**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Информационных технологий |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Информационные системы и технологии |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Лабораторный практикум № 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **по дисциплине** |  | Программирование на JavaScript | | |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | **Алексеев Кирилл Алексеевич** |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | **ДБИ-301рки** |  |
|  |  |  |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Балабаев Данила Игоревич |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2024 г.**

|  |  |
| --- | --- |
| **Задание 1. Пример 1:** | |
| public class SimpleMath {  public static void main(String[] args) {  double x = 14.26;  double y = -1.22;  double z = 3.5e-2;  double numerator = 2\*Math.cos(x-(2.0/3.0)) \* (1+(z\*z) / (3-(z\*z) / 5.0));  double denominator = (0.5) + Math.sin(y) \* Math.sin(y);  double s = numerator/denominator;  System.out.println("s = " + s);  }  } |  |
| **Задание 2. Пример 1:** | |
| public class ANAL {  public static double f(double x){  return x;  }    public static void main(String[] args) {    double x = 2.0;  double y = 3.0;  double a; // хранить результат  double product\_xy = x\*y;  if (product\_xy > 0){  double fx = f(x);  a = Math.pow((fx + y), 2) - Math.cbrt(Math.abs(fx));  }else if (product\_xy < 0){  double fx = f(x);  a = Math.pow((fx + y), 2) + Math.sin(x);  }else{  double fx = f(x);  a = Math.pow((fx + y), 2) + Math.pow(y, 3);  }  System.out.println("Для x = " + x + ", y = " + y + ":");  System.out.println("Для xy = " + product\_xy);  System.out.println("Значение a = " + a);  }  } |  |
| **Задание 3. Пример 1:** | |
| import java.math.BigDecimal;  import java.math.MathContext; // Для указания точности  public class SeriesAndFunctionComparison {  public static void main(String[] args) {  double a = 0.1;  double b = 1.0;  int numSteps = 10;  double h = (b - a) / numSteps;  int nMax = 160;    MathContext mc = new MathContext(30);  System.out.println("------------------------------------------------------------------");  System.out.printf("%-10s %-25s %-25s %-15s\n", "x", "S(x) (ряд)", "Y(x) (sin(x))", "Разница");  System.out.println("------------------------------------------------------------------");  for (double x = a; x <= b + h / 2; x += h) {  double y\_x = Math.sin(x);  BigDecimal s\_x\_big = calculateSineSeriesBigDecimal(BigDecimal.valueOf(x), nMax, mc);  double s\_x = s\_x\_big.doubleValue();  double difference = Math.abs(y\_x - s\_x);  System.out.printf("%-10.5f %-25.20f %-25.20f %-15.10f\n", x, s\_x, y\_x, difference);  }  System.out.println("------------------------------------------------------------------");  }  public static BigDecimal calculateSineSeriesBigDecimal(BigDecimal x, int nMax, MathContext mc) {  BigDecimal sum = BigDecimal.ZERO;  BigDecimal power\_x = x;  BigDecimal factorial\_val = BigDecimal.ONE;  int sign = 1;  // n = 0: x^(1) / 1!  sum = sum.add(x);  for (int n = 1; n <= nMax; n++) {  sign \*= -1;  // Обновляем степень x: x^(2n+1) = x^(2n-1) \* x^2  power\_x = power\_x.multiply(x).multiply(x, mc);  // Обновляем факториал: (2n+1)! = (2n-1)! \* (2n) \* (2n+1)  BigDecimal twoN = BigDecimal.valueOf(2L \* n);  BigDecimal twoNPlusOne = BigDecimal.valueOf(2L \* n + 1);  factorial\_val = factorial\_val.multiply(twoN, mc).multiply(twoNPlusOne, mc);  BigDecimal term = power\_x.divide(factorial\_val, mc);  if (sign == -1) {  sum = sum.subtract(term);  } else {  sum = sum.add(term);  }  }  return sum;  }  } |  |
| **Задание 4. Пример 1:** | |
| import java.util.Scanner;  public class SimpleCharArraySort {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);    System.out.print("Введите строку символов (например, a1b2c): ");  String inputString = scanner.nextLine();  char[] charArray = inputString.toCharArray();    StringBuilder digits = new StringBuilder();  StringBuilder others = new StringBuilder();    for (char c : charArray) {  if (Character.isDigit(c)) {  digits.append(c);  } else {  others.append(c);  }  }    String resultString = digits.toString() + others.toString();    char[] resultArray = resultString.toCharArray();    System.out.println("Исходный массив символов: " + new String(charArray));  System.out.println("Преобразованный массив символов (строка): " + resultString);  System.out.println("Преобразованный массив символов (char[]): " + new String(resultArray));  scanner.close();  }  } |  |
