МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе №4
в рамках дисциплины «Программирование»

Выполнил: А. С. Медведев
Студент группы R3237

Санкт-Петербург

2021

Цель работы:

- Доработать программу из лабораторной работы #3, обновив реализацию объектной модели в соответствии с новой версией описания предметной области.

Программа должна удовлетворять следующим требованиям:

- В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.
- В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).

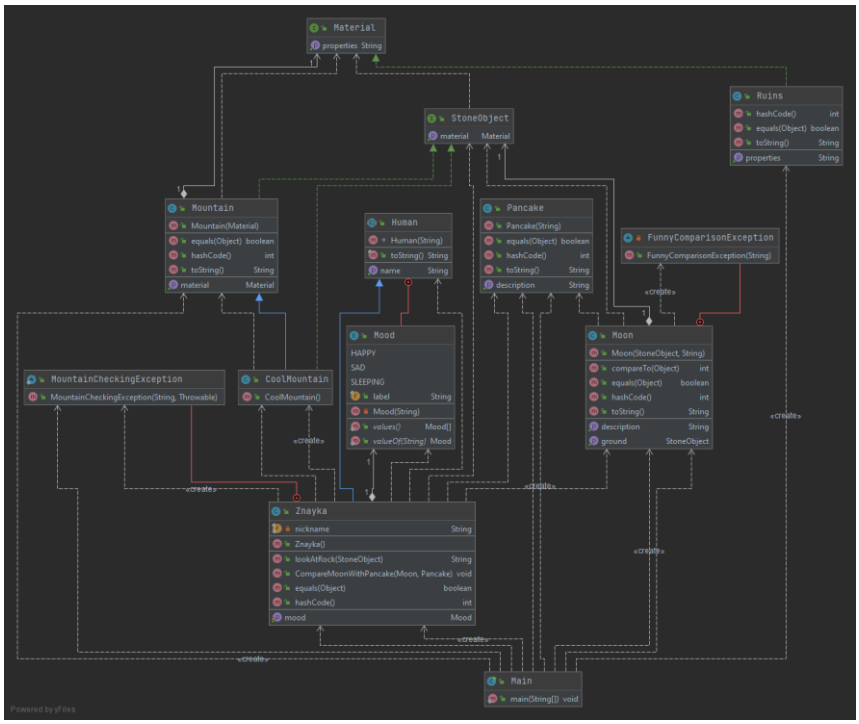
Порядок выполнения работы:

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

Выполнение работы:

Исходный код программы представлен в Git-репозитории oaleksander/Lab4 (github.com).

Диаграмма классов:



Выводы:

В ходе данной лабораторной работы была построена объектная модель с соблюдением принципов SOLID, с интерфейсами, абстрактными классами и перечисляемыми типами, собственными классами исключений и локальными, анонимными и вложенными классами.

Файл Main.java

```java
package com.company;

public class Main {
    public static void main(String[] args) {
        Moon moon = new Moon(new Mountain(new Ruins()), "крупное космическое тело");
        Pancake pancake = new Pancake("несчастный из прокисшего теста");
        Znayka znayka = new Znayka();
        try {
            znayka.CompareMoonWithPancake(moon, pancake);
        } catch (RuntimeException e) {
            System.out.println(e.getMessage());
        }
        try {
            System.out.println(znayka.lookAtRock(moon.getGround()));
        } catch (Znayka.MountainCheckingException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

Файл Moon.java

```java
package com.company;

import java.util.Objects;

public class Moon {

    final StoneObject ground;

    final String description;

    public Moon(StoneObject ground, String description) {
        this.ground = ground;
        this.description = description;
    }

    public int compareTo(Object o) {
        if (o.equals(this)) return 0;
        if (o instanceof Pancake)
            throw new FunnyComparisonException("Смешно сравнивать " + this.toString() + " с " + o.toString());
        return this.hashCode() - o.hashCode();
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Moon)) return false;
        Moon moon = (Moon) o;
        return description.equals(moon.description);
    }

