Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине «Современные платформы программирования»

Выполнила:

Студентка 3 курса

Группы ПО-3

Гаврилкович Е.В

Проверил:

Крощенко А.А.

Брест 2020 г.

Цель работы:

Приобрести практические навыки в области объектно-ориентированного проектирования

Ход работы

Задание 1:
Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:
interface Здание ← abstract class Общественное Здание ← class Театр.

Текст программы:

```
import java.util.*;

public class task1{
    public static void main(String[] args) {

        Theatre theatre = new Theatre();
        theatre.getHeight();
        theatre.getPrice();
        theatre.getCount();

    }
}

interface Building{
    void getHeight();
    void getPrice();
}

abstract class PublicBuilding implements Building{

    public void getHeight(){
        System.out.println("Height: 100");
    }

    public void getPrice(){
        System.out.println("Price: 10000$");
    }

    public void getCount(){
        System.out.print("Count: 4000");
    }
}

class Theatre extends PublicBuilding{
```

```
    public Theatre(){
        System.out.println("Theatre!");
    }
}
```

Вывод:

```
Theatre!

Height: 100

Price: 10000$
Count: 4000
```

Задание 2:

В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

Текст программы:

```
import java.util.ArrayList;

public class task2{
    public static void main(String[] args) {
        ArrayList<Worker> workers = new ArrayList<>();

        Manager manager = new Manager("Dima", 30, 3);
        workers.add(manager);
        manager.sayHello();
        manager.getSalary();

        Analyst analyst = new Analyst("Elena", 31, 7);
        workers.add(analyst);
        analyst.sayHello();
        analyst.getSalary();
        analyst.getCountOfWorkingHours();

        Programmer programmer = new Programmer("QWEERTY");
```

```
        programmer.sayHello();
        programmer.getSalary();
        programmer.getCurrentProjectName();

        Tester tester = new Tester("DFG", 6);
        tester.sayHello();
        tester.getSalary();
        tester.getCurrentProjectName();

    }
}

interface Worker{
    void getSalary();
    void sayHello();
}


class Manager implements Worker{

    String position;
    int salary;
    String name;
    int age;
    int experience;
    int bonus;

    public Manager(String name, int age, int experience){
        this.name = name;
        this.age = age;
        this.experience = experience;
        this.salary = 200;
        this.position = "manager";
        this.bonus = 100;
    }


    public void getSalary() {
        System.out.println("My salary is " + (salary + bonus));
    }

    public void sayHello(){
        System.out.println("Hello! My name is " + name +" .I am a " + position + "! " + "I'm " + ag
e + ".");
    }

}
```

```
 class Analyst implements Worker{

    String position;
    int salary;
    String name;
    int age;
    int experience;
    int bonus = 250;

    public Analyst(String name, int age, int experience){
        this.name = name;
        this.age = age;
        this.experience = experience;
        this.salary = 250;
        this.position = "analyst";
        this.bonus = 250;
    }


    public void getSalary() {
        System.out.println("My salary is " + (salary + bonus));
    }

    public void sayHello(){
        System.out.println("Hello! My name is " + name +" .I am a " + position + "! " + "I'm " + ag
e + ". My experience is " + experience + " years.");
    }

    public void getCountOfWorkingHours(){
       System.out.println("My working hours: " + (experience > 5 ? 8 : 10));
    }

}

 class Programmer implements Worker{

    String position;
    int salary;
    String projectName;
    int bonus;

    public Programmer(String projectName){
       this.projectName = projectName;
        this.salary = 1000;
        this.position = "programmer";
        this.bonus = 500;
    }
```

```
    public void getSalary() {
        System.out.println("My salary is " + (salary + bonus));
    }


    public void getCurrentProjectName(){
       System.out.println("My current project is " + projectName);
    }

    public void sayHello(){
        System.out.println("Hello! I am a " + position + "!");
    }

}

 class Tester implements Worker{

    String position;
    int salary;
    int experience;
    String projectName;
    int bonus;

    public Tester(String projectName, int experience){
        this.projectName = projectName;
        this.experience = experience;
        this.salary = 800;
        this.position = "tester";
        this.bonus = 700;
    }


    public void getSalary() {
        System.out.println("My salary is " + (salary + bonus));
    }

    public void sayHello(){
        System.out.println("Hello! I am a " + position + "! My experience is " + experience  + " y
ears.");
    }

    public void getCurrentProjectName(){
       System.out.println("My current project is " + projectName);
    }

}
```

