|  |  |
| --- | --- |
| **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 №** | 3 |

**Название:**

Классы, наследование, полиморфизм

**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:** Определить класс Дробь в виде пары (m, n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

Код

|  |
| --- |
| import java.util.\*;  import java.io.\*;  import java.lang.\*;  public class Main {  public static void main(String[] args) {  int k = 5;  Decimal[] decimals = new Decimal[k];  Scanner scanner = new Scanner(System.in);  for (int i = 0; i < k; i++) {  System.out.println("������� ����� " + (i+1) + " (� ������� m/n):");  String input = scanner.nextLine();  String[] parts = input.split("/");  int m = Integer.parseInt(parts[0]);  int n = Integer.parseInt(parts[1]);  try {  decimals[i] = new Decimal(m, n);  } catch (OutOfMemoryError e) {  System.out.println("������: �������� ������");  }  }  System.out.println("������ ������:");  for (int i = 0; i < k; i++) {  System.out.println(decimals[i]);  }  for (int i = 0; i < k-1; i += 2) {  decimals[i] = decimals[i].add(decimals[i+1]);  }  System.out.println("���������� ������ ������:");  for (int i = 0; i < k; i++) {  System.out.println(decimals[i]);  }  }  }  public class Decimal {  private int \_numerator;  private int \_denominator;  public Decimal(int numerator, int denominator) {  int nod = nod(numerator, denominator);  this.\_numerator = numerator / nod;  this.\_denominator = denominator / nod;  }  public Decimal(int numerator) {  this.\_numerator = numerator;  this.\_denominator = 1;  }  public Decimal() {  this.\_denominator = 0;  this.\_numerator = 1;  }  public Decimal add(Decimal newDecimal) {  try {  System.out.println("add" + newDecimal);  int newNumerator = this.\_numerator \* newDecimal.\_denominator + newDecimal.\_numerator \* this.\_denominator;  int newDenominator = this.\_denominator \* newDecimal.\_denominator;  return new Decimal(newNumerator, newDenominator);  } catch (ArithmeticException e) {  System.out.println("������: ������ �����");  return new Decimal();  }  }  public Decimal subtract(Decimal newDecimal) {  int newNumerator = this.\_numerator \* newDecimal.\_denominator - newDecimal.\_numerator \* this.\_denominator;  int newDenominator = this.\_denominator \* newDecimal.\_denominator;  return new Decimal(newNumerator, newDenominator);  }  public Decimal multiply(Decimal newDecimal) {  int newNumerator = this.\_numerator \* newDecimal.\_numerator;  int newDenominator = this.\_denominator \* newDecimal.\_denominator;  return new Decimal(newNumerator, newDenominator);  }  public Decimal divide(Decimal newDecimal) {  int newNumerator = this.\_numerator \* newDecimal.\_denominator;  int newDenominator = this.\_denominator \* newDecimal.\_numerator;  return new Decimal(newNumerator, newDenominator);  }  @Override  public String toString() {  return \_numerator + "/" + \_denominator;  }  private int nod(int a, int b) {  if (b == 0) {  return a;  }  return nod(b, a % b);  }  } |

**Вариант 1, задание 8:** Определить класс Комплекс. Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения, деления, присваивания комплексных чисел. Создать два вектора размерности n из комплексных координат. Передать их в метод, который выполнит их сложение.

