МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
ИТМО»

Лабораторная работа №3-4
по дисциплине “Программирование”

Студент: Студент группы P3132, Юксель Хамза
Преподаватель: Пименов Данила Дмитриевич

Санкт-Петербург
2024

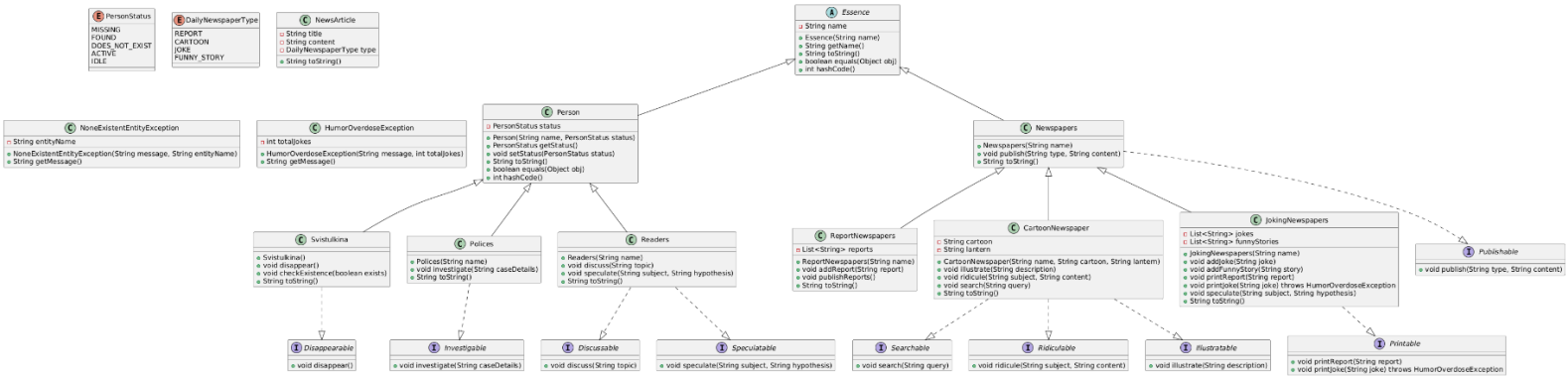
Оглавление

Задание:

В соответствии с выданным вариантом на основе предложенного текстового отрывка из литературного произведения создать объектную модель реального или воображаемого мира, описываемого данным текстом. Должны быть выделены основные персонажи и предметы со свойственным им состоянием и поведением. На основе модели написать программу на языке Java.

Вариант: 7849

В газетах каждый день печатались сообщения о том, что милиционера Свистулькина нигде не могут найти. Некоторые газеты начали подсмеиваться над тем, что сама милиция не может отыскать пропавшего милиционера. В одной газете даже напечатали карикатуру с изображением милиционера, который днем с фонарем сам себя ищет. Кончилось все это тем, что по поводу пропавшего милиционера стали печатать разные шуточки и смешные рассказы и дошли до того, что написали, будто милиционер Свистулькин совсем не исчезал, а найти его не могут потому, что никакого Свистулькина вовсе на свете не было.



Линк для умл-диаграммы : https://shorturl.at/32l4u

Исходный Код Программы:

Classes:

Essence.java:

```
package classes;

abstract class Essence {
    private String name;

    public Essence(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Name: " + name;
    }

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

```
        Essence essence = (Essence) obj;
        return name.equals(essence.name);
    }

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

CartoonNewspapers.java:

```
package classes;

import interfaces.Illustratable;
import interfaces.Ridiculable;
import interfaces.Searchable;

class CartoonNewspapers extends Newspapers implements Illustratable, Ridiculable,
Searchable {
    private String cartoon;
    private String lantern;

    public CartoonNewspapers(String name, String cartoon, String lantern) {
        super(name);
        this.cartoon = cartoon;
        this.lantern = lantern;
    }

    @Override
    public void illustrate(String description) {
        System.out.println(getName() + " illustrates: " + description);
    }

    @Override
    public void ridicule(String subject, String content) {
        System.out.println(getName() + " ridicules " + subject + ": " + content);
    }

    @Override
    public void search(String query) {
        System.out.println(getName() + " is searching for: " + query + " in the cartoon.");
    }

    @Override
    public String toString() {
        return "CartoonNewspaper{" + super.toString() + ", cartoon='" + cartoon + "', lantern='"
+ lantern + "'}";
    }
}
```

