Федеральное государственное автономное образовательное учреждение

высшего образования «Национальный исследовательский университет

ИТМО»

*Факультет программной инженерии и компьютерной техники*

*Направление подготовки: 09.03.04 – Нейротехнологии и программирование*

*Дисциплина «Программирование»*

**Отчет**

**По лабораторной работе №3**

**Вариант №88888**

Выполнил:

Мухин Никита Денисович

Группа: Р3123

Преподаватель:

Данилов Павел Юрьевич

Г. Санкт-Петербург, 2023 г.

**Оглавление**

[Задание 2](#_Toc114643031)

UML [диаграмма классов 3](#_Toc114643032)

[Код программы:](#_Toc114643033) 4

[Результат выполнения программы: 7](#_Toc114643034)

[Вывод](#_Toc114643035) 8

[Список литературы.](#_Toc114643036) 9

# Задание

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

Размечтавшись, Знайка не заметил, как погрузился в сон. И во сне ему снилась космическая ракета, и Луна, и лунные коротышки, и ещё много разных интересных вещей. А наутро Знайка исчез. К завтраку он не явился, а когда коротышки пришли к нему в комнату, они увидели на столе записку, в которой было всего три слова: «В Солнечный город», и подпись: «Знайка». Прочитав записку, коротышки сразу поняли, что Знайка уехал в Солнечный город.

**Программа должна удовлетворять следующим требованиям:**

1. Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).
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:**

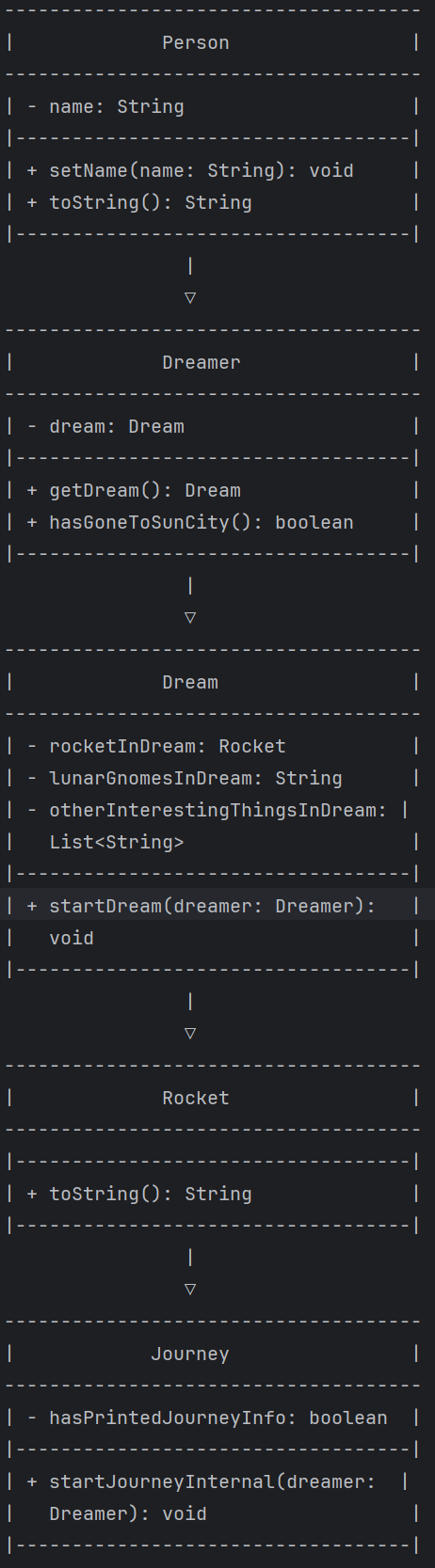
1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

