разбилась

Синяя Слива упал на Письменный стол

Синяя Слива разбилась

Потолок заткан сплетением веток

Кустарники растут

Кустарники тянут свои зелёные руки к окну

Вывод

В данной лабораторной работе я применил принципы SOLID для создания абстрактных классов, классов, интерфейсов, enum для описания ситуации в задании.