|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 4 |

**Название:**

Внутренние классы и интерфейсы

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Д.Ю.Хотин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В.Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 1, задание 7:** Создать класс Cправочная Cлужба Oбщественного Tранспорта с внутренним классом, с помощью объектов которого можно хранить информацию о времени, линиях маршрутов и стоимости проезда.

Код

|  |
| --- |
| import java.util.\*;  public class Main  {  public static void main(String[] args) {  PublicTransportDirectory transportDirectory = new PublicTransportDirectory();  transportDirectory.addRoute("Birulovo", "12:00", 55.50);  transportDirectory.printRoutes();  }  }  public class PublicTransportDirectory {  private ArrayList<TransportRoute> routes;  public PublicTransportDirectory() {  routes = new ArrayList<>();  }  public void addRoute(String line, String time, double fare) {  TransportRoute route = new TransportRoute(line, time, fare);  routes.add(route);  }  public void printRoutes() {  for (TransportRoute route : routes) {  System.out.println(route);  }  }  private class TransportRoute {  private String line;  private String time;  private double fare;  public TransportRoute(String line, String time, double fare) {  this.line = line;  this.time = time;  this.fare = fare;  }  public String getLine() {  return line;  }  public String getTime() {  return time;  }  public double getFare() {  return fare;  }  public String toString() {  return "Line: " + line + ", Time: " + time + ", Fare: " + fare;  }  }  } |

**Вариант 1, задание 8:** Создать класс Computer (компьютер) с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти.

Код

|  |
| --- |
| import java.util.\*;  public class Main  {  public static void main(String[] args) {  Computer asosComputer = new Computer("asos", 16, "windows", 10, "intel", 3200, 1000);  System.out.println(asosComputer.getBrand());  }  }  public class Computer {  private String brand;  private int storageSize;  private OperatingSystem os;  private Processor processor;  private Memory memory;  public Computer(String brand, int storageSize, String osName, int osVersion, String processorName, double processorSpeed, int memorySize) {  this.brand = brand;  this.storageSize = storageSize;  this.os = new OperatingSystem(osName, osVersion);  this.processor = new Processor(processorName, processorSpeed);  this.memory = new Memory(memorySize);  }  public String getBrand() {  return brand;  }  public int getStorageSize() {  return storageSize;  }  public OperatingSystem getOs() {  return os;  }  public Processor getProcessor() {  return processor;  }  public Memory getMemory() {  return memory;  }  public class OperatingSystem {  private String name;  private int version;  public OperatingSystem(String name, int version) {  this.name = name;  this.version = version;  }  public String getName() {  return name;  }  public int getVersion() {  return version;  }  }  public class Processor {  private String name;  private double speed;  public Processor(String name, double speed) {  this.name = name;  this.speed = speed;  }  public String getName() {  return name;  }  public double getSpeed() {  return speed;  }  }  public class Memory {  private int size;  public Memory(int size) {  this.size = size;  }  public int getSize() {  return size;  }  }  } |

**Вариант 2, задание 7:** Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов interface Врач <- class Хирург <- class Нейрохирург.

Код

|  |
| --- |
| import java.util.\*;  public class Main  {  public static void main(String[] args) {  NeuroSurgeon neuroSurgeon = new NeuroSurgeon("Andrteey");  System.out.println(neuroSurgeon.getName());  neuroSurgeon.sayHello();  }  }  abstract class Doctor {  private String \_name;  public Doctor(String name) {  this.\_name = name;  }  public String getName() {  return \_name;  }  public void sayHello() {  System.out.println("Hello. I am a doctor!");  }  }  abstract class Surgeon extends Doctor {  public Surgeon(String name) {  super(name);  }  @Override  public void sayHello() {  System.out.println("Hello. I am a surgeon!");  }  }  class NeuroSurgeon extends Surgeon {  public NeuroSurgeon(String name) {  super(name);  }  @Override  public void sayHello() {  System.out.println("Hello. I am a neuro surgeon!");  }  } |

**Вариант 2, задание 8:** Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов interface Корабль <- class Грузовой Корабль <- class Танкер.

Код

|  |
| --- |
| import java.util.\*;  public class Main  {  public static void main(String[] args) {  Tanker tanker = new Tanker("Saint Maria", 10000, 5000);  tanker.loadOil();  }  }  public interface Ship {  void sail();  }  public class CargoShip implements Ship {  private String name;  private int cargoCapacity;  public CargoShip(String name, int cargoCapacity) {  this.name = name;  this.cargoCapacity = cargoCapacity;  }  @Override  public void sail() {  System.out.println("Ship " + name + " sail in trip!");  }  public void loadCargo() {  System.out.println("Ship " + name + " loaded with cargo is " + cargoCapacity + " tons.");  }  public String getName() {  return name;  }  }  public class Tanker extends CargoShip {  private int oilCapacity;  public Tanker(String name, int cargoCapacity, int oilCapacity) {  super(name, cargoCapacity);  this.oilCapacity = oilCapacity;  }  public void loadOil() {  System.out.println("Tanker " + getName() + " loaded with oil is " + oilCapacity + " tons.");  }  public int getOilCapacity() {  return oilCapacity;  }  } |