**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.

# 

# UML-диаграмма классов:



# Код программы:

#### Классы:

// Класс Dream, представляющий сон  
class Dream {  
  
 private Rocket rocketInDream;  
 private String lunarGnomesInDream;  
 private List<String> otherInterestingThingsInDream;  
 private boolean hasPrintedDreamInfo = false;  
  
 public Dream(Rocket rocketInDream, String lunarGnomesInDream, List<String> otherInterestingThingsInDream) {  
 this.rocketInDream = rocketInDream;  
 this.lunarGnomesInDream = lunarGnomesInDream;  
 this.otherInterestingThingsInDream = otherInterestingThingsInDream;  
 }  
 public void startDream(Dreamer dreamer) {  
 if (!hasPrintedDreamInfo) {  
 Random random = new Random();  
 int randomNumber = random.nextInt(5);  
 switch (randomNumber) {  
 case 0:  
 System.*out*.println("Размечтавшись, " + dreamer + " не заметил, как погрузился в сон.");  
 System.*out*.println("И во сне ему снился сон, в котором он летал на космическом корабле по галактике.");  
 break;  
 case 1:  
 System.*out*.println("Размечтавшись, " + dreamer + " не заметил, как погрузился в сон.");  
 System.*out*.println("И во сне ему снилось, что он встретил инопланетян.");  
 break;  
 case 2:  
 System.*out*.println("Размечтавшись, " + dreamer + " не заметил, как погрузился в сон.");  
 System.*out*.println("И во сне ему снилось, что он стал супергероем.");  
 break;  
 case 3:  
 System.*out*.println("Размечтавшись, " + dreamer + " не заметил, как погрузился в сон.");  
 System.*out*.println("И во сне ему снилось, что он выиграл в лотерею.");  
 break;  
 case 4:  
 System.*out*.println("Размечтавшись, " + dreamer + " не заметил, как погрузился в сон.");  
 System.*out*.println("И во сне ему снилось, что он побывал в прошлом.");  
 break;  
 case 5:  
 System.*out*.println("Размечтавшись, " + dreamer + " не заметил, как задремал.");  
 break;  
 }  
  
 hasPrintedDreamInfo = true;  
 }  
 }  
}  
import java.util.Random;  
public class Dreamer {  
 public boolean journeySuccessful;  
 public boolean hasGoneToSunCity;  
 private String name;  
  
 public Dreamer(String name) {  
 this.name = name;  
 this.journeySuccessful = false;  
 this.hasGoneToSunCity = true;  
 this.journeyStarted = true;  
 }  
  
 public void setJourneySuccessful(boolean journeySuccessful) {  
 this.journeySuccessful = journeySuccessful;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
 public int getRandomNumber() {  
 Random random = new Random();  
 return random.nextInt(2);  
 }  
 public void onJourney() {  
 // Выводим изначальный текст  
 System.*out*.println("А наутро Знайка исчез. К завтраку он не явился,");  
 System.*out*.println("а когда коротышки пришли к нему в комнату, они увидели на столе записку,");  
 System.*out*.println("на которой было всего три слова: \"В Солнечный город\", и подпись: \"Знайка\"");  
 boolean journeyStarted = getRandomNumber() == 0;  
 // Выводим дополнительную информацию  
 if (!journeyStarted) {  
 if (getRandomNumber() == 0) {  
 this.hasGoneToSunCity = false;  
 System.*out*.println("Знайка просто решил прогуляться.");  
 }  
 else {  
 this.hasGoneToSunCity = false;  
 System.*out*.println("Знайка исчез без следа.");  
 }  
 } else {  
 // Знайка уже начал путешествие  
 Journey journey = new Journey(this);  
 journey.startJourney(true);  
  
 }  
  
  
 // Определяем результат путешествия  
 this.hasGoneToSunCity();  
 }  
 // Возвращает значение свойства hasGoneToSunCity  
 public boolean hasGoneToSunCity() {  
 return this.hasGoneToSunCity;  
 }  
 public boolean journeyStarted; // указывает, началось ли путешествие  
}

import java.util.List;  
import java.util.Random;  
  
public class Journey {  
  
 enum JourneyType {  
 *GROUND*,  
 *AIR*,  
 *SPACE*,  
  
 }  
  
 private final Dreamer dreamer;  
 private final JourneyType journeyType;  
 private boolean hasGoneToSunCity;  
  
 public Journey(Dreamer dreamer) {  
 this.dreamer = dreamer;  
 this.journeyType = JourneyType.*values*()[dreamer.getRandomNumber()];  
  