Код

|  |
| --- |
| import java.util.\*;  import java.io.\*;  import java.lang.\*;  public class Main {  public static void main(String[] args) {  int n = 5;  Complex[] vector1 = new Complex[n];  Complex[] vector2 = new Complex[n];  for (int i = 0; i < n; i++) {  vector1[i] = new Complex(i, 3 \* i);  vector2[i] = new Complex(4 \* i, i);  }  for (int i = 0; i < n; i++) {  System.out.print(vector1[i].real + " + " + vector1[i].imaginary + "i" + " ");  }  System.out.println();  for (int i = 0; i < n; i++) {  System.out.print(vector2[i].real + " + " + vector2[i].imaginary + "i" + " ");  }  System.out.println();  Complex[] sum = addVectors(vector1, vector2);  for (int i = 0; i < n; i++) {  System.out.println(sum[i].real + " + " + sum[i].imaginary + "i");  }  }  public static Complex[] addVectors(Complex[] vector1, Complex[] vector2) {  int n = vector1.length;  Complex[] sum = new Complex[n];  for (int i = 0; i < n; i++) {  sum[i] = vector1[i].add(vector2[i]);  }  return sum;  }  }  public class Complex {  public double real;  public double imaginary;  public Complex() {  this(0, 0);  }  public Complex(double real, double imaginary) {  this.real = real;  this.imaginary = imaginary;  }  public Complex(Complex other) {  this(other.real, other.imaginary);  }  public Complex add(Complex other) {  return new Complex(this.real + other.real, this.imaginary + other.imaginary);  }  public Complex subtract(Complex other) {  return new Complex(this.real - other.real, this.imaginary - other.imaginary);  }  public Complex multiply(Complex other) {  double realPart = this.real \* other.real - this.imaginary \* other.imaginary;  double imaginaryPart = this.real \* other.imaginary + this.imaginary \* other.real;  return new Complex(realPart, imaginaryPart);  }  public Complex divide(Complex other) {  try {  double denominator = other.real \* other.real + other.imaginary \* other.imaginary;  double realPart = (this.real \* other.real + this.imaginary \* other.imaginary) / denominator;  double imaginaryPart = (this.imaginary \* other.real - this.real \* other.imaginary) / denominator;  return new Complex(realPart, imaginaryPart);  } catch (ArithmeticException e) {  System.out.println("Ошибка: ошибка ввода");  return new Complex();  }  }  } |

**Вариант 2, задание 7:** Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString().Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров.

\* Создать массив объектов. Вывести:

\* a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное;

\* b) сведения об абонентах, которые пользовались междугородной связью;

\* c) сведения об абонентах в алфавитном порядке.