Вывод:

```
Hello! My name is Dima .I am a manager! I'm 30.
My salary is 300
Hello! My name is Elena .I am a analyst! I'm 31. My experience is 7 years.
My salary is 500
My working hours: 8
Hello! I am a programmer!
My salary is 1500
My current project is QWEERTY
Hello! I am a tester! My experience is 6 years.
My salary is 1500
My current project is DFG
```

Задание 3:

В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Текст программы:

```
import java.util.*;
import java.text.*;

public class task3 {

    public static void main(String[] args) {

        String[] bookNames = { "Master and Margarita", "Dead Souls", "Anna Karenina" };

        Library library = new Library();

        //create books

        Book[] books = { new Book(bookNames[0], "Bulgakov", 3), new Book(bookNames[1], "
Gogol", 5),
                new Book(bookNames[2], "Tolstoy", 1) };

        //create readers

        Reader[] readers = { new Reader("Ivan Ivanov", "24.05.2019"), new Reader("Ivan Petro
v", "04.12.2019"),
                    new Reader("Ivan Sergeev", "01.12.2019") , new Reader("Ivan Semenov", "05.1
2.2019")};

        //add books in arrayList
```

```
        for (Book book : books) {
            library.addBook(book);
        }
        //add readers in arrayList

        for (Reader reader : readers) {
            library.addReader(reader);
        }

        library.showAll();

        // System.out.println("\nTry to add Bulgakov to Ivan Ivanov\n");

       library.createOrder(readers[0], books[0]);

    //    System.out.println("\nTry to add Gogol to Ivan Ivanov\n");

       library.createOrder(readers[0], books[1]);


    //    System.out.println("\nTry to add Tolstoi to Ivan Petrov\n");

       library.createOrder(readers[1], books[2]);

    //    System.out.println("\nTry to add Tolstoi to Ivan Sergeev\n");

       library.createOrder(readers[2], books[2]);

    //    System.out.println("\nTry to add Gogol to Ivan Sergeev\n");

       library.createOrder(readers[2], books[1]);

    //    System.out.println("\nTry to add Tolstoi to Ivan Semenov\n");

        library.createOrder(readers[3], books[2]);

      for(Reader reader: readers){
            System.out.println("Reader name: " + reader.getName()
                        + "\nDate: " + reader.getDate() + "\n");
            reader.showAll();
      }

      library.showBlackList();

    }
}

interface Show{
```

```
    void showAll();
}

class Library implements Show{

    private ArrayList<Book> books = new ArrayList<>();
    private ArrayList<Reader> readers = new ArrayList<>();
    private ArrayList<Reader> blackList = new ArrayList<>();
    private ArrayList<Order> orders = new ArrayList<>();

    public void addBook(Book book) {
        books.add(book);
    }

    public void createOrder(Reader reader, Book book) {

        if(reader.checkDate(new Date(), reader.getDate()) > 20){
            blackList.add(reader);
            return;
        }

        for(Book currentBook : books){

            if(currentBook.getId() == book.getId() && currentBook.getNumber() > 0){
                Order order = new Order(book.getId(), reader.getName(), book.getTitle());
                orders.add(order);
                order.showAll();
                removeBook(currentBook);
                reader.addBook(currentBook);
                removeOrder(order);
            }

        }


    }

    public void showBlackList(){
        System.out.println("\nBlack list\n");
        for(Reader reader: blackList){
            System.out.println("\nReader name: " + reader.getName()
                    + "\nDate: " + reader.getDate());
        }
    }

    public void showAll(){
        System.out.println("\nAll books in the library:\n");
        for(Book book: books){
```

```
            System.out.println("\nBooks title: " + book.getTitle()
                    + "\nAuthor: " + book.getAuthor()
                    + "\nCount: " + book.getNumber() + "\n");
        }
    }

    public void removeBook(Book book){
        book.setNumber(book.getNumber()-1);

    }

    public void removeOrder(Order order){
        orders.remove(order);
    }

    public void addReader(Reader reader){
        readers.add(reader);
    }
}

class Book {
    private int id;
    private String title;
    private String author;
    private int number;

    private static int booksCount = 1;

    public Book(String title, String author, int number) {
        id = booksCount++;

        setTitle(title);
        setAuthor(author);
        setNumber(number);
    }

    public int getId() {
        return id;
    }

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }
```

```
    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

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

class Reader implements Show{

    private int id;
    private String name;
    private Date bookDate = null;
    private ArrayList<Book>books = new ArrayList<>();

    private static int readersCount = 1;

    public Reader(String name, String date) {
        id = readersCount++;

        setName(name);
        setDate(date);
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setDate(String date){
        SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy");
        try{
```

```
            bookDate = format.parse(date);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public Date getDate(){
        return bookDate;
    }

    public long checkDate(Date date, Date bookDate){
        long difference = date.getTime() - bookDate.getTime();
        long days =  difference / (24 * 60 * 60 * 1000);
        return days;
    }

    public void addBook(Book book){
        books.add(book);
    }

    public void showAll(){
        for(Book book: books){
            if(books.size() == 0) {System.out.println("There is not any books");}
            else {
              System.out.println("Book name: " + book.getTitle() + "\n" +
                    "Author: " + book.getAuthor() + "\n");
            }
        }
    }
}

class Order implements Show{
    private int id;
    private int bookId;
    private String readerName;
    private String bookTitle;

    private static int ordersCount = 1;

    public Order(int bookId, String readerName, String bookTitle) {
        id = ordersCount++;

        setBookId(bookId);
        setReaderName(readerName);
        setBookTitle(bookTitle);
    }

    public String getReaderName() {
```

```
        return readerName;
    }

    public void setReaderName(String readerName) {
        this.readerName = readerName;
    }

    public int getBookId() {
        return bookId;
    }

    public void setBookId(int bookId) {
        this.bookId = bookId;
    }
    public void setBookTitle(String bookTitle){
        this.bookTitle = bookTitle;
    }
    public String getBookTitle(){
       return bookTitle;
    }
    public void showAll(){
        System.out.println("Try to add " + bookTitle + " to" + readerName);
    }
}
```