JokingNewspapers.java:

```
package classes;

import java.util.ArrayList;
import java.util.List;
import exeptions.HumorOverdoseException;
import interfaces.Printable;
import interfaces.Speculatable;

public class JokingNewspapers extends Newspapers implements Printable, Speculatable {
    private List<String> jokes = new ArrayList<>();
    private List<String> funnyStories = new ArrayList<>();

    public JokingNewspapers(String name) {
        super(name);
    }

    public void addJoke(String joke) {
        jokes.add(joke);
        System.out.println(getName() + " added a joke: " + joke);
    }

    public void addFunnyStory(String story) {
        funnyStories.add(story);
        System.out.println(getName() + " added a funny story: " + story);
    }

    @Override
    public void printReport(String report) {
        throw new UnsupportedOperationException("Reports are not supported in Joking
Newspapers.");
    }

    @Override
    public void printJoke(String joke) throws HumorOverdoseException {
        jokes.add(joke);
        System.out.println(getName() + " published a joke: " + joke);
        if (jokes.size() > 5) {
            throw new HumorOverdoseException("Too many jokes! Readers can't stop laughing.
Total jokes: " + jokes.size());
        }
    }

    @Override
    public void speculate(String subject, String hypothesis) {
        System.out.println(getName() + " speculates that " + subject + ": " + hypothesis);
    }
```

```
    @Override
    public String toString() {
        return "JokingNewspapers{" + super.toString() + ", jokes=" + jokes + ", funnyStories="
+ funnyStories + "}";
    }
}
```

Newspapers.java:

```
package classes;

import interfaces.Publishable;

class Newspapers extends Essence implements Publishable {
    public Newspapers(String name) {
        super(name);
    }

    @Override
    public void publish(String type, String content) {
        System.out.println(getName() + " published a " + type + ": " + content);
    }


    @Override
    public String toString() {
        return "Newspapers{" + super.toString() + "}";
    }
}
```

Person.java:

```
package classes;

import enums.PersonStatus;

class Person extends Essence {
        private PersonStatus status;

        public Person(String name, PersonStatus status) {
            super(name);
            this.status = status;
        }

        public PersonStatus getStatus() {
            return status;
        }
```

```
    public void setStatus(PersonStatus status) {
        this.status = status;
    }

    @Override
    public String toString() {
        return super.toString() + ", status=" + status;
    }

    @Override
    public boolean equals(Object obj) {
        if (!super.equals(obj)) return false;
        Person person = (Person) obj;
        return status == person.status;
    }

    @Override
    public int hashCode() {
        return super.hashCode() * 31 + status.hashCode();
    }
}
```

Polices.java:

```
package classes;

import enums.PersonStatus;
import interfaces.Investigable;

public class Polices extends Person implements Investigable {
    public Polices(String name) {
        super(name, PersonStatus.ACTIVE);
    }

    @Override
    public void investigate(String caseDetails) {
        System.out.println(getName() + " is investigating: " + caseDetails);
    }

    @Override
    public String toString() {
        return "Polices{" + super.toString() + "}";
    }
}
```

Readers.java:

```
package classes;
```

```
import enums.PersonStatus;
import interfaces.Discussable;
import interfaces.Speculatable;

public class Readers extends Person implements Discussable, Speculatable {
    public Readers(String name) {
        super(name, PersonStatus.IDLE);
    }

    @Override
    public void discuss(String topic) {
        System.out.println(getName() + " is discussing: " + topic);
    }

    @Override
    public void speculate(String subject, String hypothesis) {
        System.out.println(getName() + " speculates about " + subject + ": " + hypothesis);
    }


    @Override
    public String toString() {
        return "Readers{" + super.toString() + "}";
    }
}
```

