Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

По программированию

Вариант 312000

Выполнил:
Студент группы P3117
Васильченко Роман Антонович
Преподаватель:
Письмак А.Е.

Санкт-Петербург

2021

Оглавление

Задание

Лабораторная работа #3

Введите вариант: 312000

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

> Ощупывая перед собой землю тростью, Скуперфильд добрался до противоположного склона оврага и стал карабкаться по нему вверх. Несколько раз он срывался и скатывался обратно, но наконец ему все же удалось выбраться на поверхность. Отдышавшись немного и заметив, что туман стал прозрачнее, Скуперфильд отправился дальше. Вскоре туман рассеялся, и Скуперфильд обнаружил, что шагает по рыхлой земле, усаженной какими-то темно-зелеными, ломкими кустиками, достигавшими ему до колен. Выдернув из земли один кустик, он увидел несколько прицепившихся к корням желтоватых клубней. Осмотрев клубни внимательно, Скуперфильд начал догадываться, что перед ним самый обыкновенный картофель. Впрочем, он далеко не был уверен в своей догадке, так как до этого видел картофель только в жареном или вареном виде и к тому же почему-то воображал, что картофель растет на деревьях.

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:

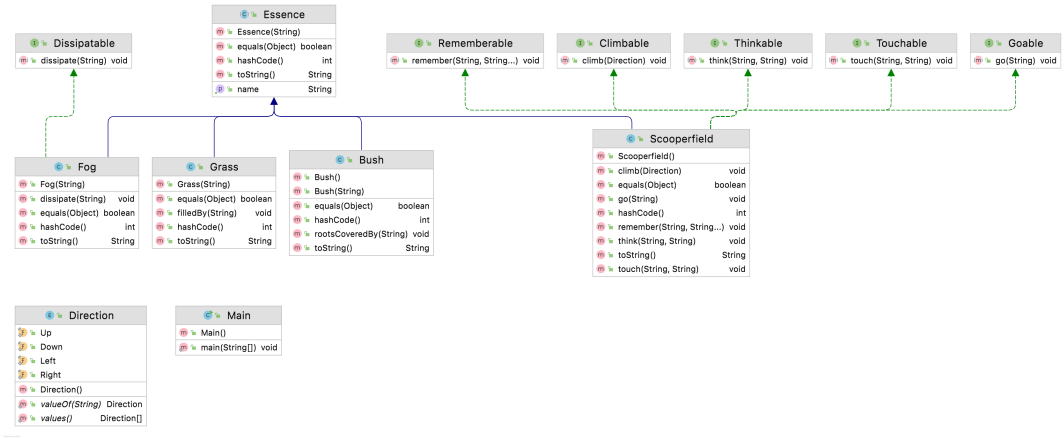
1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

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



Main

```
import java.util.List;
import java.util.Random;

public class Main {
    public static void main(String[] args) {
        Scooperfield scooperfield = new Scooperfield();
        Fog fog = new Fog("туман");
        Grass grass = new Grass("Рыхлая земля");
        Random random = new Random();

        scooperfield.touch("трость", "земля");
        scooperfield.go("противоположный склон оврага");
        scooperfield.climb(Direction.Up);

        scooperfield.climb(Direction.Down);
        scooperfield.climb(Direction.Up);
        scooperfield.climb(Direction.Down);
        scooperfield.climb(Direction.Up);

        scooperfield.go("поверхности");
        fog.dissipate("прозрачнее");
        scooperfield.go("рыхлая земля");
        fog.dissipate("рассеялся");
        grass.filledBy(" темно-зеленые, ломкие кустики");
        Bush[] bushes = new Bush[13];
        bushes[2] = new Bush("темно-зеленый, ломкий кустик");
        bushes[2].rootsCoveredBy("картофель");
        scooperfield.remember("картофель", "жареный", "вареный");
        scooperfield.think("картофель", "растет на деревьях");
    }
}
```

Bush

```
public class Bush extends Essence{
    public Bush(String name) {
        super(name);
    }
    public Bush(){
        super("темно-зеленымй, ломкий кустик");
    }
    public void rootsCoveredBy(String objects){
        System.out.println("Корни покрыты " + objects);
    }

    @Override
    public int hashCode() {
        return super.hashCode() + this.getName().hashCode();
    }

    @Override
    public boolean equals(Object obj) {
        return obj.hashCode() == this.hashCode();
    }

    @Override
    public String toString() {
        return "Имя: " + getName();
    }
}
```