| **Задание 5. Пример 1:** | |
| import java.util.Random;  import java.util.Scanner;  public class SimpleMatrixTask {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Random random = new Random();    System.out.print("Введите количество строк N: ");  int N = scanner.nextInt();  System.out.print("Введите количество столбцов M: ");  int M = scanner.nextInt();    int[][] A = new int[N][M];    System.out.println("\nИсходная матрица A:");  for (int i = 0; i < N; i++) {  for (int j = 0; j < M; j++) {  A[i][j] = random.nextInt(2);  System.out.printf("%3d", A[i][j]);  }  System.out.println();  }    int[] B = new int[M];    for (int j = 0; j < M; j++) {  boolean allZerosInColumn = true;  for (int i = 0; i < N; i++) {  if (A[i][j] != 0) {  allZerosInColumn = false;  break;  }  }  if (allZerosInColumn) {  B[j] = 0;  } else {  B[j] = 1;  }  }    System.out.print("\nПолученный массив B: [");  for (int i = 0; i < B.length; i++) {  System.out.print(B[i]);  if (i < B.length - 1) {  System.out.print(", ");  }  }  System.out.println("]");  scanner.close();  }  } |  |
| **Задание 6. Пример 1:** | |
| import java.util.Scanner;  public class SimpleBinaryGroupCounter {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);    System.out.println("Введите строку с группами нулей и единиц, разделенными пробелами:");  System.out.println("Например: 00110 10101 11111 000 101010");  String inputString = scanner.nextLine();  int countFiveSymbolGroups = 0;  String[] groups = inputString.split("\\s+");    for (String group : groups) {    if (!group.isEmpty()) {    if (group.length() == 5) {  countFiveSymbolGroups++;  }  }  }    System.out.println("\nИсходная строка: " + inputString);  System.out.println("Количество групп с пятью символами: " + countFiveSymbolGroups);  scanner.close();  }  } |  |
| **Задание 7. Пример 1:** | |
| import java.util.ArrayList;  import java.util.HashSet;  import java.util.List;  import java.util.Objects;  import java.util.Set;  class Client {  String fullName;  String address;  double discount;  public Client(String fullName, String address, double discount) {  this.fullName = fullName;  this.address = address;  this.discount = discount;  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (o == null || getClass() != o.getClass()) return false;  Client client = (Client) o;  return Objects.equals(fullName, client.fullName) &&  Objects.equals(address, client.address);  }  @Override  public int hashCode() {  return Objects.hash(fullName, address);  }  @Override  public String toString() {  return "ФИО: '" + fullName + "', Адрес: '" + address + "', Скидка: " + discount + "%";  }  }  public class RemoveDuplicateClients {  public static void main(String[] args) {  List<Client> originalClients = new ArrayList<>();  originalClients.add(new Client("Иванов И.И.", "ул. Ленина, 1", 10.0));  originalClients.add(new Client("Петров П.П.", "ул. Пушкина, 5", 5.0));  originalClients.add(new Client("Иванов И.И.", "ул. Ленина, 1", 10.0)); // Дубликат (ФИО и адрес)  originalClients.add(new Client("Сидоров С.С.", "пр. Мира, 10", 15.0));  originalClients.add(new Client("Иванов И.И.", "ул. Ленина, 1", 12.0)); // Дубликат (ФИО и адрес), но другая скидка  originalClients.add(new Client("Петров П.П.", "пр. Гагарина, 3", 8.0)); // Не дубликат (другой адрес)  System.out.println("--- Исходный список клиентов (" + originalClients.size() + " записей) ---");  for (Client client : originalClients) {  System.out.println(client);  }  Set<Client> uniqueClientsSet = new HashSet<>(originalClients);  List<Client> distinctClients = new ArrayList<>(uniqueClientsSet);  System.out.println("\n--- Список клиентов без дубликатов (" + distinctClients.size() + " записей) ---");  for (Client client : distinctClients) {  System.out.println(client);  }  }  } |  |