 }  
  
 public void startJourney(boolean hasGoneToSunCity) {  
 // Проверяем, отправился ли Знайка уже в Солнечный город  
 if (hasGoneToSunCity) {  
 // Знайка не отправился в Солнечный город  
 // Не выводим текст  
 }  
 else {  
 // Знайка отправился в Солнечный город  
 switch (journeyType) {  
 case *GROUND*:  
 System.*out*.println(dreamer + " отправился в Солнечный город на своем личном автомобиле.");  
 break;  
 case *AIR*:  
 System.*out*.println(dreamer + " отправился в Солнечный город на своем личном самолете.");  
 break;  
 case *SPACE*:  
 System.*out*.println(dreamer + " отправился в Солнечный город на своей личной космической ракете.");  
 break;  
 }  
 // Проводим путешествие  
 // Генерируем случайное значение для переменной successful  
 boolean successful = new Random().nextBoolean();  
  
 // Выводим сообщение об успешности или неудаче путешествия  
 if (successful) {  
 System.*out*.println(dreamer + " успешно добрался до Солнечного города.");  
 }  
 else {  
 System.*out*.println(dreamer + " разбился.");  
 }  
 }  
 // Знайка начал путешествие  
 hasGoneToSunCity = true;  
 }  
}

// Абстрактный класс Person  
public abstract class Person {  
 protected String name;  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
}

// Класс Rocket, представляющий космическую ракету  
class Rocket {  
 @Override  
 public String toString() {  
 return "космическая ракета";  
 }  
}

**Main:**

import java.util.List;  
  
public class Main {  
 public static void main(String[] args) {  
 Dreamer znayka = new Dreamer("Знайка");  
  
 Rocket rocketInDream = new Rocket();  
 String lunarGnomesInDream = "лунные коротышки";  
 List<String> otherInterestingThingsInDream = List.*of*("Еще много разных интересных вещей");  
  
 Journey journey = new Journey(znayka);  
 Dream dream = new Dream(rocketInDream, lunarGnomesInDream, otherInterestingThingsInDream);  
  
 // Вывод текста из Dream  
 dream.startDream(znayka);  
  
 // Вывод текста из Dreamer  
 znayka.hasGoneToSunCity();  
 znayka.onJourney();  
  
 // Вывод текста из Journey  
 if (znayka.hasGoneToSunCity()) {  
 journey.startJourney(false);  
 } else {  
 }  
 }  
}

# Результат работы программы:

Размечтавшись, Знайка не заметил, как погрузился в сон.

И во сне ему снилось, что он выиграл в лотерею.

А наутро Знайка исчез. К завтраку он не явился,

а когда коротышки пришли к нему в комнату, они увидели на столе записку,

на которой было всего три слова: "В Солнечный город", и подпись: "Знайка"

Знайка отправился в Солнечный город на своем личном автомобиле.

Знайка разбился.

# Вывод:

В ходе выполнения лабораторной работы была разработана объектная модель на основе предметной области на основе фрагмента текста. Модель была доработана в соответствии с принципами SOLID, и включала в себя как минимум два интерфейса, один абстрактный класс, один перечисляемый тип и переопределенные методы equals(), hashCode() и toString(). Диаграмма классов была перерисована для отображения внесенных в модель изменений.

# Список литературы.

1. https://se.ifmo.ru/courses/programming
2. <https://javarush.ru/groups/posts/1981-dlja-chego-v-java-nuzhnih-interfeysih>
3. <https://javarush.ru/groups/posts/591-vot-tak-final>
4. <https://www.youtube.com/watch?v=3Xo6zSBgdgk>
5. <https://javarush.ru/groups/posts/1985-raznica-mezhdu-abstraktnihmi-klassami-i-interfeysami>
6. <https://skillbox.ru/media/base/klassy-i-obekty-v-java/>
7. <https://www.jetbrains.com/help/idea/class-diagram.html#analyze_graph>
8. <https://metanit.com/java/tutorial/3.3.php>
9. JAVA from EPAM, 2-е издание, исправленное. И. Н. Блинов, В. С. Романчук
10. <https://javarush.ru/groups/posts/1967-otnoshenija-mezhdu-klassami-nasledovanie-kompozicija-i-agregirovanie->
11. <https://javarush.ru/groups/posts/599-inicializacija-poley>
12. https://javarush.ru/groups/posts/2347-klass-string-v-java