Код

|  |
| --- |
| import java.util.\*;  public class Main {  public static int getRandomInt() {  return 0 + (int)(Math.random() \* 4);  }  public static String generateRandomName() {  String[] names = {"Alice", "Bob", "Charlie", "David", "Emma"};  return names[getRandomInt()];  }  public static String generateRandomSecondName() {  String[] surnames = {"Smith", "Johnson", "Williams", "Jones", "Brown"};  return surnames[getRandomInt()];  }  public static String generateRandomThirdName() {  String[] patronymics = {"James", "William", "Thomas", "Michael", "David"};  return patronymics[getRandomInt()];  }  public static String generateRandomStreet() {  String[] streets = {"Main Street", "Elm Street", "Park Avenue", "Broadway", "Maple Street"};  return streets[getRandomInt()];  }  public static void main(String[] args) {  Phone[] phoneArray = new Phone[5];  for (int i = 1; i < 6; i++) {  try {  phoneArray[i-1] = new Phone(i, generateRandomSecondName(), generateRandomName(), generateRandomThirdName(), generateRandomStreet(), i\*1000 + i\*100 + i\*11, i\*500, i\*200, i % 2 == 0 ? 0 : i, i % 2 != 0 ? 0 : i);  } catch (PhoneException e) {  System.out.println(e.getMessage());  }  }  PhoneBook phoneBook = new PhoneBook(phoneArray);  System.out.println("getInternationalCalls");  for (Phone phone:phoneBook.getInternationalCalls()) {  System.out.println(phone.toString());  }  System.out.println("getLocalCallsAboveDuration");  for (Phone phone:phoneBook.getLocalCallsAboveDuration(4)) {  System.out.println(phone.toString());  }  System.out.println("sortByName");  for (Phone phone:phoneBook.sortByName()) {  System.out.println(phone.toString());  }  }  }  public class Phone {  private int id;  private String lastName;  private String firstName;  private String middleName;  private String address;  private int creditCardNumber;  private double debit;  private double credit;  private double localCallDuration;  private double internationalCallDuration;  public Phone(int id, String lastName, String firstName, String middleName, String address,  int creditCardNumber, double debit, double credit, double localCallDuration, double internationalCallDuration) throws PhoneException {  if (id <= 0) {  throw new PhoneException("Invalid id: " + id);  }  if (lastName == null || lastName.isEmpty()) {  throw new PhoneException("Surname is null or empty");  }  if (firstName == null || firstName.isEmpty()) {  throw new PhoneException("Name is null or empty");  }  if (middleName == null || middleName.isEmpty()) {  throw new PhoneException("Name is null or empty");  }  if (address == null || address.isEmpty()) {  throw new PhoneException("Address is null or empty");  }  if (creditCardNumber <=0) {  throw new PhoneException("Invalid card number");  }  if (debit < 0) {  throw new PhoneException("Invalid debit: " + debit);  }  if (credit < 0) {  throw new PhoneException("Invalid credit: " + credit);  }  if (localCallDuration < 0) {  throw new PhoneException("Invalid local calls time: " + localCallDuration);  }  if (internationalCallDuration < 0) {  throw new PhoneException("Invalid intercity calls time: " + internationalCallDuration);  }  this.id = id;  this.lastName = lastName;  this.firstName = firstName;  this.middleName = middleName;  this.address = address;  this.creditCardNumber = creditCardNumber;  this.debit = debit;  this.credit = credit;  this.localCallDuration = localCallDuration;  this.internationalCallDuration = internationalCallDuration;  }  public void setId(int id) {  this.id = id;  }  public int getId() {  return id;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }  public String getLastName() {  return lastName;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getFirstName() {  return firstName;  }  public void setMiddleName(String middleName) {  this.middleName = middleName;  }  public String getMiddleName() {  return middleName;  }  public void setAddress(String address) {  this.address = address;  }  public String getAddress() {  return address;  }  public void setCreditCardNumber(int creditCardNumber) {  this.creditCardNumber = creditCardNumber;  }  public int getCreditCardNumber() {  return creditCardNumber;  }  public void setDebit(double debit) {  this.debit = debit;  }  public double getDebit() {  return debit;  }  public void setCredit(double credit) {  this.credit = credit;  }  public double getCredit() {  return credit;  }  public void setLocalCallDuration(double localCallDuration) {  this.localCallDuration = localCallDuration;  }  public double getLocalCallDuration() {  return localCallDuration;  }  public void setInternationalCallDuration(double internationalCallDuration) {  this.internationalCallDuration = internationalCallDuration;  }  public double getInternationalCallDuration() {  return internationalCallDuration;  }  public String toString() {  return "ID: " + id + ", FIO: " + lastName + " " + firstName + " " + middleName + ", Address: " + address  + ", Credit card number: " + creditCardNumber + ", Debet: " + debit + ", Credit: " + credit  + ", Local call duration: " + localCallDuration + ", International call duration: " + internationalCallDuration;  }  }  public class PhoneBook {  private Phone[] phoneArray;  public PhoneBook() {};  public PhoneBook(Phone[] phoneArray) {  this.phoneArray = phoneArray;  }  public void setPhoneArray(Phone[] phoneArray) {  this.phoneArray = phoneArray;  }  public Phone[] getPhoneArray() {  return phoneArray;  }  public Phone[] getLocalCallsAboveDuration(double duration) {  Phone[] result = new Phone[phoneArray.length];  int count = 0;  for (Phone phone : phoneArray) {  if (phone.getLocalCallDuration() > duration) {  result[count] = phone;  count++;  }  }  return Arrays.copyOfRange(result, 0, count);  }  public Phone[] getInternationalCalls() {  Phone[] result = new Phone[phoneArray.length];  int count = 0;  for (Phone phone : phoneArray) {  if (phone.getInternationalCallDuration() > 0) {  result[count] = phone;  count++;  }  }  return Arrays.copyOfRange(result, 0, count);  }  public Phone[] sortByName() {  Phone[] result = Arrays.copyOf(phoneArray, phoneArray.length);  Arrays.sort(result, Comparator.comparing(Phone::getLastName).thenComparing(Phone::getFirstName).thenComparing(Phone::getMiddleName));  return result;  }  }  class PhoneException extends Throwable {  private String message;  public PhoneException(String message) {  this.message = message;  }  public String getMessage() {  return this.message;  }  } |

**Вариант 2, задание 8:** Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Создать массив объектов.

\* Вывести:

\* a) список автомобилей заданной марки;

\* b) список автомобилей заданной модели, которые эксплуатируются больше n лет;

\* c) список автомобилей заданного года выпуска, цена которых больше указанной.

