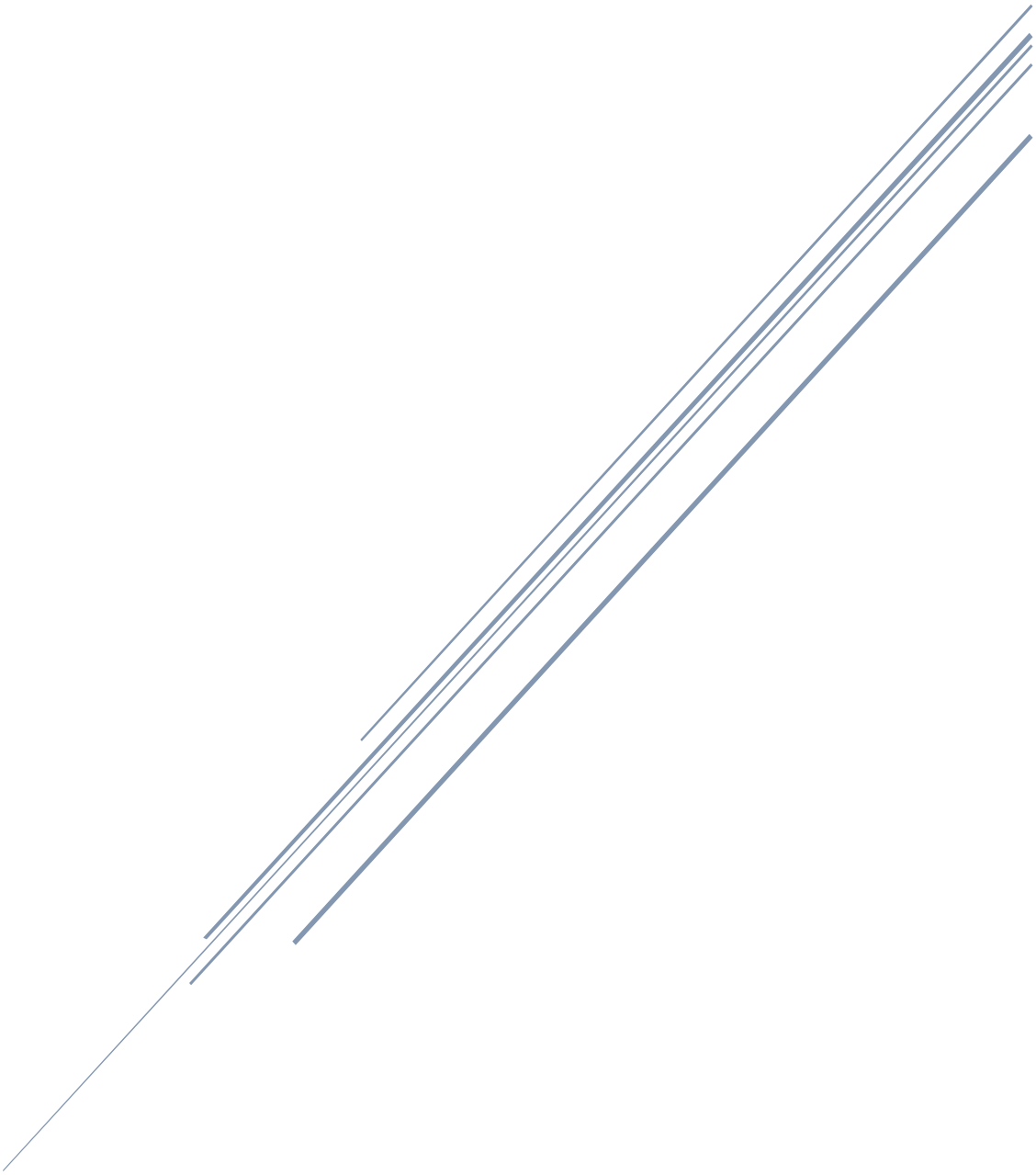
RÁMCOVÉ ZADANIE

Zadanie č.3

STU FIIT

ZOOP



Adrián Vančo

ID: 103171

Použité OOP Princípy

Zadanie 3:

1. **Implementovane funkcionality**
    a. Vytváranie, editovanie, mazanie ,miestnosti pomocou zamestnávateľa, expozičných miest pomocou zamestnanca.
    b. Prijímanie zamestnancov a pridávanie umelcov do galérie pomocou zamestnávateľa.
    c. Vystavovanie diel umelcov do expozičných miest umelcom cez zamestnanca.
    d. Pripísanie ceny poplatku umelcovi cez interface.

