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-6

Юсковец М.А.

Проверил:

Монтик Н.С.

Брест, 2022

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

**Ход работы:**

Вариант 25

**Задание 1:**

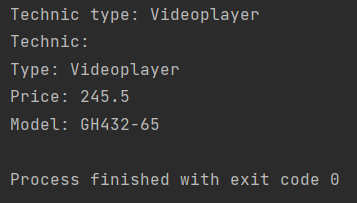
Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Техника ← abstract class Плеер ← class Видеоплеер.

**Текст программы:**

public class Main {  
 public static void main(String[] args) {  
 Videoplayer vp = new Videoplayer("Videoplayer", 245.5, "GH432-65");  
 vp.showType();  
 vp.showAll();  
 }  
}  
interface Technic {  
 void showType();  
 void showAll();  
}  
  
abstract class Player implements Technic {  
 private String type;  
 private Double price;  
 public Player(String type, Double price) {  
 this.type = type;  
 this.price = price;  
 }  
 public String getType() {  
 return this.type;  
 }  
 public Double getPrice() {  
 return this.price;  
 }  
}  
  
class Videoplayer extends Player {  
 private String model;  
 public Videoplayer(String type, Double price, String model) {  
 super(type, price);  
 this.model = model;  
 }  
 @Override  
 public void showType() {  
 System.*out*.println("Technic type: "+super.getType());  
 }  
 @Override  
 public void showAll() {  
 System.*out*.println("Technic:");  
 System.*out*.println("Type: "+super.getType()  
 +"\nPrice: "+super.getPrice()  
 +"\nModel: "+this.model);  
 }  
}

**Результат программы:**



**Задание 2:**

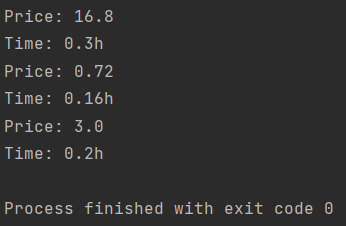
В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты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

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

**Текст программы:**

import java.util.ArrayList;  
  
public class Main {  
 public static void main(String[] args) {  
 ArrayList<Transport> list = new ArrayList<>();  
 Car car1 = new Car(21.0);  
 Bicycle bicycle1 = new Bicycle(2.4);  
 Cart cart1 = new Cart(6.0);  
  
 list.add(car1);  
 list.add(bicycle1);  
 list.add(cart1);  
  
 for(Transport t : list) {  
 t.getPrice();  
 t.getTime();  
 }  
 }  
}  
  
abstract class Transport {  
 private Double kilometersCount;  
 public Transport(Double km) {  
 this.kilometersCount = km;  
 }  
 public Double getKilometersCount() {  
 return this.kilometersCount;  
 }  
 public abstract void getPrice();  
 public abstract void getTime();  
}  
  
class Car extends Transport {  
 private Double speed = 70.0;  
 private Double priceOneKm = 0.8;  
 public Car(Double km) {  
 super(km);  
 }  
 public Double getSpeed() {  
 return this.speed;  
 }  
 public Double getPriceOneKm() {  
 return this.priceOneKm;  
 }  
 public void getPrice() {  
 System.*out*.println("Price: "+(super.getKilometersCount()\*this.priceOneKm));  
 }  
 public void getTime() {  
 System.*out*.println("Time: "+(super.getKilometersCount()/this.speed) + "h");  
 }  
}  
  
class Bicycle extends Transport {  
 private Double speed = 15.0;  
 private Double priceOneKm = 0.3;  
  
 public Bicycle(Double km) {  
 super(km);  
 }  
 public Double getSpeed() {  
 return this.speed;  
 }  
 public Double getPriceOneKm() {  
 return this.priceOneKm;  
 }  
 public void getPrice() {  
 System.*out*.println("Price: "+(super.getKilometersCount()\*this.priceOneKm));  
 }  
 public void getTime() {  
 System.*out*.println("Time: "+(super.getKilometersCount()/this.speed) + "h");  
 }  
}  
  
class Cart extends Transport {  
 private Double speed = 30.0;  
 private Double priceOneKm = 0.5;  
 public Cart(Double km) {  
 super(km);  
 }  
 public Double getSpeed() {  
 return this.speed;  
 }  
 public Double getPriceOneKm() {  
 return this.priceOneKm;  
 }  
 public void getPrice() {  
 System.*out*.println("Price: "+(super.getKilometersCount()\*this.priceOneKm));  
 }  
 public void getTime() {  
 System.*out*.println("Time: "+(super.getKilometersCount()/this.speed) + "h");  
 }  
}

**Результат программы:**



**Задание 3:**

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

**Текст программы:**

class Person {  
 private String name;  
 private String surname;  
 public Person(String n, String s) {  
 this.name = n;  
 this.surname = s;  
 }  
 public String getName() {  
 return this.name;  
 }  
 public String getSurname() {  
 return this.surname;  
 }  
}  
  
class Doctor extends Person {  
 private String speciality;  
  
 public Doctor(String name, String surname, String speciality) {  
 super(name, surname);  
 this.speciality = speciality;  
 }  
 public String getSpeciality() {  
 return this.speciality;  
 }  
  
 public void addAppointments(String appointments, Patient p, Nurse n) {  
 p.setAppointments(appointments);  
 p.setDischarge(false);  
 n.setAppointments(false);  
 }  
}  
  
class Nurse extends Person {  
 private Boolean appointmentsDone;  
  
 public Nurse(String name, String surname) {  
 super(name, surname);  
 }  
 public Boolean getAppointmentsDone() {  
 return this.appointmentsDone;  
 }  
  
 public void setAppointments(Boolean res) {  
 this.appointmentsDone = res;  
 }  
  
 public void setAppointmentsDone(Patient p) {  
 this.appointmentsDone = true;  
 p.setDischarge(true);  
 }  
}  
  
class Patient extends Person {  
 private Doctor doc;  
 private String appointments;  
 private Boolean discharge;  
  
 public Patient(String name, String surname, Doctor doc) {  
 super(name, surname);  
 this.doc = doc;  
 }  
  
 public Doctor getDoctor() {  
 return this.doc;  
 }  
 public String getAppointments() {  
 return this.appointments;  
 }  
 public void setAppointments(String appointments) {  
 this.appointments = appointments;  
 }  
 public Boolean getDischarge() {  
 return this.discharge;  
 }  
 public void setDischarge(Boolean d) {  
 this.discharge = d;  
 }  
}

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