Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

по дисциплине “Современные платформы программирования”

Выполнил:

Студент 3 курса

Группы ПО-8

Бондаренко К.А.

Проверил:

Крощенко А.А.

Брест 2024

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования.

Ход работы

Вариант 1

Задание 1. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

Абстрактный класс Книга (Шифр, Автор, Название, Год, Издательство). Подклассы Справочник и Энциклопедия.

Код программы:

Класс Книга:

```
import java.util.Objects;

public abstract class Book {
   protected int cypher;
   protected String author;
   protected String name;
   protected int year;
   protected String publisher;

   public Book(int cypher, String author, String name, int year, String publisher) {
      this.cypher = cypher;
      this.author = author;
      this.name = name;
      this.year = year;
      this.publisher = publisher;
   }

   public abstract void printInfo();
}
```

Класс Справочник:

```
import java.util.Objects;

public class Handbook extends Book {
   private int numberOfSections;

   public Handbook(int cypher, String author, String name, int year, String publishingHouse, int
numberOfSections) {
      super(cypher, author, name, year, publishingHouse);
      this.numberOfSections = numberOfSections;
   }

   @Override
   public void printInfo() {
```

```
        System.out.println("Справочник:\nШифр: " + super.cypher
                + "\nАвтор: " + super.author
                + "\nНазвание: " + super.name
                + "\nГод: " + super.year
                + "\nИздательство: " + super.publisher
                + "\nКоличество разделов: " + numberOfSections);

    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Handbook handbook)) return false;
        if (!super.equals(o)) return false;
        return numberOfSections == handbook.numberOfSections;
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), numberOfSections);
    }
}
```

Класс Энциклопедия:

```
import java.util.Objects;

public class Encyclopedia extends Book {
    private int numberOfVolumes;

    public Encyclopedia(int cypher, String author, String name, int year, String publishingHouse, int
numberOfVolumes) {
        super(cypher, author, name, year, publishingHouse);
        this.numberOfVolumes = numberOfVolumes;
    }

    @Override
    public void printInfo() {
        System.out.println("Энциклопедия:\nШифр: " + super.cypher
                + "\nАвтор: " + super.author
                + "\nНазвание: " + super.name
                + "\nГод: " + super.year
                + "\nИздательство: " + super.publisher
                + "\nКоличество томов: " + numberOfVolumes);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Encyclopedia that)) return false;
        if (!super.equals(o)) return false;
        return numberOfVolumes == that.numberOfVolumes;
    }
```

```
    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), numberOfVolumes);
    }
}
```

Класс с методом main:

```
public class Main {
    public static void main(String[] args) {
        Handbook handbook = new Handbook(456, "Марк Яковлевич Выгодский", "Справочник по высшей математике", 2022, "АСТ", 10);
        Encyclopedia encyclopedia = new Encyclopedia(123, "Евгений Яковлевич Гик", "Большая энциклопедия спорта", 2008, "ОлмаМедиаГрупп", 2);

        System.out.println("Информация о справочнике:");
        handbook.printInfo();

        System.out.println("\nИнформация об энциклопедии:");
        encyclopedia.printInfo();
    }
}
```

Результаты работы программы:

```
Информация о справочнике:
Справочник:
Шифр: 456
Автор: Марк Яковлевич Выгодский
Название: Справочник по высшей математике
Год: 2022
Издательство: АСТ
Количество разделов: 10
```

```
Информация об энциклопедии:
Энциклопедия:
Шифр: 123
Автор: Евгений Яковлевич Гик
Название: Большая энциклопедия спорта
Год: 2008
Издательство: ОлмаМедиаГрупп
Количество томов: 2
```

Задание 2. 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.

Создать суперкласс Транспортное средство и подклассы Автомобиль, Велосипед, Повозка. Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

Код программы:

Класс Транспортное средство:

```
public abstract class Vehicle {
    protected String model;
    protected String sign;
    protected int maxSpeed;
    protected int passengersCapacity;
    protected int cargoCapacity;
    protected int currentSpeed;
    protected int currentPassengers;
    protected int currentCargo;
    protected double passengersCoefficient;
    protected double cargoCoefficient;

    public Vehicle(String model, String sign, int maxSpeed, int passengersCapacity, int cargoCapacity) {
        this.model = model;
        this.sign = sign;
        this.maxSpeed = maxSpeed;
        this.passengersCapacity = passengersCapacity;
        this.cargoCapacity = cargoCapacity;
        currentSpeed = 0;
        currentPassengers = 0;
        currentCargo = 0;
    }

    public String getModel() { return model; }

    public void setModel(String model) { this.model = model; }

    public String getSign() { return sign; }

    public void setSign(String sign) { this.sign = sign; }

    public int getCurrentSpeed() { return currentSpeed; }

    public void increaseSpeed(int increment) {
        currentSpeed += increment;
        if (currentSpeed > maxSpeed) {
            currentSpeed = maxSpeed;
        }
    }

    public void decreaseSpeed(int decrement) {
```

```
        currentSpeed -= decrement;
        if (currentSpeed < 0) {
            currentSpeed = 0;
        }
    }

    public void setCurrentPassengers(int passengers) {
        if (passengers > passengersCapacity) {
            currentPassengers = passengersCapacity;
            System.out.println("Недостаточно мест, смогли сесть только " + currentPassengers + "
пассажир(ов/а)");
        }
        currentPassengers = passengers;
        maxSpeed -= (int) (currentPassengers * passengersCoefficient);
    }

    public void setCurrentCargo(int cargo) {
        if (cargo > cargoCapacity) {
            currentCargo = cargoCapacity;
            System.out.println("Перевес, смогло уместиться только " + currentCargo +  "кг");
        }
        currentCargo = cargo;
        maxSpeed -= (int) (currentCargo * cargoCoefficient);
    }

    public abstract double countTransportationTime(int distance);
    public abstract double countTransportationCost(int distance);
}
```

Класс Автомобиль:

```
public class Car extends Vehicle {
    private int mileage;
    private double fuelConsumption;

    public Car(String model, String sign, int maxSpeed, int passengersCapacity, int cargoCapacity, double
fuelConsumption) {
        super(model, sign, maxSpeed, passengersCapacity, cargoCapacity);
        this.mileage = 0;
        this.fuelConsumption = fuelConsumption;
        super.passengersCoefficient = 0.07;
        super.cargoCoefficient = 0.001;
    }

    @Override
    public double countTransportationTime(int distance) {
        if (super.maxSpeed < 0) {
            System.out.println("Автомобиль " + super.model + " " + super.sign + " не может сдвинуться из-
за перевеса");
            return -1;
        }
        if (currentSpeed == 0) {
            System.out.println("Автомобиль " + super.model + " " + super.sign + "стоит на месте");
            return -1;
```

```
        }
        mileage += distance;
        return (double) distance / currentSpeed;
    }

    @Override
    public double countTransportationCost(int distance) {
        if (super.maxSpeed < 0) {
            System.out.println("Автомобиль " + super.model + " " + super.sign + " не может сдвинуться из-
за перевеса");
            return -1;
        }
        if (currentSpeed == 0) {
            System.out.println("Автомобиль " + super.model + " " + super.sign + "стоит на месте");
            return -1;
        }
        return (double) distance / currentSpeed * (fuelConsumption + super.currentPassengers *
passengersCoefficient + super.currentCargo * cargoCoefficient) * Main.fuelCost;
    }
}
```

Класс Велосипед:

```
public class Bicycle extends Vehicle {
    private int diskRadius;

    public Bicycle(String model, String sign, int maxSpeed, int passengersCapacity, int cargoCapacity, int
diskRadius) {
        super(model, sign, maxSpeed, passengersCapacity, cargoCapacity);
        this.diskRadius = diskRadius;
        super.passengersCoefficient = 5;
        super.cargoCoefficient = 0.1;
    }

    @Override
    public double countTransportationTime(int distance) {
        if (super.maxSpeed < 0) {
            System.out.println("Велосипед " + super.model + " " + super.sign + " не может сдвинуться из-за
перевеса");
            return -1;
        }
        if (currentSpeed == 0) {
            System.out.println("Велосипед " + super.model + " " + super.sign + "стоит на месте");
            return -1;
        }
        return (double) distance / (currentSpeed * (1.0 - (double) diskRadius / 100));
    }

    @Override
    public double countTransportationCost(int distance) {
        if (super.maxSpeed < 0) {
            System.out.println("Велосипед " + super.model + " " + super.sign + " не может сдвинуться из-за
перевеса");
            return -1;
```

```
        }
        if (currentSpeed == 0) {
            System.out.println("Велосипед " + super.model + " " + super.sign + "стоит на месте");
            return -1;
        }
        return 0;
    }
}
```