2. **Dedenie**
    a. Artist, Employee a Employer dedia od triedy Person.
    b. Artwork dedí od triedy Thing.

```
public class Employer extends Person {
    //upravene pri zadani 3 na finals
//zadanie 3 implementovany interface Responsible
public class Employee extends Person implements Responsible{
public class Artist extends Person implements canRentExposure{

public class Artwork extends Thing{
```

3. Modifikátory prístupu

V rámci projektu využívam zapúzdrenie pomocou modifikátoru prístupu private a protected a sprístupňovanie atribútov pomocou get a set metód.

```
public class Room{
    private String title;
    private ArrayList<ExposurePlace> places = new ArrayList<ExposurePlace>(); //kompozicia
    private Employee responsibleEmployee;
    private boolean haveResponsibleEmployee = false;    //pridane pri zadani 3
```

```
//getery
public String getTitle() {
    return title;
}

public boolean isHaveResponsibleEmployee() {
    return haveResponsibleEmployee;
}

public ArrayList<ExposurePlace> getPlaces() {
    return places;
}


public Employee getResponsibleEmployee() {
    return responsibleEmployee;
}
```

```
//setery
public void setTitle(String title) {
    this.title = title;
}

public void setResponsibleEmployee(Employee employee) {
    this.responsibleEmployee = employee;
}

public void setHaveResponsibleEmployee(boolean haveResponsibleEmployee) {
    this.haveResponsibleEmployee = haveResponsibleEmployee;
}
```

4. Preťažovanie

a. Metódy

Prvá metóda editRoom pre zmenu názvu, druha metóda editRoom pre nastavenie zodpovednej osoby.

```
public  void editRoom(String oldTitle, String newTitle) {
    boolean notExist = true;
    for(Room i: this.GALLERY.getRooms() ) {
        if( oldTitle == i.getTitle() ) {
            System.out.println("Room [" + oldTitle + "] bola premenovana na [" + newTitle + "].\n");
            i.setTitle(newTitle);
            notExist = false;
            break;
        }
    }
    if(notExist) {
    System.out.println("Room [" + oldTitle + "] sa nenachadza v Gallery.\n");
    }
}

//pridane pri zadani 3
public void editRoom(String title, Employee employee) {
    for(Room i: this.GALLERY.getRooms() ) {
        if( title == i.getTitle() ) {
            if(i.isHaveResponsibleEmployee()) {
                i.setResponsibleEmployee(employee);
                System.out.println("V Room [" + title + "] bol zmeneny zodpovedny zamestnanec na [" + employee.getName() +"].");
            } else {
                i.setHaveResponsibleEmployee(true);
                i.setResponsibleEmployee(employee);
                System.out.println("Room [" + title + "] bola  pridelena zamestnancovi [" + employee.getName() +"].");
            }
            break;
        }
    }
}
```

b. Konštruktora

Napríklad v triede Artist

```
public Artist(String name, String idCardNumber, String gender, String styleOfArt, Gallery gallery) {
    super(name, idCardNumber,gender);
    this.styleOfArt = styleOfArt;
    System.out.println("Bol vytvoreny umelec [" + name + "].\n");
    this.gallery = gallery;
}

public Artist(String name, String idCardNumber, int age, String styleOfArt, Gallery gallery) {
    super(name, idCardNumber,age);
    this.styleOfArt = styleOfArt;
    System.out.println("Bol vytvoreny umelec [" + name + "].\n");
    this.gallery = gallery;
}
```

5. Agregácia a Kompozícia

Napríklad v triede Gallery mám agregáciu zamestnancov a umelcov do ArrayListov. Ak zanikne galéria zamestnanci a umelci nezaniknú.