Код

|  |
| --- |
| import java.util.\*;  public class Main {  public static int getRandomInt() {  return 0 + (int)(Math.random() \* 4);  }  public static String generateRandomBrand() {  String[] carBrands = {"Toyota", "Honda", "Ford", "Chevrolet", "BMW"};  return carBrands[getRandomInt()];  }  public static String generateRandomModel() {  String[] carModels = {"Camry", "Civic", "Mustang", "Camaro", "X5"};  return carModels[getRandomInt()];  }  public static String generateRandomColors() {  String[] colors = {"red", "green", "blue", "yellow", "orange"};  return colors[getRandomInt()];  }  public static String generateRandomNumbers() {  String[] carNumbers = {"A123BC", "X789YZ", "K456LM", "P987QR", "G654TH"};  return carNumbers[getRandomInt()];  }  public static int generateRandomYear() {  int[] carYears = {2010, 2015, 2019, 2018, 2000};  return carYears[getRandomInt()];  }  public static void main(String[] args) {  Car[] cars = new Car[5];  for (int i = 1; i < 6; i++) {  cars[i-1] = new Car(i, generateRandomBrand(), generateRandomModel(), generateRandomYear(), generateRandomColors(), 2000 + (i-1), generateRandomNumbers());  }  CarList carList = new CarList(cars);  System.out.println("getCarsByBrand");  for (Car car:carList.getCarsByBrand(generateRandomBrand())) {  System.out.println(car.toString());  }  System.out.println("getCarsByModelAndYears");  for (Car car:carList.getCarsByModelAndYears(generateRandomModel(), generateRandomYear())) {  System.out.println(car.toString());  }  System.out.println("getCarsByYearAndPrice");  for (Car car:carList.getCarsByYearAndPrice(generateRandomYear(), 2000)) {  System.out.println(car.toString());  }  }  }  public class Car {  private int id;  private String brand;  private String model;  private int year;  private String color;  private double price;  private String registrationNumber;  public Car(int id, String brand, String model, int year, String color, double price, String registrationNumber) {  this.id = id;  this.brand = brand;  this.model = model;  this.year = year;  this.color = color;  this.price = price;  this.registrationNumber = registrationNumber;  }  // геттеры и сеттеры для всех полей класса Car  public void setId(int id) {  this.id = id;  }  public int getId() {  return id;  }  public void setBrand(String brand) {  this.brand = brand;  }  public String getBrand() {  return brand;  }  public void setModel(String model) {  this.model = model;  }  public String getModel() {  return model;  }  public void setYear(int year) {  this.year = year;  }  public int getYear() {  return year;  }  public void setColor(String color) {  this.color = color;  }  public String getColor() {  return color;  }  public void setPrice(double price) {  this.price = price;  }  public double getPrice() {  return price;  }  public void setRegistrationNumber(String registrationNumber) {  this.registrationNumber = registrationNumber;  }  public String getRegistrationNumber() {  return registrationNumber;  }  public String toString() {  return "Car{" +  "id=" + id +  ", brand='" + brand + '\'' +  ", model='" + model + '\'' +  ", year=" + year +  ", color='" + color + '\'' +  ", price=" + price +  ", registrationNumber='" + registrationNumber + '\'' +  '}';  }  }  public class CarList {  private Car[] cars;  public CarList(Car[] cars) {  this.cars = cars;  }  // методы для получения списка автомобилей заданной марки  public Car[] getCarsByBrand(String brand) {  List<Car> carsList = new ArrayList<>();  for (Car car : cars) {  if (car.getBrand().equals(brand)) {  carsList.add(car);  }  }  return carsList.toArray(new Car[carsList.size()]);  }  // методы для получения списка автомобилей заданной модели, которые эксплуатируются больше n лет  public Car[] getCarsByModelAndYears(String model, int years) {  List<Car> carsList = new ArrayList<>();  for (Car car : cars) {  if (car.getModel().equals(model) && (2023 - car.getYear()) > years) {  carsList.add(car);  }  }  return carsList.toArray(new Car[carsList.size()]);  }  // методы для получения списка автомобилей заданного года выпуска, цена которых больше указанной  public Car[] getCarsByYearAndPrice(int year, double price) {  List<Car> carsList = new ArrayList<>();  for (Car car : cars) {  if (car.getYear() == year && car.getPrice() > price) {  carsList.add(car);  }  }  return carsList.toArray(new Car[carsList.size()]);  }  } |