Вывод:

All books in the library:

Books title: Master and Margarita
Author: Bulgakov
Count: 3

Books title: Dead Souls
Author: Gogol
Count: 5

Books title: Anna Karenina
Author: Tolstoy
Count: 1

Reader name: Ivan Ivanov
Date: Fri May 24 00:00:00 MSK 2019

Reader name: Ivan Petrov

Date: Wed Dec 04 00:00:00 MSK 2019

Reader name: Ivan Sergeev
Date: Sun Dec 01 00:00:00 MSK 2019

Reader name: Ivan Semenov
Date: Thu Dec 05 00:00:00 MSK 2019

Black list

Reader name: Ivan Ivanov
Date: Fri May 24 00:00:00 MSK 2019

Reader name: Ivan Ivanov
Date: Fri May 24 00:00:00 MSK 2019

Reader name: Ivan Petrov
Date: Wed Dec 04 00:00:00 MSK 2019

Reader name: Ivan Sergeev
Date: Sun Dec 01 00:00:00 MSK 2019

Reader name: Ivan Sergeev
Date: Sun Dec 01 00:00:00 MSK 2019

Reader name: Ivan Semenov
Date: Thu Dec 05 00:00:00 MSK 2019

Закрепили базовые знания языка программирования Java при решении практических задач.