| **Задание 8. Пример 1:** | |
| import java.math.BigDecimal;  import java.math.MathContext;  public class SeriesWithAccuracy {    public static double calculateY(double x) {    if (x >= 1.0 || x <= -1.0) {  System.err.println("Ошибка: Для Y(x) аргумент x должен быть в диапазоне (-1, 1).");  return Double.NaN;  }  return 0.5 \* Math.log((1.0 + x) / (1.0 - x));  }    public static double[] calculateS(double x, double epsilon) {    if (Math.abs(x) >= 1.0) {  System.err.println("Ошибка: Для S(x) аргумент x должен быть в диапазоне (-1, 1).");  return new double[]{Double.NaN, 0};  }  double sum = 0.0;  double term = x;  int k = 0;    sum += term;    while (Math.abs(term) >= epsilon && k < 10000) {  k++;      double oldDenominator = 2.0 \* (k-1) + 1;  double newDenominator = 2.0 \* k + 1;    if (oldDenominator != 0) {  term = term \* x \* x \* (oldDenominator / newDenominator);  } else {  term = Math.pow(x, 2.0 \* k + 1) / (2.0 \* k + 1);  }    sum += term;  }  return new double[]{sum, k};  }  public static void main(String[] args) {  double a = -0.9;  double b = 0.9;  int numSteps = 10;  double h = (b - a) / numSteps;  double epsilon = 0.0001;  System.out.println("-------------------------------------------------------------------------------------------------");  System.out.printf("%-10s %-20s %-20s %-20s %-10s\n", "x", "S(x) (ряд)", "Y(x) (функция)", "Разница", "Итерации k");  System.out.println("-------------------------------------------------------------------------------------------------");    for (double x = a; x <= b + h / 2; x += h) {  double y\_x = calculateY(x);  double[] s\_result = calculateS(x, epsilon);  double s\_x = s\_result[0];  int iterations = (int) s\_result[1];  double difference = Math.abs(y\_x - s\_x);  System.out.printf("%-10.5f %-20.10f %-20.10f %-20.10f %-10d\n", x, s\_x, y\_x, difference, iterations);  }  System.out.println("-------------------------------------------------------------------------------------------------");  }  } |  |
| **Задание 9. Пример 1:** | |
| import java.util.Arrays;  import java.util.Random;  public class AverageCalculator {  public static void main(String[] args) {  Random random = new Random();      int arraySize = 5;  int[] array = new int[arraySize];  for (int i = 0; i < arraySize; i++) {  array[i] = random.nextInt(100);  }  System.out.println("Исходный массив: " + Arrays.toString(array));    double averageIterative = calculateAverageIterative(array);  System.out.println("Среднее значение (итеративно): " + averageIterative);    if (array.length == 0) {  System.out.println("Массив пуст, среднее значение (рекурсивно) не определено.");  } else {  double sumRecursive = calculateSumRecursive(array, array.length - 1);  double averageRecursive = sumRecursive / array.length;  System.out.println("Среднее значение (рекурсивно): " + averageRecursive);  }  }    public static double calculateAverageIterative(int[] arr) {  if (arr == null || arr.length == 0) {  return 0.0;  }  int sum = 0;  for (int num : arr) {  sum += num;  }  return (double) sum / arr.length;  }    public static int calculateSumRecursive(int[] arr, int index) {    if (index == 0) {  return arr[0];  }    return arr[index] + calculateSumRecursive(arr, index - 1);  }  } |  |
| **Задание 10. Пример 1:** | |
| import java.time.LocalDate; import java.time.format.DateTimeFormatter; import java.util.