Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Уральский государственный университет имени первого президента России
Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий
Центр ускоренного обучения

Отчёт по лабораторной работе №8

«Введение в алгоритмы и структуры данных Java»

Руководитель ст. преподаватель — Н. А. Архипов

Студент гр. РИЗ-100028у — И. С. Арсентьев

Екатеринбург 2021

Лабораторная работа №8.

«Введение в наследование»

Цель: приобретение навыков работы с рекурсивными методами, знакомство с динамическими структурами данных, приобретение навыков создания и использования простейшей динамической структуры.

Описание задачи:

Составить 6 программ по представленным задачам, представить листинги программ, во вложении к отчёту приложить файлы готовых классов.

Ход выполнения задач:

1. Создать приложения для демонстрации примеров 1 – 5 из раздела 1. Для примера 5 дополнительно вывести последовательность обхода дерева рекурсивных вызовов. Отработать код с помощью отладчика и привести скриншоты минимум трех точек, обработанных отладчиком.

```
package example_02_81;
public class Rec1 {
   public static void m(int x) {
      System.out.println("x="+x);
      if ( (2*x+1) <20) {
         m(2*x+1);
      }
   }
   public static void main(String[] args) {
      m(1);
   }
}
package example_02_81;
public class Rec2 {
   public static void m(int x){
      if ((2*x+1)<20){
         m(2*x+1);
      }
      System.out.println("x= "+ x);
   }
   public static void main(String[] args) {
      m(1);
   }
}
package example_02_81;
public class Rec3 {
   private static int step = 0;
   public static void m(int x){
      space();
```

```
        System.out.println(""+x+"->");
        step++;
        if ((2*x+1)<20){
            m(2*x+1);
        }
        step--;
        space();
        System.out.println(""+x+"<-");
    }
    public static void space (){
        for (int i = 0; i < step; i++){
            System.out.println(" ");
        }
    }
    public static void main(String[] args) {
        m(1);
    }
}
package example_02_81;
import java.util.Scanner;
public class Rec4 {
    public static int fact(int n){
        int result;
        if (n==1){
            return 1;
        }else {
            result = fact(n-1)*n;
            return result;
        }
    }
    public static void main(String[] args) {
        Scanner inCMD = new Scanner(System.in);
        System.out.print("Введите число: ");
        int num = inCMD.nextInt();
        System.out.println(fact(num));
    }
}
package example_02_81;
import java.util.Scanner;
public class Rec5 {
    public static int f(int n){
        if (n==0){
            return 0;
        }else {
            if (n==1){
                return 1;
            }else {
                return f(n-2)+f(n-1);
            }
```

```
        }
      }
      public static void main(String[] args) {
        Scanner inCMD = new Scanner(System.in);
        System.out.print("Введите число: ");
        int num = inCMD.nextInt();
        System.out.println(f(num));
      }
    }
```

2. Создать приложение с использованием рекурсии для перевода целого числа, введенного с клавиатуры, в двоичную систему счисления.

```
import java.util.Scanner;

public class example_02_82_Main {
  public static void main(String[] args) {
    Scanner inCMD = new Scanner(System.in);
    System.out.print("Введите десятичное число ");
    int num = inCMD.nextInt();
    example07_02_Rec REC = new example07_02_Rec();
    REC.setNum(num);
    REC.getNum();
  }
}
//метод рекурсии для вычисления значения двоичного кода
public class example_02_82_Rec{
  //переменные для работы - внешние
  private int num;
  private int arra[];
  //переменные для работы - внутренние
  private int i = 0;
  public void setNum(int num) {
    this.num = num;
    this.arra = new int[this.num];
    Translate(this.num, this.arra, this.i);
  }
  public int getNum() {
    for (int q = this.i; q >= 0; q--){
      System.out.print(arra[q]);
    }
    return num;
  }
  private void Translate (int num, int [] arra, int i){
    int buf =num / 2;
    arra[this.i] = num - buf * 2;
    if (num - buf == 1 || num - buf == 0){
      return;
    }
    num = buf;
    this.i++;
```

```
        Translate(num,arra, this.i);
    }
}
```