Класс Повозка:

```
public class Wagon extends Vehicle {
    private int horsesAmount;
    private int horsesSpeed;
    private final double baseFare = 5;

    public Wagon(String model, String sign, int passengersCapacity, int cargoCapacity, int horsesAmount,
int horsesSpeed) {
        super(model, sign, horsesAmount * horsesSpeed, passengersCapacity, cargoCapacity);
        this.horsesAmount = horsesAmount;
        this.horsesSpeed = 10;
        super.passengersCoefficient = 0.07;
        super.cargoCoefficient = 0.1;
    }

    @Override
    public double countTransportationTime(int distance) {
        if (super.maxSpeed < 0) {
            System.out.println("Повозка " + super.model + " " + super.sign + " не может сдвинуться из-за
перевеса");
            return -1;
        }
        if (super.currentSpeed == 0) {
            System.out.println("Повозка " + super.model + " " + super.sign + "стоит на месте");
            return -1;
        }
        return (double) distance / currentSpeed;
    }

    @Override
    public double countTransportationCost(int distance) {
        if (super.maxSpeed < 0) {
            System.out.println("Повозка " + super.model + " " + super.sign + " не может сдвинуться из-за
перевеса");
            return -1;
        }
        if (currentSpeed == 0) {
            System.out.println("Автомобиль " + super.model + " " + super.sign + "стоит на месте");
            return -1;
        }
        return baseFare + super.currentPassengers + super.currentCargo * cargoCoefficient;
    }
}
```

Класс с методом main:

```
public class Main {
    public static final double fuelCost = 2.36;

    public static void main(String[] args) {
        Vehicle[] vehicles = new Vehicle[3];

        vehicles[0] = new Car("Porsche Caynne", "ABC123", 250, 4, 1500, 15.5);
        vehicles[1] = new Bicycle("Aeroad", "XYZ456", 50, 1, 30, 21);
        vehicles[2] = new Wagon("HorseCart", "DEF789", 3, 200, 3, 10);

        vehicles[0].setCurrentPassengers(3);
        vehicles[0].setCurrentCargo(500);
        vehicles[0].increaseSpeed(90);
        if (vehicles[0].countTransportationTime(100) >= 0) {
            System.out.println("Автомобиль " + vehicles[0].getModel() + " " + vehicles[0].getSign() + "
проедет 100 км за время "
                    + String.format("%,.2f", vehicles[0].countTransportationTime(100)) + " ч, стоимость "
                    + String.format("%.2f", vehicles[0].countTransportationCost(100)) + " у.е.");
        }

        vehicles[1].setCurrentPassengers(0);
        vehicles[1].setCurrentCargo(30);
        vehicles[1].increaseSpeed(30);
        if (vehicles[1].countTransportationTime(50) >= 0) {
            System.out.println("Велосипед " + vehicles[1].getModel() + " " + vehicles[1].getSign() + "
проедет 50 км за время "
                    + String.format("%,.2f", vehicles[1].countTransportationTime(50)) + " ч, стоимость "
                    + String.format("%.2f", vehicles[1].countTransportationCost(50)) + " у.е.");
        }

        vehicles[2].setCurrentPassengers(4);
        vehicles[2].setCurrentCargo(100);
        vehicles[2].increaseSpeed(20);
        if (vehicles[1].countTransportationTime(35) >= 0) {
            System.out.println("Велосипед " + vehicles[2].getModel() + " " + vehicles[2].getSign() + "
проедет 35 км за время "
                    + String.format("%,.2f", vehicles[2].countTransportationTime(35)) + " ч, стоимость "
                    + String.format("%.2f", vehicles[2].countTransportationCost(35)) + " у.е.");
        }
    }
}
```

Результаты работы программы:

```
Автомобиль Porsche Caynne ABC123 проедет 100 км за время 1,11 ч, стоимость 42,51 у.е.
Велосипед Aeroad XYZ456 проедет 50 км за время 2,11 ч, стоимость 0,00 у.е.
Недостаточно мест, смогли сесть только 3 пассажир(ов/а)
Велосипед HorseCart DEF789 проедет 35 км за время 1,75 ч, стоимость 19,00 у.е.
```

Задание 3. В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

Система Факультет. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

Решение: добавим абстрактный класс Человек, от него создадим подклассы Студент и Преподаватель.

Код программы:

Класс Факультет:

```
import java.util.ArrayList;

public class Faculty {
    private String name;
    private ArrayList<Course> courses;

    public Faculty(String name) {
        this.name = name;
        courses = new ArrayList<>();
    }

    public String getName() { return name; }

    public void addCourse(Course course) {
        courses.add(course);
    }

    public void removeCourse(Course course) {
        courses.remove(course);
    }

    public void setCourses(ArrayList<Course> courses) {
        this.courses = courses;
    }

    public ArrayList<Course> getCourses() {
        return courses;
    }
}
```