A kompozíciu miestností do ArrayListu, takže ak zanikne galéria zaniknú aj miestnosti.

```
public final class Gallery {
    private ArrayList<Room> rooms = new ArrayList<Room>();  //kompozícia
    protected ArrayList<Employee> employees = new ArrayList<Employee>();    //agregacia
    protected ArrayList<Artist> artists = new ArrayList<Artist>();  //agregacia
    private  String title;
```

6. Asociácia

Trieda Employee používa Room na prístup k expozičným miestam, kde obraz vystaví.

```
public class Employee extends Person implements Responsible{
    //zadanie 3 ID a STARTDATE na final pridany boolean responsible
    private final int ID;
    private int salary;
    private int money = 0;
    private final int STARTDATE;
    private String employmentRelationship;
    private Artist artist;  //asociacia
    private Artwork artworkToAdd;   //asociacia
    private Room managedRoom;   //asociacia
    private boolean responsible = false;
```

7. Final

a. Atribut

```
public class Employer extends Person {
    //upravene pri zadani 3 na finals
    private final int ORGANIZATIONIDENTIFICATIONNUMBER; //IČO (OIN)
    private final String TAXIDENTIFICATIONNUMBER;   //DIČ (TIN)
    private final String VALUEADDEDTAXIDENTIFICATIONNUMBER; //IČ DPH (IN_VAT)
    public final Gallery GALLERY;   //agregacia
```

b. Trieda

```
public final class Gallery {
    private ArrayList<Room> rooms = new ArrayList<Room>();  //kompozícia
    protected ArrayList<Employee> employees = new ArrayList<Employee>();    //agregacia
    protected ArrayList<Artist> artists = new ArrayList<Artist>();  //agregacia
    private  String title;

    private static Gallery gallery = new Gallery("Galeria Umenia"); //singleton pridane pri zadani 3
    private Gallery(String title) {
        this.title = title;
    }
```

8. Abstrakt

a. Trieda

```
public abstract class Person {
    private String name;
    private String gender;
    private String idCardNumber;
    private int age;
```

b. Metóda v Person

```
  public abstract void infoAboutMe();
```

9. Static

Interface Responsible má static metódu na zistenie koľko bude umelec platiť za prenájom za sumu a počet mesiacov, na ktoré si expozičné miesto prenajme.

```
package gallery.persons.staff;

import gallery.persons.Artist;
//zadanie 3 cely interface
public interface Responsible {
    public void  wasExposed(Artist artist,int price);
    static void howMuchTotalRentPrice(int price, int months) {
        if(price <= 0 || months <= 0) {
            System.out.println("Vsetko musi byt kladne a vacsie ako nula\n");
        } else {
            int discount = 0;
            if(months>2 && months<=4) { discount = price / 40;}
            else if(months>4) { discount = price / 25;}
            System.out.println("Vypocitana cena pre prenajom s cenou [" + price + "] na pocet mesiacov ["+months+"]
        }
    }
}
```

10.Interface

a. Responsible pre zamestnancov

```
package gallery.persons.staff;

import gallery.persons.Artist;
//zadanie 3 cely interface
public interface Responsible {
    public void  wasExposed(Artist artist,int price);
    static void howMuchTotalRentPrice(int price, int months) {
        if(price <= 0 || months <= 0) {
            System.out.println("Vsetko musi byt kladne a vacsie ako nula\n");
        } else {
            int discount = 0;
            if(months>2 && months<=4) { discount = price / 40;}
            else if(months>4) { discount = price / 25;}
            System.out.println("Vypocitana cena pre prenajom s cenou [" + price + "] na pocet mesiacov ["+months+"]
        }
    }
}
```

b. canRentExposure

```
public interface canRentExposure {
    public void increaseFee(int fee);
}
```

11.Upcasting

a. V class Employer upcastnem triedu zamestnanec alebo artist do funkcie pre pridanie do galerie.

```
//upravene pri zadani 3
//upcasting
public void addPersonToGallery(Person person){
    this.GALLERY.addPersonToRightArray(person);
}
```