Climbable

```
public interface Climbable {
    public void climb(Direction direction);
```

Direction

```
public enum Direction {
    Up,
```

```
}
```

```
    Down,
    Left,
    Right
}
```

Dissipatable

```
public interface Dissipatable {
    public void dissipate(String state);
}
```

Goable

```
public interface Goable {
    public void go(String toObject);
}
```

Essence

```
public abstract class Essence {
    private String name;
    public Essence(String name){
        this.name = name;
    }
    public String getName(){
        return this.name;
    }

    @Override
    public int hashCode() {
        return super.hashCode() + this.getName().hashCode();
    }

    @Override
    public boolean equals(Object obj) {
        return obj.hashCode() == this.hashCode();
    }

    @Override
    public String toString() {
        return "Имя: " + getName();
    }
}
```

Fog

```
public class Fog extends Essence implements Dissipatable {
    public Fog(String name) {
        super(name);
        System.out.println("На улице туман");
    }
    @Override
    public int hashCode() {
        return super.hashCode() + this.getName().hashCode();
    }

    @Override
    public boolean equals(Object obj) {
        return obj.hashCode() == this.hashCode();
    }
    @Override
    public String toString() {
        return "Имя: " + getName();
    }
    @Override
    public void dissipate(String state) {
        System.out.println("Туман стал " + state);
    }
}
```

Grass

```
public class Grass extends Essence{
    public Grass(String name) {
        super(name);
    }
    public void filledBy(String objects){
        System.out.println(getName() + " усаженная " + objects);
    }
    @Override
    public int hashCode() {
        return super.hashCode() + this.getName().hashCode();
    }
    @Override
    public boolean equals(Object obj) {
        return obj.hashCode() == this.hashCode();
    }
    @Override
    public String toString() {
        return "Имя: " + getName();
    }
}
```

Rememberable

```
public interface Rememberable {

    void remember(String object, String... states);
}
```

Thinkable

```
public interface Thinkable {
    public void think(String object, String action);
}
```

Touchable

```
public interface Touchable {
    public void touch(String byObject, String toObject);
}
```

Scooperfield

```
public class Scooperfield extends Essence implements Touchable, Goable, Climbable, Rememberable, Thinkable {
    public Scooperfield(){
        super("Скуперфильд");
    }
    @Override
    public int hashCode() {
        return super.hashCode() + this.getName().hashCode();
    }
    @Override
    public boolean equals(Object obj) {
        return obj.hashCode() == this.hashCode();
    }
    @Override
    public String toString() {
        return "Имя: " + getName();
    }
    @Override
    public void touch(String byObject, String toObject) {
        System.out.println(getName() + " ощупывая " + toObject + " при помощи " + byObject);
    }
    @Override
    public void go(String toObject) {
        System.out.println(getName() + " добрался до " + toObject);
    }
    @Override
    public void climb(Direction direction) {
        System.out.print(getName() + " начал ");
        switch (direction){
            case Up:
                System.out.println("карабкаться наверх");
                break;
            case Down:
                System.out.println("срываться и скатываться вниз");
                break;
            case Left:
                System.out.println("карабкаться налево");
                break;
            case Right:
                System.out.println("карабкаться направо");
                break;
            default:
                break;
        }
    }
    @Override
    public void remember(String object, String... states) {
        System.out.print(getName() + " помнит только ");
        for (int i = 0; i < states.length; i++) System.out.print(states[i] + ", ");
        System.out.println(object);
    }
    @Override
    public void think(String object, String action) {
        System.out.println(getName() + "думал, что " + object + " " + action);
    }
}
```

Вывод программы

```
На улице туман
Скуперфильд ощупывая земля при помощи трость
Скуперфильд добрался до противоположный склон оврага
Скуперфильд начал карабкаться наверх
Скуперфильд начал срываться и скатываться вниз
Скуперфильд начал карабкаться наверх
Скуперфильд начал срываться и скатываться вниз
Скуперфильд начал карабкаться наверх
Скуперфильд добрался до поверхности
Туман стал прозрачнее
Скуперфильд добрался до рыхлая земля
Туман стал рассеялся
Рыхлая земля усаженная  темно-зелеными, ломкие кустики
Корни покрыты картофель
Скуперфильд помнит только жареный, вареный, картофель
Скуперфильддумал, что картофель растет на деревьях
```

Вывод

Во время написания данной лабораторной работы я изучил принципы SOLID, а также использовал абстрактные классы, интерфейсы и enum для описания ситуации в задании, а также научился переопределять некоторые методы.