\*;  class Product { private String name; private int quantity; private double price; private LocalDate arrivalDate;  public Product(String name, int quantity, double price, LocalDate arrivalDate) {  this.name = name;  this.quantity = quantity;  this.price = price;  this.arrivalDate = arrivalDate; }  public double getTotalCost() {  return quantity \* price; }  public boolean isOlderThanMonth() {  return arrivalDate.plusMonths(1).isBefore(LocalDate.now()); }  @Override public String toString() {  return String.format("Товар: %s | Кол-во: %d | Цена: %.2f | Дата: %s | Стоимость: %.2f",  name, quantity, price, arrivalDate, getTotalCost()); }  }  public class WarehouseSimple { public static void main(String[] args) { Scanner scanner = new Scanner(System.in); DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy"); List products = new ArrayList<>();  System.out.print("Сколько товаров ввести? ");  int n = scanner.nextInt();  scanner.nextLine(); // очистка буфера   for (int i = 0; i < n; i++) {  System.out.println("Введите данные для товара " + (i + 1));  System.out.print("Наименование: ");  String name = scanner.nextLine();   System.out.print("Количество: ");  int qty = scanner.nextInt();   System.out.print("Цена: ");  double price = scanner.nextDouble();  scanner.nextLine(); // очистка   System.out.print("Дата поступления (dd.MM.yyyy): ");  String dateStr = scanner.nextLine();  LocalDate date = LocalDate.parse(dateStr, formatter);   products.add(new Product(name, qty, price, date));  }   System.out.println("\nВсе товары:");  products.forEach(System.out::println);   System.out.println("\nТовары старше месяца и стоимостью > 1 000 000:");  products.stream()  .filter(p -> p.isOlderThanMonth() && p.getTotalCost() > 1\_000\_000)  .forEach(System.out::println); }  } |  |
| **Задание 11. Пример 1:** | |
| import java.time.LocalDate;  import java.time.format.DateTimeFormatter;  import java.util.Scanner;  class Product {  String name;  int quantity;  double price;  LocalDate arrivalDate;  Product(String name, int quantity, double price, LocalDate arrivalDate) {  this.name = name;  this.quantity = quantity;  this.price = price;  this.arrivalDate = arrivalDate;  }  @Override  public String toString() {  return String.format("Товар: %s | Кол-во: %d | Цена: %.2f | Дата: %s",  name, quantity, price, arrivalDate);  }  }  public class WarehouseSort {  // ======== Быстрая сортировка (QuickSort) ========  public static void quickSort(Product[] arr, int low, int high) {  if (low < high) {  int pi = partition(arr, low, high);  quickSort(arr, low, pi - 1);  quickSort(arr, pi + 1, high);  }  }  private static int partition(Product[] arr, int low, int high) {  double pivot = arr[high].price; // опорный элемент  int i = (low - 1);  for (int j = low; j < high; j++) {  if (arr[j].price <= pivot) {  i++;  swap(arr, i, j);  }  }  swap(arr, i + 1, high);  return i + 1;  }  // ======== Сортировка выбором ========  public static void selectionSort(Product[] arr) {  for (int i = 0; i < arr.length - 1; i++) {  int minIndex = i;  for (int j = i + 1; j < arr.length; j++) {  if (arr[j].price < arr[minIndex].price) {  minIndex = j;  }  }  swap(arr, i, minIndex);  }  }  // ======== Обмен элементов ========  private static void swap(Product[] arr, int i, int j) {  Product temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  // ======== Основная программа ========  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");  System.out.print("Сколько товаров ввести? ");  int n = scanner.nextInt();  scanner.nextLine();  Product[] products = new Product[n];  for (int i = 0; i < n; i++) {  System.out.println("Введите данные для товара " + (i + 1));  System.out.print("Наименование: ");  String name = scanner.nextLine();  System.out.print("Количество: ");  int qty = scanner.nextInt();  System.out.print("Цена: ");  double price = scanner.nextDouble();  scanner.nextLine();  System.out.print("Дата поступления (dd.MM.yyyy): ");  String dateStr = scanner.nextLine();  LocalDate date = LocalDate.parse(dateStr, formatter);  products[i] = new Product(name, qty, price, date);  }  // Копия массива для второй сортировки  Product[] productsQuick = products.clone();  Product[] productsSelect = products.clone();  // Сортировка QuickSort  quickSort(productsQuick, 0, productsQuick.length - 1);  System.out.println("\nСортировка QuickSort по цене:");  for (Product p : productsQuick) {  System.out.println(p);  }  // Сортировка выбором  selectionSort(productsSelect);  System.out.println("\nСортировка выбором по цене:");  for (Product p : productsSelect) {  System.out.println(p);  }  }  } |  |
| **Задание 12. Пример 1:** | |
