**Белорусский национальный технический университет**

**Кафедра «Программное обеспечение информационных систем и технологий»**

**Дисциплина «Программирование на языке Java»**

**Лабораторная работа №3**

Выполнили: студенты гр. 10702122 Тимощенко Н.В   
 Кудрявец Д.Е

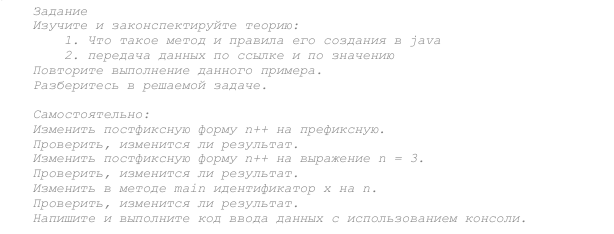
Проверил: Сидорик В. В.

Минск 2024

**Practice\_Increment**

**Цель работы**

Изучить работу оператора инремент и его префиксные и постфиксные формы

**Задание:**

**Код программы:**

Код программы Increment;

package Task1;  
*/\* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class Increment {  
 public static void main(String[] args){  
 int x = 1;  
 System.*out*.println("Before the call, x is "+ x);  
 *increment*(x); *// вызов метода increment* System.*out*.println("After the call, x is " + x);  
 }  
  
 public static void increment(int n) { *//объявление метода increment* n++;  
 System.*out*.println("n inside the method is " + n);  
 }  
}

Измененный код метода increment в программеIncrement\_v2:

public static void increment(int n) { *//объявление метода increment* ++n;  
 System.*out*.println("n inside the method is " + n);  
}

Измененный код метода increment в программеIncrement\_v3;

public static void increment(int n) { *//объявление метода increment* n = 3;  
 System.*out*.println("n inside the method is " + n);  
}

Измененный код программы Increment\_v4:

package Task1;  
  
import java.util.Scanner;  
  
*/\* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class Increment {  
 public static void main(String[] args){  
 Scanner input = new Scanner(System.*in*);  
 int n = input.nextInt();  
 System.*out*.println("Before the call, x is "+ n);  
 *increment*(n); *// вызов метода increment* System.*out*.println("After the call, x is " + n);  
 }  
  
 public static void increment(int n) { *//объявление метода increment* ++n;  
 System.*out*.println("n inside the method is " + n);  
 }  
}

**Скриншот выполнения программы**



Рисунок 1 – результат работы программыIncrement



Рисунок 2 – результат работы программыIncrement\_v2



Рисунок 3 – результат работы программыIncrement\_v3

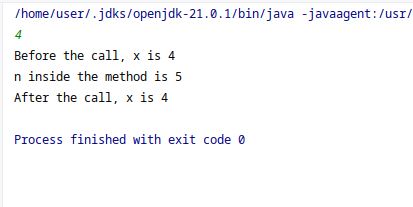


Рисунок 4 – результат работы программыIncrement\_v4

**Контрольные вопросы:**

В чем отличие передачи данных по ссылке и по значению?

Ссылка в Java – это нечто похожее на адрес объекта в памяти. Похожая сущность в C++ называется указатель. Передача параметра по значению – это копирование значения в переменную-параметр метода. По ссылке – передача ссылки, то есть по сути использование той же самой переменной.

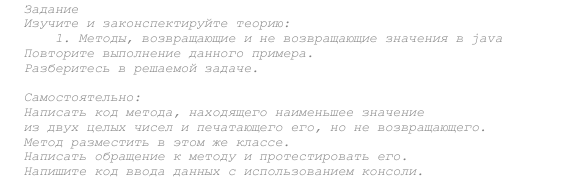
**Глоссарий**

**MaxTest**

**Цель работы**

Изучить работу методов, возвращающих и не возвращающих значений

**Задания:**

дан

**Код программы**:

package Task2;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class MaxTest {  
 */\*\* Main method \*/* public static void main(String[] args) {  
 Scanner inputNum1 = new Scanner(System.*in*);  
 int num1 = inputNum1.nextInt();  
 Scanner inputNum2 = new Scanner(System.*in*);  
 int num2 = inputNum2.nextInt();  
  
 *//call method min  
 min*(num1, num2);  
 }  
  
 */\*\* Output the min between two numbers in console \*/* public static void min(int num1, int num2){  
 int min = (num1 < num2) ? num1 : num2;  
 System.*out*.println("Наименьшее из двух чисел: " + min);  
 }  
}

**Скриншот выполнения программы**

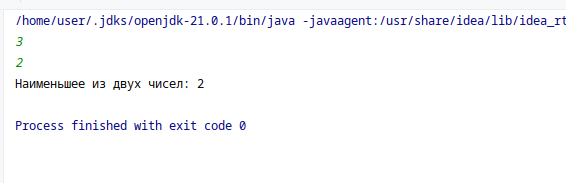


Рисунок 5 – результат работы программы MaxTest

**Контрольные вопросы**:

1. Какой оператор используется для возвращения значения из метода и его синтаксис?

Оператор **return**.

1. В чем отличия объявления методов, возращающих и не возвращающих значения?