12.Downcasting

a. V gallery sa trieda Person downcastne bud na triedu Artist alebo Employee a podla toho sa zaradí do daného ArrayListu v Gallery

```
        //downcasting   pridane pri zadani 3
public void addPersonToRightArray(Person person) {
    if(person instanceof Employee) {
        employees.add( (Employee) person);
        System.out.println("Employee [" + person.getName() + "] bol pridany do Gallery [" + this.title + "].\n");
    }
    if(person instanceof Artist) {
        artists.add( (Artist) person);
        System.out.println("Artist [" + person.getName() + "] bol pridany do Gallery [" + this.title + "].\n");
    }
}
```

13.Singleton

a. Gallery

```
public final class Gallery {
    private ArrayList<Room> rooms = new ArrayList<Room>();  //kompozícia
    protected ArrayList<Employee> employees = new ArrayList<Employee>();    //agregacia
    protected ArrayList<Artist> artists = new ArrayList<Artist>();  //agregacia
    private  String title;

    private static Gallery gallery = new Gallery("Galeria Umenia"); //singleton pridane pri zadani 3
    private Gallery(String title) {
        this.title = title;
    }
```

b. Gallery System

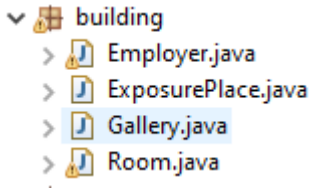
```
public class GallerySystem {

private static GallerySystem GallerySystem = new GallerySystem(); //singleton
    private GallerySystem() {
    }
```

Zo zadania 2:

1. **Dedenie**
    a. Artist, Employee a Employer dedia od triedy Person
    b. Artwork dedi od triedy Thing

2. **Zapuzdrenie**
    a. V rámci projektu som využil v class Gallery pre ArrayListy modifikátory prístupu **private** a cez protected metódy môže upravovať polia Employer. Artist a Employee nemôže pridávať ani mazať, môžu iba čítať. Tento typ zapuzdrenia planujem využiť aj v ďalších prípadoch




```java
public class Gallery {
    private ArrayList<Room> rooms = new ArrayList<Room>();  //kompozícia
    private ArrayList<Employee> employees = new ArrayList<Employee>();  //agregacia
    private ArrayList<Artist> artists = new ArrayList<Artist>();    //agregacia
    private String title;

    public Gallery(String title) {
        this.title = title;
    }



    //add
    protected void addEmployee(Employee employee) {
        employees.add(employee);
    }

    protected void addRoom(String title) {
        this.rooms.add(new Room(title));
    }

    protected void addArtist(Artist artist) {
        artists.add(artist);
    }



    //remove
    protected void removeEmployee(Employee employee) {
        employees.remove(employee);
    }

    protected void removeArtist(Artist artist) {
        artists.remove(artist);
    }
```

3. Preťaženie

a. V triede Person a Artist som využil preťaženie konštruktor a ďalej plánujem aj preťaženie metódy.

```java
//uviedol minimum udajov a pohlavie
public Person(String name, String surName, String idCardNumber, String gender){
    this.name = name;
    this.surName = surName;
    this.idCardNumber = idCardNumber;
    this.gender = gender;
}
//uviedol minimum udajov a rok (overloading s predchadzajucim)
public Person(String name, String surName, String idCardNumber, int age){
    this.name = name;
    this.surName = surName;
    this.idCardNumber = idCardNumber;
    this.age = age;
}
```

4. Prekonanie

a. V triede Person mám metódu infoAboutMe() a v Triedach Employer a Artist je prekonanie tejto metódy.