    @Override
    public int hashCode() {
        return Objects.hash(description);
    }

    @Override
    public String toString() {
        return "Луна - " + getDescription();
    }

    public StoneObject getGround() {
        return ground;
    }

    public String getDescription() {
        return description;
    }

    private class FunnyComparisonException extends IllegalArgumentException {
        public FunnyComparisonException(String message) {
            super(message);
        }
    }
}

```

Файл Human.java

```java
package com.company;

public abstract class Human {

    private final String name;

    Human(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public final String toString() {
        return "Человек по имени " + name;
    }

    public enum Mood {
        HAPPY("Весело"),
        SAD("Грустно"),
        SLEEPING("Сонный");

        public final String label;

        Mood(String label) {
            this.label = label;
        }
    }
}

```

Файл Ruins.java

```java
package com.company;

public class Ruins implements Material {

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

    @Override
    public boolean equals(Object obj) {
        return super.equals(obj);
    }

    @Override
    public String toString() {
        return getProperties() + ", потерявшие первончальную форму";
    }

    @Override
    public String getProperties() {
        return "Остатки разрушившейся от времени гигантской кирпичной стены";
    }
}

```

Файл Znayka.java

```java
package com.company;

import java.util.Objects;

public class Znayka extends Human {

    private static final String nickname = "Знайка";
    private Mood mood;
    public Znayka() {
        super(nickname);
    }

    public Mood getMood() {
        return mood;
    }

    public void setMood(Mood mood) {
        this.mood = mood;
    }

    public String lookAtRock(StoneObject mountain) throws MountainCheckingException {
        String response;
        try {
            if (mountain.getClass() == CoolMountain.class) {
                response = this.toString() + " нашел " + mountain.toString() + ". Какая радость!";
                setMood(Mood.HAPPY);
            } else {
                setMood(Mood.SAD);
                throw new IllegalArgumentException();
            }
        } catch (IllegalArgumentException err) {
            throw new MountainCheckingException(mountain.toString() + " не содержит " + new CoolMountain
().getMaterial().getProperties(), err);
        }
        return response;
    }

    public void CompareMoonWithPancake(Moon moon, Pancake pancake) {
        moon.compareTo(pancake);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Znayka)) return false;
        Znayka znayka = (Znayka) o;
        return getMood() == znayka.getMood();
    }

    @Override
    public int hashCode() {
        return Objects.hash(getMood());
    }

    public static class MountainCheckingException extends Exception {
        public MountainCheckingException(String errorMessage, Throwable err) {
            super(errorMessage, err);
        }
    }
}

```

Файл Pancake.java

```java
package com.company;

import java.util.Objects;

public class Pancake {
    final String description;

    public Pancake(String description) {
        this.description = description;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Pancake)) return false;
        Pancake pancake = (Pancake) o;
        return Objects.equals(description, pancake.description);
    }

    @Override
    public int hashCode() {
        return Objects.hash(description);
    }

    @Override
    public String toString() {
        return "Блин - " + getDescription();
    }

    public String getDescription() {
        return description;
    }
}

```

Файл Material.java

```java
package com.company;

public interface Material {
    String getProperties();
}

```

Файл Mountain.java

```java
package com.company;

import java.util.Objects;

public class Mountain implements StoneObject {

    private final Material material;

    public Mountain(Material material) {
        this.material = material;
    }

    @Override
    public Material getMaterial() {
        return material;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Mountain)) return false;
        Mountain that = (Mountain) o;
        return Objects.equals(getMaterial(), that.getMaterial());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getMaterial());
    }

    @Override
    public String toString() {
        return "Гора из " + material.getProperties();
    }
}

```

Файл StoneObject.java

```java
package com.company;

public interface StoneObject {
    Material getMaterial();
}

```

Файл CoolMountain.java

```java
package com.company;

public class CoolMountain extends Mountain implements StoneObject {
    public CoolMountain() {
        super(() -> "Куски горной породы");
    }
}

```