Класс Человек:

```
public abstract class Person {
    protected String name;

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

    public String getName() { return name; }

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

Класс Преподаватель:

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class Teacher extends Person {
    private ArrayList<Course> courses;

    public Teacher(String name) {
        super(name);
        courses = new ArrayList<>();
    }

    public void announceRegistryForCourse(Course course) {
        courses.add(course);
        course.setTeacher(this);
        System.out.println("Прием на курс " + course.getName());
    }

    public void stopTeachingCourse(Course course) {
        System.out.println("Завершение курса " + course.getName());
        Scanner scanner = new Scanner(System.in);

        ArrayList<Student> students = course.getStudents();
        Iterator<Student> iterator = students.iterator();
        while (iterator.hasNext()) {
            Student student = iterator.next();
            System.out.println("Поставьте оценку студенту " + student.getName() + " по предмету " +
course.getName());
            Grade grade = new Grade(scanner.nextInt());
            Archive.addRecord(student, course, grade);
            iterator.remove();
            student.stopStudying(course);
        }
        courses.remove(course);
    }
}
```

Класс Курс:

```
import java.util.ArrayList;

public class Course {
    private String name;
    private Teacher teacher;
    private ArrayList<Student> students;

    public Course(String name) {
        this.name = name;
        students = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public void setTeacher(Teacher teacher) {
        this.teacher = teacher;
    }

    public ArrayList<Student> getStudents() {
        return students;
    }

    public void enrollStudent(Student student) {
        students.add(student);
        student.startStudying(this);
    }

    public void excludeStudent(Student student) {
        students.remove(student);
        student.stopStudying(this);
    }
}
```

Класс Студент:

```
public class Student extends Person {
    public Student(String name) {
        super(name);
    }

    public void startStudying(Course course) {
        System.out.println("Студент " + super.name + " начал изучать курс " + course.getName());
    }

    public void stopStudying(Course course) {
        System.out.println("Студент " + super.name + " закончил изучать курс " + course.getName());
    }
}
```

Класс Оценка:

```
public class Grade {
    private int value;

    public Grade(int value) {
        this.value = value;
    }

    public int getValue() {
        return value;
    }
}
```

Класс Архив:

```
import java.util.HashMap;

public class Archive {
    private static HashMap<Student, HashMap<Course, Grade>> records = new HashMap<>();

    public static void addRecord(Student student, Course course, Grade grade) {
        records.computeIfAbsent(student, k -> new HashMap<>()).put(course, grade);
    }

    public HashMap<Course, Grade> getStudentRecords(Student student) {
        return records.get(student);
    }

    public static void printAllRecords() {
        StringBuilder builder = new StringBuilder();
        for (HashMap.Entry<Student, HashMap<Course, Grade>> entry : records.entrySet()) {
            Student student = entry.getKey();
            HashMap<Course, Grade> studentRecords = entry.getValue();
            builder.append("Стундент: ").append(student.getName()).append("\n");
            for (HashMap.Entry<Course, Grade> recordEntry : studentRecords.entrySet()) {
                Course course = recordEntry.getKey();
                Grade grade = recordEntry.getValue();
                builder.append("\tКурс: ").append(course.getName()).append(", Оценка:
").append(grade.getValue()).append("\n");
            }
        }
        System.out.println(builder.toString());
    }
}
```

Класс с методом main:

```
public class Main {
    public static void main(String[] args) {
        Faculty faculty = new Faculty("Факультет электронно-информационных систем");

        Course course = new Course("Основы алгоритмизации и программирования");
        faculty.addCourse(course);
```

```
        Teacher teacher = new Teacher("Щербаков Марк Егорович");
        teacher.announceRegistryForCourse(course);

        Student student1 = new Student("Степанова Алиса Константиновна");
        Student student2 = new Student("Жаров Артём Ильич");

        course.enrollStudent(student1);
        course.enrollStudent(student2);

        teacher.stopTeachingCourse(course);

        Archive.printAllRecords();
    }
}
```

Результаты работы программы:

```
Прием на курс Основы алгоритмизации и программирования
Студент Степанова Алиса Константиновна начал изучать курс Основы алгоритмизации и программирования
Студент Жаров Артём Ильич начал изучать курс Основы алгоритмизации и программирования
Завершение курса Основы алгоритмизации и программирования
Поставьте оценку студенту Степанова Алиса Константиновна по предмету Основы алгоритмизации и программирования
4
Студент Степанова Алиса Константиновна закончил изучать курс Основы алгоритмизации и программирования
Поставьте оценку студенту Жаров Артём Ильич по предмету Основы алгоритмизации и программирования
6
Студент Жаров Артём Ильич закончил изучать курс Основы алгоритмизации и программирования
Стундент: Жаров Артём Ильич
    Курс: Основы алгоритмизации и программирования, Оценка: 6
Стундент: Степанова Алиса Константиновна
    Курс: Основы алгоритмизации и программирования, Оценка: 4
```

Вывод: приобрели практические навыки в области объектно-ориентированного проектирования.