| import java.time.LocalDate;  import java.time.format.DateTimeFormatter;  import java.util.Scanner;  class Product {  String name;  int quantity;  double price;  LocalDate arrivalDate;  Product(String name, int quantity, double price, LocalDate arrivalDate) {  this.name = name;  this.quantity = quantity;  this.price = price;  this.arrivalDate = arrivalDate;  }  @Override  public String toString() {  return String.format("Товар: %s | Кол-во: %d | Цена: %.2f | Дата: %s",  name, quantity, price, arrivalDate);  }  }  public class WarehouseSearch {  // ---------- Быстрая сортировка ----------  public static void quickSort(Product[] arr, int low, int high) {  if (low < high) {  int pi = partition(arr, low, high);  quickSort(arr, low, pi - 1);  quickSort(arr, pi + 1, high);  }  }  private static int partition(Product[] arr, int low, int high) {  double pivot = arr[high].price;  int i = (low - 1);  for (int j = low; j < high; j++) {  if (arr[j].price <= pivot) {  i++;  swap(arr, i, j);  }  }  swap(arr, i + 1, high);  return i + 1;  }  private static void swap(Product[] arr, int i, int j) {  Product temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  // ---------- Линейный поиск ----------  public static Product linearSearch(Product[] arr, double keyPrice) {  for (Product p : arr) {  if (p.price == keyPrice) {  return p;  }  }  return null;  }  // ---------- Двоичный поиск ----------  public static Product binarySearch(Product[] arr, double keyPrice) {  int left = 0, right = arr.length - 1;  while (left <= right) {  int mid = (left + right) / 2;  if (arr[mid].price == keyPrice) {  return arr[mid];  } else if (arr[mid].price < keyPrice) {  left = mid + 1;  } else {  right = mid - 1;  }  }  return null;  }  // ---------- Главная программа ----------  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");  System.out.print("Сколько товаров ввести? ");  int n = scanner.nextInt();  scanner.nextLine();  Product[] products = new Product[n];  for (int i = 0; i < n; i++) {  System.out.println("Введите данные для товара " + (i + 1));  System.out.print("Наименование: ");  String name = scanner.nextLine();  System.out.print("Количество: ");  int qty = scanner.nextInt();  System.out.print("Цена: ");  double price = scanner.nextDouble();  scanner.nextLine();  System.out.print("Дата поступления (dd.MM.yyyy): ");  String dateStr = scanner.nextLine();  LocalDate date = LocalDate.parse(dateStr, formatter);  products[i] = new Product(name, qty, price, date);  }  // Сортировка массива по цене (чтобы работал двоичный поиск)  quickSort(products, 0, products.length - 1);  System.out.println("\nОтсортированный массив:");  for (Product p : products) {  System.out.println(p);  }  double searchPrice = 150000;  // Линейный поиск  Product foundLinear = linearSearch(products, searchPrice);  System.out.println("\nЛинейный поиск (цена " + searchPrice + "): " +  (foundLinear != null ? foundLinear : "Товар не найден."));  // Двоичный поиск  Product foundBinary = binarySearch(products, searchPrice);  System.out.println("Двоичный поиск (цена " + searchPrice + "): " +  (foundBinary != null ? foundBinary : "Товар не найден."));  }  } |  |
| **Задание 13. Пример 1:** | |
| import java.util.Random;  import java.util.Stack;  public class StackSplit {  public static void main(String[] args) {  Random rand = new Random();  // Исходный стек  Stack<Integer> original = new Stack<>();  // Заполняем случайными числами из диапазона [-50; 50]  for (int i = 0; i < 20; i++) { // можно изменить количество элементов  int num = rand.nextInt(101) - 50; // от -50 до +50  original.push(num);  }  System.out.println("Исходный стек: " + original);  // Два новых стека  Stack<Integer> positiveStack = new Stack<>();  Stack<Integer> negativeStack = new Stack<>();  // Разделение стека  while (!original.isEmpty()) {  int value = original.pop();  if (value > 0) {  positiveStack.push(value);  } else if (value < 0) {  negativeStack.push(value);  }  // если 0 — игнорируем  }  System.out.println("Стек с положительными: " + positiveStack);  System.out.println("Стек с отрицательными: " + negativeStack);  // Очистка стеков (удаление)  positiveStack.clear();  negativeStack.clear();  original.clear();  System.out.println("\nПосле удаления:");  System.out.println("Исходный стек: " + original);  System.out.println("Положительный стек: " + positiveStack);  System.out.println("Отрицательный стек: " + negativeStack);  }  } |  |
