|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКИМ ЗАНЯТИЯМ №5, №6**

Выполнил студент группы ИНБО-02-20 Зотов Е. А.

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва 2021 г.**

**Практическое занятие №5**

**Цель работы:** Изучение работы с рекурсией.

**Задание 12.**

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите все нечетные числа из этой последовательности, сохраняя их порядок.

В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

**Задание 13.**

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел. Завершающий ноль выводить не надо.

В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

**Задание 14.**

Дано натуральное число N. Выведите все его цифры по одной, в обычном порядке, разделяя их пробелами или новыми строками.

При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика.

**Код программы:**

**Класс Sol14**

package ru.mirea.practice5;  
  
public class Sol14 {  
 public static String recursion(int n) {  
 if (n < 10) {  
 return Integer.*toString*(n);  
 }  
 else {  
 return *recursion*(n/10) + ' ' + n % 10;  
 }  
 }  
 public static void main(String[] args) {  
 System.*out*.println(*recursion*(1234));  
 }  
}

**Класс Sol15**

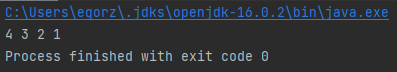
package ru.mirea.practice5;  
  
public class Sol15 {  
 public static int recursion(int n) {  
 if (n < 10) {  
 System.*out*.print(n);  
 return n;  
 }  
 else {  
 System.*out*.print(n % 10 + " ");  
 return *recursion*(n / 10);  
 }  
 }  
 public static void main(String[] args) {  
 *recursion*(1234);  
 }  
}

**Класс Sol16**

package ru.mirea.practice5;  
  
public class Sol16 {  
 public static void recursion(int max, int count) {  
 java.util.Scanner in = new java.util.Scanner(System.*in*);  
 int n = in.nextInt();  
 if (n > 0) {  
 if (n > max) {  
 *recursion*(n, 1);  
 }  
 else if (n == max) {  
 *recursion*(max, ++count);  
 }  
 else {  
 *recursion*(max, count);  
 }  
 } else {  
 System.*out*.println(count);  
 }  
 }  
 public static void main(String[] args) {  
 *recursion*(0, 0);  
 }  
}

**Тестирование**







**Практическое занятие №6**

**Цель работы:**

Целью данной практической работы освоить на практике сортировки различными методами.

**Задание 1.**

Написать тестовый класс, который создает массив класса Student и сортирует

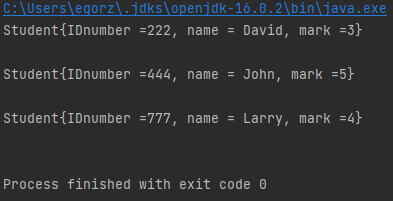
массив iDNumber.

**Код программы:**

**Класс Main**

public class Main {  
 public static void insertionSort(Student[] arr) {  
 for (int i = 1; i < arr.length; i++) {  
 Student currElem = arr[i];  
 int prevKey = i - 1;  
 while (prevKey >= 0 && arr[prevKey].getIDnumber() > currElem.getIDnumber()) {  
 arr[prevKey + 1] = arr[prevKey];  
 prevKey--;  
 }  
 arr[prevKey + 1] = currElem;  
 }  
 }  
  
 public static void main(String[] args) {  
 Student[] students = new Student[3];  
 students[0] = new Student(444, "John",5);  
 students[1] = new Student(222, "David",3);  
 students[2] = new Student(777, "Larry",4);  
 *insertionSort*(students);  
 for (Student s : students) {  
 System.*out*.println(s.toString());  
 }  
 }  
}

**Тестирование**

****

**Задание 2 и задание 3.**

Напишите класс SortingStudentsByGPA который реализует интерфейс

Comparator таким образом, что она сортирует студентов с их итоговым баллом в

порядке убывания.

Напишите программу, которая объединяет два списка данных о студентах в один отсортированный списках.