ery.java | Employer.java | Employee.java | Artist.java | Person.java | Artwork.java | ExposurePlace.java | Room.jav

```java
public class Person {
    private String name;
    private String surName;
    private String gender;
    private String idCardNumber;
    private int age;

    //uviedol minimum udajov
    public Person(String name, S
        this.name = name;
        this.surName = surName;
        this.idCardNumber = idCa
    }

    //uviedol minimum udajov a p
    public Person(String name, S
        this.name = name;
        this.surName = surName;
        this.idCardNumber = idCa
        this.gender = gender;
    }
    //uviedol minimum udajov a r
    public Person(String name, S
        this.name = name;
        this.surName = surName;
        this.idCardNumber = idCa
        this.age = age;
    }
    //uviedol vsetky osobne udaj
    public Person(String name, S
        this.name = name;
        this.surName = surName;
        this.idCardNumber = idCa
        this.gender = gender;
        this.age = age;
    }

    //predstavenie
    public void infoAboutMe() {
        System.out.println("Ahoj volám sa " + this.getName() + " " + this.getSurName());
    }
```

```java
package gallery.persons;

import java.util.ArrayList;

public class Artist extends Person {
    private int exhibitionFee;
    private String styleOfArt;
    private ArrayList<Artwork> works = new ArrayList<Artwork>();    //kompozicia
    private String aboutTheAuthor;
    private ArrayList<ExposurePlace> expositions = new ArrayList<ExposurePlace>();

    public Artist(String name, String surName, String idCardNumber, String styleOfArt) {
        super(name, surName, idCardNumber);
        this.styleOfArt = styleOfArt;
    }

    public Artist(String name, String surName, String idCardNumber, String gender, String styleOfArt) {
        super(name, surName, idCardNumber,gender);
        this.styleOfArt = styleOfArt;
    }

    public Artist(String name, String surName, String idCardNumber, int age, String styleOfArt) {
        super(name, surName, idCardNumber,age);
        this.styleOfArt = styleOfArt;
    }

    //predstavenie (overriding)
    public void infoAboutMe() {
        System.out.println("Ahoj volám sa " + this.getName() + " " + this.getSurName() + " a zaoberám sa o štýl " +this.getStyleOfArt());
    }
```

5. **Asociácia**
    a. Ešte nespravené metóda na asociáciu ale treda Artist pužije triedu Employee na vystavenie svojho obrazu a triedu Fee na poslanie peňazí za prenajom na účet.
6. **Agregácia**
    a. Trieda Gallery má ArrayList pre Employee a pre Artist ak trieda Gallery zanikne zamestnanci a umelci prežijú pretože ich posielam ako hotový objekt do galérie.

```java
import java.util.ArrayList;

public class Gallery {
    private ArrayList<Room> rooms = new ArrayList<Room>();  //kompozícia
    private ArrayList<Employee> employees = new ArrayList<Employee>();  //agregacia
    private ArrayList<Artist> artists = new ArrayList<Artist>();    //agregacia
    private String title;

    public Gallery(String title) {
        this.title = title;
    }

    //add
    protected void addEmployee(Employee employee) {
        employees.add(employee);
    }

    protected void addRoom(String title) {
        this.rooms.add(new Room(title));
    }

    protected void addArtist(Artist artist) {
        artists.add(artist);
    }


    //remove
    protected void removeEmployee(Employee employee) {
        employees.remove(employee);
    }

    protected void removeArtist(Artist artist) {
        artists.remove(artist);
    }
```

7. Kompozícia

a. Trieda Gallery má ArrayList pre Room ak galéria zanikne zaniknú aj miestnosti pretože miestnosti vytváram v galérii. A v Triede Room mám zas ArrayList pre triedu ExpositionPlace kde po zániku miestnosti zaniknú aj expozičné miesta.

```java
import java.util.ArrayList;

public class Gallery {
    private ArrayList<Room> rooms = new ArrayList<Room>();  //kompozícia
    private ArrayList<Employee> employees = new ArrayList<Employee>();  //agregacia
    private ArrayList<Artist> artists = new ArrayList<Artist>();    //agregacia
    private String title;

    public Gallery(String title) {
        this.title = title;
    }


    //add
    protected void addEmployee(Employee employee) {
        employees.add(employee);
    }

    protected void addRoom(String title) {
        this.rooms.add(new Room(title));
    }

    protected void addArtist(Artist artist) {
        artists.add(artist);
    }



    //remove
    protected void removeEmployee(Employee employee) {
        employees.remove(employee);
    }

    protected void removeArtist(Artist artist) {
        artists.remove(artist);
    }
```