**Вариант 3, задание 6:** Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, вывести на консоль цвет бутона.

Код

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  Petal[] petals = new Petal[5];  for (int i = 0; i < 5; i++) {  petals[i] = new Petal("red");  }  Bud bud = new Bud("green", false, petals);  Rose rose = new Rose("Red rose", bud);  System.out.println(rose);  System.out.println(bud.hashCode());  System.out.println(petals[0].hashCode());  System.out.println(rose.hashCode());  rose.bloom();  System.out.println(rose);  rose.wilt();  System.out.println(rose);  }  }  public class Rose {  private String name;  private Bud bud;  public Rose(String name, Bud bud) {  this.name = name;  this.bud = bud;  }  public void bloom() {  bud.setOpen(true);  }  public void wilt() {  bud.setOpen(false);  }  public String getBudColor() {  return bud.getColor();  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (!(o instanceof Rose)) return false;  Rose rose = (Rose) o;  return name.equals(rose.name) && bud.equals(rose.bud);  }  @Override  public int hashCode() {  int result = name.hashCode();  result = 31 \* result + bud.hashCode();  return result;  }  @Override  public String toString() {  return "Rose{" +  "name='" + name + '\'' +  ", bud=" + bud +  '}';  }  }  public class Petal {  private String color;  public Petal(String color) {  this.color = color;  }  public String getColor() {  return color;  }  public void setColor(String color) {  this.color = color;  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (!(o instanceof Petal)) return false;  Petal petal = (Petal) o;  return color.equals(petal.color);  }  @Override  public int hashCode() {  int result = color.hashCode();  return result;  }  @Override  public String toString() {  return "Petal{" +  "color='" + color +  '}';  }  }  public class Bud {  private String color;  private boolean isOpen;  private Petal[] petals;  public Bud(String color, boolean isOpen, Petal[] petals) {  this.color = color;  this.isOpen = isOpen;  this.petals = petals;  }  public String getColor() {  return color;  }  public void setColor(String color) {  this.color = color;  }  public boolean isOpen() {  return isOpen;  }  public void setOpen(boolean open) {  isOpen = open;  }  public void open() {  isOpen = true;  }  public void close() {  isOpen = false;  }  public Petal[] getPetals() {  return petals;  }  public void setPetals(Petal[] petals) {  this.petals = petals;  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (!(o instanceof Bud)) return false;  Bud bud = (Bud) o;  return isOpen == bud.isOpen &&  Objects.equals(color, bud.color) &&  Arrays.equals(petals, bud.petals);  }  @Override  public int hashCode() {  int result = Objects.hash(color, isOpen);  result = 31 \* result + Arrays.hashCode(petals);  return result;  }  @Override  public String toString() {  return "Bud{" +  "color='" + color + '\'' +  ", isOpen=" + isOpen +  ", petals=" + Arrays.toString(petals) +  '}';  }  } |

**Вариант 3, задание 7:** Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().Создать объект класса Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.

Код

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  Leaf greenLeaf = new Leaf("green");  Leaf redLeaf = new Leaf("red");  Leaf yellowLeaf = new Leaf("yellow");  Tree tree = new Tree();  tree.addLeaf(greenLeaf);  tree.addLeaf(redLeaf);  tree.addLeaf(yellowLeaf);  System.out.println("Initial state: " + tree);  tree.bloom();  System.out.println("After blooming: " + tree);  tree.turnLeavesYellow();  System.out.println("After turning leaves yellow: " + tree);  tree.fall();  System.out.println("After falling: " + tree);  tree.coverLeavesWithFrost();  System.out.println("After covering frost: " + tree);  }  }  public class Leaf {  private String color;  private boolean isFalling;  private boolean isCoveredByFrost;  public Leaf(String color) {  this.color = color;  this.isFalling = false;  this.isCoveredByFrost = false;  }  public void bloom() {  this.isFalling = false;  }  public void fall() {  this.isFalling = true;  this.isCoveredByFrost = false;  }  public void coverWithFrost() {  this.isCoveredByFrost = true;  }  public void turnYellow() {  this.color = "yellow";  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (!(o instanceof Leaf)) return false;  Leaf leaf = (Leaf) o;  if (isFalling != leaf.isFalling) return false;  if (isCoveredByFrost != leaf.isCoveredByFrost) return false;  return color.equals(leaf.color);  }  @Override  public int hashCode() {  int result = color.hashCode();  result = 31 \* result + (isFalling ? 1 : 0);  result = 31 \* result + (isCoveredByFrost ? 1 : 0);  return result;  }  @Override  public String toString() {  return "Leaf{" +  "color='" + color + '\'' +  ", isFalling=" + isFalling +  ", isCoveredByFrost=" + isCoveredByFrost +  '}';  }  }  public class Tree {  private List<Leaf> leaves;  public Tree() {  this.leaves = new ArrayList<>();  }  public void addLeaf(Leaf leaf) {  this.leaves.add(leaf);  }  public void bloom() {  for (Leaf leaf : leaves) {  leaf.bloom();  }  }  public void fall() {  for (Leaf leaf : leaves) {  leaf.fall();  }  }  public void coverLeavesWithFrost() {  for (Leaf leaf : leaves) {  leaf.coverWithFrost();  }  }  public void turnLeavesYellow() {  for (Leaf leaf : leaves) {  leaf.turnYellow();  }  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (!(o instanceof Tree)) return false;  Tree tree = (Tree) o;  return leaves.equals(tree.leaves);  }  @Override  public int hashCode() {  return leaves.hashCode();  }  @Override  public String toString() {  return "Tree{" +  "leaves=" + leaves +  '}';  }  } |