3. Создать приложение, позволяющее ввести и вывести одномерный массив целых чисел. Для ввода и вывода массива разработать рекурсивные методы вместо циклов for.

```
import java.util.Scanner;
public class example_02_83_Main {
    public static void main(String[] args) {
        Scanner inCMD = new Scanner(System.in);
        System.out.print("Введите количество элементов ");
        int num = inCMD.nextInt();
        example_02_83_Rec Obj = new example_02_83_Rec();
        Obj.setNum(num);
        System.out.println("Вывод массива");
        Obj.getNum();
    }
}
//метод рекурсии для ввода элементов массива
import java.util.Scanner;
public class example_02_83_Rec {
    private int num;
    private int arra [];
    private int i = 0;
    public void setNum(int num) {
        this.num = num;
        this.arra = new int[this.num];
        Rec_In(this.arra, this.i);
    }
    public int getNum() {
        Rec_Out(this.arra, this.i);
        return num;
    }
    private void Rec_In (int[] arra, int i){
        Scanner inCMD = new Scanner(System.in);
        System.out.print("Введите элемент ");
        int num = inCMD.nextInt();
        this.arra[this.i] = num;
        this.i++;
        if (this.i >= this.num){
            return;
        }
        Rec_In(this.arra,this.i);
    }
    private void Rec_Out (int[] arra, int i ){
        if (i <= 0 ){
            return;
        }
        this.i--;
```

```
      System.out.print(this.arra[this.i]);
      Rec_Out(this.arra, this.i);
   }
}
```

4. Выполнить пример 1 из раздела 2. Отработать код с помощью отладчика и привести скриншоты минимум трех точек обработанных отладчиком.

```
package example_02_84;
public class din {
   public static void main(String[] args) {
// создание несвязанных узлов с помощью конструктора
      Node node0 = new Node(0, null); // 0-й узел – будет головой в списке
      Node node1 = new Node(1, null);
      Node node2 = new Node(2, null);
      Node node3 = new Node(3, null); // последний узел – будет хвостом в списке
// связывание узлов в список с помощью ссылок
      node0.next = node1;
      node1.next = node2;
      node2.next = node3;
// вывод списка с использованием вспомогательной переменной ref,
// соответствующей текущему значению ссылки при прохождении по списку
      Node ref = node0; // для перемещения по списку достаточно помнить голову
      while (ref != null) {
         System.out.print(" " + ref.value);
         ref = ref.next;
      } }}
// КЛАСС – СТРУКТУРА ЭЛЕМЕНТА СПИСКА
package example_02_84;
public class Node { // КЛАСС – СТРУКТУРА ЭЛЕМЕНТА СПИСКА
   public int value; // значение
   public Node next; // поле – ссылка (указатель) на следующий узел
   Node(int value, Node next) { // конструктор класса
      this.value = value;
      this.next = next;
   }}
```

5. Создать два проекта, в которых продемонстрировать два способа создания линейного однонаправленного списка (с головы и с хвоста) согласно примеру 2 из второго раздела. Отработать код с помощью отладчика и привести скриншоты минимум трех точек обработанных отладчиком.

```
package example_02_85;
public class DSD_create {
   public static void main(String[] args) {
      // создание 1-го узла, который изначально является и головой, и хвостом списка
      Node head=new Node(0, null);
      Node tail=head;
// добавление элементов с наращиванием хвоста
      for (int i = 0; i <9; i++) {
         tail.next=new Node(i+1, null);
         tail=tail.next; // указатель на созданный элемент запоминается
```

```
      } // как указатель на новый хвост
// вывод элементов на экран
      Node ref = head; // ref – рабочая переменная для текущего узла
      while (ref != null) {
         System.out.print(" " + ref.value);
         ref = ref.next;
      }}}
package example_02_85;
public class Node {
   public int value;
   public Node next;
   Node(int value, Node next) { // конструктор
      this.value = value;
      this.next = next;
   }
}
```

Вывод:

В ходе работы был ознакомлен с навыками работы с рекурсивными методами, с динамическими структурами данных, с навыками создания и использования простейшей динамической структуры.

Каждое решение задания сопровождено листингами программ, в архиве с классами будут добавлены соответствующие файлы.