**Код Программы:**

**Класс Main**

package ru.mirea.practice6;  
  
import java.util.\*;  
import java.util.List;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Student[] students = new Student[3];  
 students[0] = new Student(123, "John",3);  
 students[1] = new Student(456, "George",4);  
 students[2] = new Student(789, "William",5);  
 *insertionSort*(students);  
 for (Student s : students) {  
 System.*out*.println(s.toString());  
 }  
 List<Student> one = new ArrayList<>();  
 one.add(students[0]);  
 one.add(students[1]);  
 one.add(students[2]);  
 Comparator<Student> k = new SortingStudentsByGPA();  
 one.sort(k);  
 for (Student s : one) {  
 System.*out*.println(s.toString());  
 }  
 List<Student> two = new ArrayList<>();  
 two.add(new Student(987,"Henry",3));  
 two.add(new Student(654,"Mark",2));  
 two.add(new Student(321,"Robert",2));  
 two.sort(k);  
 List<Student> three = new ArrayList<>();  
 int posA = 0, posB = 0;  
 for (int i = 0; i < one.size() + two.size(); i++){  
 if (posA == one.size()){  
 three.add(two.get(posB));  
 posB++;  
 } else if (posB == two.size()) {  
 three.add(one.get(posA));  
 posA++;  
 } else if (one.get(posA).getMark() > two.get(posB).getMark()) {  
 three.add(one.get(posA));  
 posA++;  
 } else {  
 three.add(two.get(posB));  
 posB++;  
 }  
 }  
 for(Student s: three) {  
 System.*out*.println(s.toString());  
 }  
  
 }  
 public static void insertionSort(Student[] arr) {  
 for (int i = 1; i < arr.length; i++) {  
 Student currElem = arr[i];  
 int prevKey = i - 1;  
 while (prevKey >= 0 && arr[prevKey].getIDnumber() > currElem.getIDnumber()) {  
 arr[prevKey + 1] = arr[prevKey];  
 prevKey--;  
 }  
 arr[prevKey + 1] = currElem;  
 }  
 }  
}

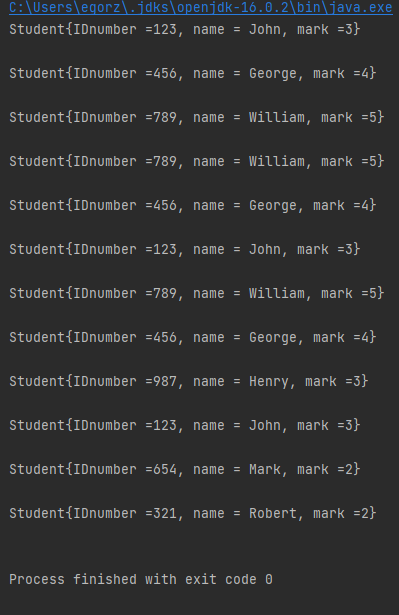
**Класс SortingStudentsByGPA**

package ru.mirea.practice6;  
  
import java.util.Comparator;  
  
public class SortingStudentsByGPA implements Comparator<Student> {  
 @Override  
 public int compare(Student a, Student b) {  
 return -(a.getMark() - b.getMark());  
 }  
}

**Класс Student**

package ru.mirea.practice6;  
  
public class Student {  
 private int IDnumber;  
 private String name;  
 private int mark;  
 public Student(int IDnumber, String name,int mark) {  
 this.IDnumber = IDnumber;  
 this.name = name;  
 this.mark = mark;  
 }  
  
 public int getIDnumber() {  
 return IDnumber;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setIDnumber(int IDnumber) {  
 this.IDnumber = IDnumber;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getMark() {  
 return mark;  
 }  
  
 public void setMark(int mark) {  
 this.mark = mark;  
 }  
  
 @Override  
 public String toString() {  
 return "Student{" +  
 "IDnumber =" + IDnumber +  
 ", name = " + name +  
 ", mark =" + mark +  
 '}' + '\n';  
 }  
}

**Тестирование**

****

**Выводы**

По ходу работы были получены навыки работы с рекурсией, а также освоение на практике различных методов сортировки.

Git: <https://github.com/kwazr/mirea-practice/tree/master/src/ru/mirea>