**Вариант 4, задание 7:** Построить модель программной системы. Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

Код

|  |
| --- |
| public class Main {  public static void main(String[] args) {  }  }  public class TelephoneExchange {  private List<Subscriber> subscribers; // список абонентов  public TelephoneExchange() {  this.subscribers = new ArrayList<>();  }  public void addSubscriber(Subscriber subscriber) {  subscribers.add(subscriber);  }  public void removeSubscriber(Subscriber subscriber) {  subscribers.remove(subscriber);  }  public List<Subscriber> getSubscribers() {  return subscribers;  }  }  public class Subscriber {  private String name;  private String phoneNumber;  private double account; // баланс абонента  private boolean isBlocked; // заблокирован ли абонент  private List<Service> services; // список подключенных услуг  public Subscriber(String name, String phoneNumber) {  this.name = name;  this.phoneNumber = phoneNumber;  this.account = 0;  this.isBlocked = false;  this.services = new ArrayList<>();  }  public void addService(Service service) {  services.add(service);  }  public void removeService(Service service) {  services.remove(service);  }  public void changePhoneNumber(String newNumber) {  this.phoneNumber = newNumber;  }  public void payBill(double amount) {  account -= amount;  }  public void requestDisconnect() {  isBlocked = true;  }  public void requestReconnect() {  isBlocked = false;  }  public String getPhoneNumber() {  return phoneNumber;  }  public double getAccount() {  return account;  }  public boolean isBlocked() {  return isBlocked;  }  public List<Service> getServices() {  return services;  }  }  public class Service {  private String name;  private double cost;  public Service(String name, double cost) {  this.name = name;  this.cost = cost;  }  public double getCost() {  return cost;  }  }  public class Administrator {  public void changePhoneNumber(Subscriber subscriber, String newNumber) {  subscriber.changePhoneNumber(newNumber);  }  public void addService(Subscriber subscriber, Service service) {  subscriber.addService(service);  }  public void removeService(Subscriber subscriber, Service service) {  subscriber.removeService(service);  }  public void blockSubscriber(Subscriber subscriber) {  subscriber.requestDisconnect();  }  public void unblockSubscriber(Subscriber subscriber) {  subscriber.requestReconnect();  }  public void updateAccount(Subscriber subscriber, double amount) {  subscriber.payBill(amount);  }  } |

**Вариант 4, задание 8:** Построить модель программной системы. Система Автобаза. Диспетчер распределяет заявки на Рейсы между Водителями и назначает для этого Автомобиль. Водитель может сделать заявку на ремонт. Диспетчер может отстранить Водителя от работы. Водитель делает отметку о выполнении Рейса и состоянии Автомобиля.