| **Задание 14. Пример 1:** | |
| import java.util.LinkedList;  import java.util.Queue;  import java.util.Random;  public class QueueEvenRemove {  public static void main(String[] args) {  Random rand = new Random();  // Создаём очередь  Queue<Integer> queue = new LinkedList<>();  // Заполняем случайными числами из диапазона [-50; 50]  for (int i = 0; i < 20; i++) { // можно поменять количество элементов  int num = rand.nextInt(101) - 50; // от -50 до +50  queue.add(num);  }  System.out.println("Исходная очередь: " + queue);  // Создаём новую очередь без чётных чисел  Queue<Integer> filteredQueue = new LinkedList<>();  for (int value : queue) {  if (value % 2 != 0) { // только нечётные  filteredQueue.add(value);  }  }  // Перезаписываем очередь  queue = filteredQueue;  System.out.println("Очередь без чётных чисел: " + queue);  // Очистка очереди (удаление)  queue.clear();  System.out.println("\nПосле удаления:");  System.out.println("Очередь: " + queue);  }  } |  |
| **Задание 15. Пример 1:** | |
| import java.util.Collections;  import java.util.LinkedList;  import java.util.Random;  public class DoublyLinkedListTask {  public static void main(String[] args) {  Random rand = new Random();  // 1. Создаём двунаправленный список  LinkedList<Integer> list = new LinkedList<>();  // Заполняем числами из диапазона [-50; 50]  for (int i = 0; i < 15; i++) { // можно изменить количество элементов  list.add(rand.nextInt(101) - 50);  }  System.out.println("Исходный список: " + list);  // 2. Сортировка списка  Collections.sort(list);  System.out.println("Отсортированный список: " + list);  // 3. Поиск элемента (номер варианта)  int variant = 7; // <-- здесь твой номер варианта  int index = list.indexOf(variant);  if (index != -1) {  System.out.println("Элемент " + variant + " найден на позиции: " + index);  } else {  System.out.println("Элемент " + variant + " в списке отсутствует.");  }  // 4. Найти минимальный элемент и сделать его первым  int minValue = Collections.min(list);  list.remove(Integer.valueOf(minValue));  list.addFirst(minValue);  System.out.println("Список после перемещения минимума в начало: " + list);  // 5. Очистка списка (удаление)  list.clear();  System.out.println("\nПосле удаления список: " + list);  }  } |  |
| **Задание 16. Пример 1:** | |
| class AVLTree {  static class Node {  int key;  Node left, right;  int height;  Node(int key) {  this.key = key;  this.height = 1;  }  }  private Node root;  // -------------------- Вспомогательные функции --------------------  private int height(Node n) {  return (n == null) ? 0 : n.height;  }  private int getBalance(Node n) {  return (n == null) ? 0 : height(n.left) - height(n.right);  }  private Node rightRotate(Node y) {  Node x = y.left;  Node T2 = x.right;  x.right = y;  y.left = T2;  y.height = Math.max(height(y.left), height(y.right)) + 1;  x.height = Math.max(height(x.left), height(x.right)) + 1;  return x;  }  private Node leftRotate(Node x) {  Node y = x.right;  Node T2 = y.left;  y.left = x;  x.right = T2;  x.height = Math.max(height(x.left), height(x.right)) + 1;  y.height = Math.max(height(y.left), height(y.right)) + 1;  return y;  }  // -------------------- Добавление --------------------  public void insert(int key) {  root = insert(root, key);  }  private Node insert(Node node, int key) {  if (node == null) return new Node(key);  if (key < node.key) node.left = insert(node.left, key);  else if (key > node.key) node.right = insert(node.right, key);  else return node; // одинаковые ключи не вставляем  node.height = 1 + Math.max(height(node.left), height(node.right));  int balance = getBalance(node);  // Левый-левый случай  if (balance > 1 && key < node.left.key) return rightRotate(node);  // Правый-правый случай  if (balance < -1 && key > node.right.key) return leftRotate(node);  // Левый-правый случай  if (balance > 1 && key > node.left.key) {  node.left = leftRotate(node.left);  return rightRotate(node);  }  // Правый-левый случай  if (balance < -1 && key < node.right.key) {  node.right = rightRotate(node.right);  return leftRotate(node);  }  return node;  }  // -------------------- Удаление --------------------  public void delete(int key) {  root = delete(root, key);  }  private Node delete(Node node, int key) {  if (node == null) return null;  if (key < node.key) node.left = delete(node.left, key);  else if (key > node.key) node.right = delete(node.right, key);  else {  if (node.left == null || node.right == null) {  Node temp = (node.left != null) ? node.left : node.right;  if (temp == null) {  node = null;  } else {  node = temp;  }  } else {  Node temp = minValueNode(node.right);  node.key = temp.key;  node.right = delete(node.right, temp.key);  }  }  if (node == null) return null;  node.height = 1 + Math.max(height(node.left), height(node.right));  int balance = getBalance(node);  if (balance > 1 && getBalance(node.left) >= 0) return rightRotate(node);  if (balance > 1 && getBalance(node.left) < 0) {  node.left = leftRotate(node.left);  return rightRotate(node);  }  if (balance < -1 && getBalance(node.right) <= 0) return leftRotate(node);  if (balance < -1 && getBalance(node.right) > 0) {  node.right = rightRotate(node.right);  return leftRotate(node);  }  return node;  }  // -------------------- Поиск --------------------  public boolean search(int key) {  return search(root, key);  }  private boolean search(Node node, int key) {  if (node == null) return false;  if (key == node.key) return true;  return key < node.key ? search(node.left, key) : search(node.right, key);  }  // -------------------- Обходы --------------------  public void inOrder() {  inOrder(root);  System.out.println();  }  private void inOrder(Node node) {  if (node != null) {  inOrder(node.left);  System.out.print(node.key + " ");  inOrder(node.right);  }  }  public void preOrder() {  preOrder(root);  System.out.println();  }  private void preOrder(Node node) {  if (node != null) {  System.out.print(node.key + " ");  preOrder(node.left);  preOrder(node.right);  }  }  public void postOrder() {  postOrder(root);  System.out.println();  }  private void postOrder(Node node) {  if (node != null) {  postOrder(node.left);  postOrder(node.right);  System.out.print(node.key + " ");  }  }  // -------------------- Минимум и максимум --------------------  private Node minValueNode(Node node) {  Node current = node;  while (current.left != null) current = current.left;  return current;  }  private Node maxValueNode(Node node) {  Node current = node;  while (current.right != null) current = current.right;  return current;  }  // -------------------- Обмен макс и мин --------------------  public void swapMinMax() {  if (root == null) return;  Node minNode = minValueNode(root);  Node maxNode = maxValueNode(root);  int temp = minNode.key;  minNode.key = maxNode.key;  maxNode.key = temp;  }  // -------------------- Очистка --------------------  public void clear() {  root = null; // Java очистит память автоматически через GC  }  }  // -------------------- Тестирование --------------------  public class Main {  public static void main(String[] args) {  AVLTree tree = new AVLTree();  // Добавляем числа от -50 до +50  for (int i = -50; i <= 50; i += 5) { // например через 5, чтобы не перегружать дерево  tree.insert(i);  }  System.out.println("Прямой обход (preOrder):");  tree.preOrder();  System.out.println("Обратный обход (postOrder):");  tree.postOrder();  System.out.println("Обход в порядке возрастания (inOrder):");  tree.inOrder();  // Поиск  System.out.println("Поиск числа 25: " + tree.search(25));  System.out.println("Поиск числа 100: " + tree.search(100));  // Удаление  tree.delete(0);  System.out.println("После удаления 0 (inOrder):");  tree.inOrder();  // Обмен местами макс и мин  tree.swapMinMax();  System.out.println("После обмена местами мин и макс (inOrder):");  tree.inOrder();  // Очистка дерева  tree.clear();  }  } |  |