ReportNewspapers.java:

```
package classes;

import java.util.ArrayList;
import java.util.List;

import exeptions.HumorOverdoseException;
import interfaces.Printable;

// Class ReportNewspapers
public class ReportNewspapers extends Newspapers implements Printable {
    private List<String> reports;

    public ReportNewspapers(String name) {
        super(name);
        this.reports = new ArrayList<>();
    }

    public void addReport(String report) {
        reports.add(report);
        System.out.println(getName() + " added a report: " + report);
    }
```

```
    @Override
    public void printReport(String report) {
        System.out.println(getName() + " is publishing reports...");
    }

            @Override
            public void printJoke(String joke) throws HumorOverdoseException {
        throw new UnsupportedOperationException("Jokes are not supported in Report
Newspapers.");

            }
            @Override
            public String toString() {
        return "ReportNewspapers{" + super.toString() + ", reports=" + reports + "}";
    }
}
```

Svistulkina:

```
package classes;

import enums.PersonStatus;
import interfaces.Dissapearable;

public class Svistulkina extends Person implements Dissapearable {
    public Svistulkina() {
        super("Свистулькина", PersonStatus.MISSING);
    }

    @Override
    public void disappear() {
        System.out.println(getName() + " has disappeared!");
    }

    public void checkExistence(boolean exists) {
        if (!exists) {
            setStatus(PersonStatus.DOES_NOT_EXIST);
            System.out.println(getName() + " is now considered non-existent!");
        } else {
            System.out.println(getName() + " is still missing but might exist.");
        }
    }


    @Override
    public String toString() {
        return "Свистулькина{" + super.toString() + "}";
```

```
    }
}
```

Enums:

DailyNewspaperTypes.java:

```
package enums;

public enum DailyNewspaperTypes {
    REPORT,
    CARTOON,
    JOKE,
    FUNNY_STORY
}
```

PersonStatus.java:

```
package enums;

public enum PersonStatus {

    MISSING,
    FOUND,
    DOES_NOT_EXIST,
    ACTIVE,
    IDLE
}
```

Interfaces:

Discussable:

```
package interfaces;

public interface Discussable {
    void discuss(String topic);
}
```

Dissabearable.java:

```
package interfaces;
```

```
public interface Dissapearable {

	void disappear();
}
```

Illustrable.java:

```
package interfaces;

public interface Illustratable {
   void illustrate(String description);
}
```

Investigateable.java:

```
package interfaces;

public interface Investigable {
   void investigate(String caseDetails);
}
```

Printable.java:

```
package interfaces;

import exeptions.*;

public interface Printable {
   void printReport(String report);
   void printJoke(String joke) throws HumorOverdoseException;
}
```

Publishable.java:

```
package interfaces;

public interface Publishable {
   void publish(String type, String content);
}
```

Ridiculable.java:

```
package interfaces;

public interface Ridiculable {
   void ridicule(String subject, String content);
}
```

Searchable.java:

```
package interfaces;


public interface Searchable {
   void search(String query);
}
```

Speculatable.java:

```
package interfaces;

public interface Speculatable {
   void speculate(String subject, String hypothesis);
}
```

Exceptions.java:

HumorOverdoseException.java:

```
package exeptions;

public class HumorOverdoseException extends RuntimeException {
   /**
         *
         */
        private static final long serialVersionUID = 1L;


   public HumorOverdoseException(String message) {
      super(message);
   }

   @Override
   public String getMessage() {
```

```
        return "Warning: " + super.getMessage();
    }
}
```

NoneExcistentEntityException.java:

```
package exeptions;

class NoneExistentEntityException extends Exception {
    /**
            *
            */
           private static final long serialVersionUID = 1L;
           private String entityName;

    public NoneExistentEntityException(String message, String entityName) {
        super(message);
        this.entityName = entityName;
    }

    @Override
    public String getMessage() {
        return "Error: " + super.getMessage() + " Entity: " + entityName;
    }
}
```