Код

|  |
| --- |
| import java.util.\*;  public class Autobaza {  public static void main(String[] args) {  Dispatcher dispatcher = new Dispatcher();  // create some drivers  Driver driver1 = new Driver("John");  Driver driver2 = new Driver("Sarah");  Driver driver3 = new Driver("Bob");  // add drivers to dispatcher  dispatcher.addDriver(driver1);  dispatcher.addDriver(driver2);  dispatcher.addDriver(driver3);  // create some cars  Car car1 = new Car("Toyota", "Camry", 2010);  Car car2 = new Car("Honda", "Accord", 2012);  Car car3 = new Car("Nissan", "Altima", 2015);  // add cars to dispatcher  dispatcher.addCar(car1);  dispatcher.addCar(car2);  dispatcher.addCar(car3);  // assign driver1 to car1  dispatcher.assignDriverToCar(driver1, car1);  // mark driver1's trip as complete  dispatcher.markTripComplete(driver1);  // mark driver1's car for repair  dispatcher.markDriverForRepair(driver1);  // suspend driver1  dispatcher.suspendDriver(driver1);  // assign driver2 to car2  dispatcher.assignDriverToCar(driver2, car2);  // mark driver2's trip as complete  dispatcher.markTripComplete(driver2);  // suspend driver2  dispatcher.suspendDriver(driver2);  // reinstate driver2  dispatcher.reinstateDriver(driver2);  System.out.println(dispatcher.getDrivers());  System.out.println(dispatcher.getCars());  }  }  class Driver {  private String name;  private boolean available;  private boolean onDuty;  private boolean needsRepair;  private Car car;  public Driver(String name) {  this.name = name;  this.available = true;  this.onDuty = false;  this.needsRepair = false;  this.car = null;  }  public void setOnDuty(boolean onDuty) {  this.onDuty = onDuty;  }  public void setAvailable(boolean available) {  this.available = available;  }  public void setNeedsRepair(boolean needsRepair) {  this.needsRepair = needsRepair;  }  public void assignCar(Car car) {  this.car = car;  }  public boolean isAvailable() {  return available;  }  public boolean isOnDuty() {  return onDuty;  }  public boolean needsRepair() {  return needsRepair;  }  public Car getCar() {  return car;  }  public String toString() {  return "{ name: " + this.name + ", available: " + this.available + ", onDuty: " + this.onDuty + ", needsRepair: " + this.needsRepair + (this.car != null ? ", car: " + this.car.toString() : "") + "}";  }  }  class Car {  private String mark;  private String model;  private int year;  private boolean needsRepair;  public Car(String mark, String model, int year) {  this.mark = mark;  this.model = model;  this.year = year;  this.needsRepair = false;  }  public void setNeedsRepair(boolean needsRepair) {  this.needsRepair = needsRepair;  }  public boolean needsRepair() {  return needsRepair;  }  public String getMark() {  return mark;  }  public String getModel() {  return model;  }  public int getYear() {  return year;  }  public String toString() {  return "{ mark: " + this.mark + ", model: " + this.model + ", year: " + this.year + ", needsRepair: " + this.needsRepair;  }  }  class Dispatcher {  private ArrayList<Driver> drivers;  private ArrayList<Car> cars;  private String result;  public Dispatcher() {  this.drivers = new ArrayList<>();  this.cars = new ArrayList<>();  }  public void addDriver(Driver driver) {  drivers.add(driver);  }  public void removeDriver(Driver driver) {  drivers.remove(driver);  }  public void addCar(Car car) {  cars.add(car);  }  public void removeCar(Car car) {  cars.remove(car);  }  public void assignDriverToCar(Driver driver, Car car) {  if (driver.getCar() != null) {  driver.getCar().setNeedsRepair(true);  }  driver.assignCar(car);  driver.setAvailable(false);  }  public void markTripComplete(Driver driver) {  driver.setAvailable(true);  driver.getCar().setNeedsRepair(false);  }  public void markDriverForRepair(Driver driver) {  driver.setNeedsRepair(true);  driver.setAvailable(false);  }  public void suspendDriver(Driver driver) {  driver.setOnDuty(false);  driver.setAvailable(false);  driver.getCar().setNeedsRepair(false);  }  public void reinstateDriver(Driver driver) {  driver.setOnDuty(true);  }  public String getDrivers() {  result = "";  this.drivers.forEach(driver -> result += driver.toString() + "\n");  return result;  }  public String getCars() {  result = "";  this.cars.forEach(car -> result += car.toString() + "\n");  return result;  }  } |