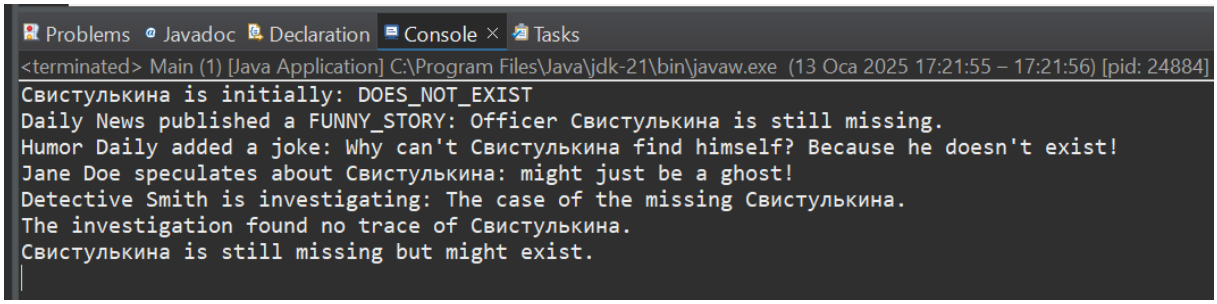
Main:

Main.java:

```
package main;

import classes.*;
import java.util.Random;
import enums.DailyNewspaperTypes;
import enums.PersonStatus;

public class Main {
    public static void main(String[] args) {
        Random random = new Random();

        // Random initial status for Свистулькина
        PersonStatus initialStatus = random.nextBoolean() ? PersonStatus.MISSING :
PersonStatus.DOES_NOT_EXIST;
        Svistulkina officer = new Svistulkina();
        officer.setStatus(initialStatus);
```

```
        System.out.println(officer.getName() + " is initially: " + officer.getStatus());

        // Randomly select a type of newspaper activity
        String[] articles = {
            "Officer Свистулькина is still missing.",
            "The police are unable to locate officer Свистулькина.",
            "Rumors suggest that officer Свистулькина never existed."
        };
        DailyNewspaperTypes[] types = DailyNewspaperTypes.values();
        int randomArticleIndex = random.nextInt(articles.length);
        DailyNewspaperTypes randomType = types[random.nextInt(types.length)];

        ReportNewspapers dailyReport = new ReportNewspapers("Daily News");
        dailyReport.publish(randomType.name(), articles[randomArticleIndex]);

        // Random joke creation
        JokingNewspapers humorPaper = new JokingNewspapers("Humor Daily");
        if (random.nextBoolean()) {
            humorPaper.addJoke("Why can't Свистулькина find himself? Because he doesn't
exist!");
        } else {
            humorPaper.addFunnyStory("Свистулькина was last seen chasing his shadow!");
        }

        // Readers' random reactions
        Readers reader = new Readers("Jane Doe");
        if (random.nextBoolean()) {
            reader.discuss("Some newspapers ridicule the officer's existence.");
        } else {
            reader.speculate("Свистулькина", "might just be a ghost!");
        }

        // Police investigation with random outcomes
        Polices investigator = new Polices("Detective Smith");
        investigator.investigate("The case of the missing Свистулькина.");
        if (random.nextBoolean()) {
            System.out.println("The investigation found no trace of Свистулькина.");
        } else {
            System.out.println("The investigation suggests Свистулькина might still exist.");
        }

        // Update officer's status dynamically
        boolean exists = random.nextBoolean();
        officer.checkExistence(exists);
    }
}
```

Результат Работы Программы:



Выводы по Работе:

Проект демонстрирует хорошо структурированное применение принципов объектно-ориентированного программирования, акцентируя внимание на абстракции, наследовании, полиморфизме и инкапсуляции. Модульная и расширяемая структура классов в сочетании с интерфейсами и перечислениями обеспечивает повторное использование, масштабируемость и эффективное управление состоянием. Упрощение и оптимизация компонентов, таких как класс ReportNewspapers, повышают читаемость и обеспечивают единообразие функциональности, что упрощает сопровождение и расширение кода. Пользовательские исключения и динамические симуляции повышают надежность и удобство тестирования, а использование современных возможностей Java, таких как records, упрощает обработку данных. Четкая структура проекта и наименования способствуют совместной работе и адаптируемости, что делает его пригодным для реальных приложений, таких как системы управления контентом или образовательные симуляции, и обеспечивает прочную основу для дальнейших улучшений.