Для методов не возращающих значения используется ключевое слово void, а для методов возвращающих используется ключевое слово, несущее в себе информацию о типе данных, который будет возвращать метод

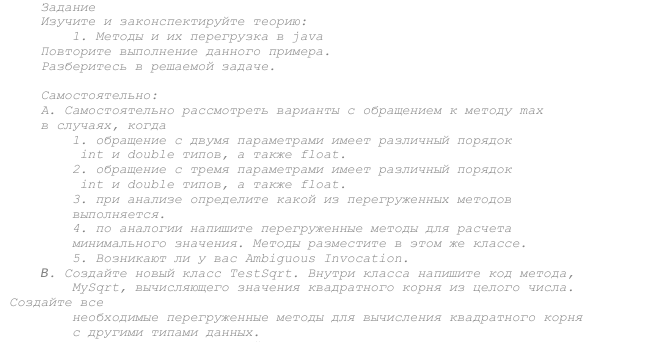
**Глоссарий**

**TestMethodOverloading**

**Цель работы**

Изучить методы и их перегрузки в java

**Задания:**



**Код программы**:

Код программы TestMethodOverloading;

package Task3;  
*/\*\*  
 \* перегрузка методов  
 \* имена методов одинаковые, различные сигнатуры  
 \* количество параметров и их типы отличаются  
 \* на примере нахождения максимума и минимума  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class TestMethodOverloading {  
 */\*\*  
 \* Main method  
 \*/* public static void main(String[] args) {  
 *// Вызов метода max метода с двумя int параметрами* System.*out*.println("The maximum between 3 and 4 is "  
 + *max*(3, 4));  
 *// Вызов метода max с двумя double параметрами* System.*out*.println("The maximum between 3.0 and 5.4 is " + + *max*(3.0, 5.4));  
 *// Вызов метода max с int и double параметрами* System.*out*.println("The maximum between 3 and 5.4 is " + *max*(3, 5.4));  
 *// Вызов метода max с double и int параметрами* System.*out*.println("The maximum between 3.5 and 5 is " + *max*(3.5, 5));  
 *// Вызов метода max с double и float параметрами* System.*out*.println("The maximum between 3.5 and 5.23f is " + *max*(3.5, 5.23f));  
 *// Вызов метода max с тремя! double параметрами* System.*out*.println("The maximum between 3.0, 5.4, and 10.14 is " + *max*(3.0, 5.4, 10.14));  
 *// Вызов метода max с float, int, double параметрами* System.*out*.println("The maximum between 7.05f, 5, and 1.14 is " + *max*(7.05f, 5, 1.14));  
  
 *// Вызов метода min с двумя double параметрами* System.*out*.println("The minimum between 3.0 and 5.4 is " + + *min*(3.0, 5.4));  
 *// Вызов метода min с int и double параметрами* System.*out*.println("The minimum between 3 and 5.4 is " + *min*(3, 5.4));  
 *// Вызов метода min с double и int параметрами* System.*out*.println("The minimum between 3.5 and 5 is " + *min*(3.5, 5));  
 *// Вызов метода min с double и float параметрами* System.*out*.println("The minimum between 3.5 and 5.23f is " + *min*(3.5, 5.23f));  
 *// Вызов метода min с тремя! double параметрами* System.*out*.println("The minimum between 3.0, 5.4, and 10.14 is " + *min*(3.0, 5.4, 10.14));  
 *// Вызов метода min с float, int, double параметрами* System.*out*.println("The minimum between 7.05f, 5, and 1.14 is " + *min*(7.05f, 5, 1.14));  
  
 }  
 */\*\*  
 \* Возвращает max из двух int значений  
 \*/* public static int max(int num1, int num2) {  
 if (num1 > num2)  
 return num1;  
 else  
 return num2;  
 }  
 */\*\*  
 \* Возвращает max из двух double значений  
 \*/* public static double max(double num1, double num2) {  
 if (num1 > num2)  
 return num1;  
 else  
 return num2;  
 }  
 */\*\*  
 \* Возвращает max из трех double значений  
 \* используется обращение к методу с двумя параметрами  
 \*/* public static double max(double num1, double num2, double num3) { return *max*(*max*(num1, num2), num3);  
 }  
 */\*\*  
 \* Возвращает min из двух int значений  
 \*/* public static int min(int num1, int num2) {  
 if (num1 < num2)  
 return num1;  
 else  
 return num2;  
 }  
 */\*\*  
 \* Возвращает min из двух double значений  
 \*/* public static double min(double num1, double num2) {  
 if (num1 < num2)  
 return num1;  
 else  
 return num2;  
 }  
 */\*\*  
 \* Возвращает min из трех double значений  
 \* используется обращение к методу с двумя параметрами  
 \*/* public static double min(double num1, double num2, double num3) { return *min*(*min*(num1, num2), num3);  
 }  
}

Код программы TestSqrt;

package Task3;  
  
public class TestSqrt {  
 */\*\*  
 \* Main method  
 \*/* public static void main(String[] args) {  
 *// Вызов метода mySqrt метода с двумя int параметрами* System.*out*.println("The sqrt 3 and 4 is " + *mySqrt*(3, 4));  
 *// Вызов метода mySqrt с двумя double параметрами* System.*out*.println("The sqrt 3.0 and 5.4 is " + *mySqrt*(3.0, 5.4));  
 *// Вызов метода mySqrt с int и double параметрами* System.*out*.println("The sqrt 3 and 5.4 is " + *mySqrt*(3, 5.4));  
 *// Вызов метода mySqrt с double и int параметрами* System.*out*.println("The sqrt 3.5 and 5 is " + *mySqrt*(3.5, 5));  
 *// Вызов метода mySqrt с double и float параметрами* System.*out*.println("The sqrt 3.5 and 5.23f is " + *mySqrt*(3.5, 5.23f));  
  
 }  
 */\*\*  
 \* Возвращает mySqrt из двух int значений  
 \*/* public static int mySqrt(int num1, int num2) {  
 return num2\*num1;  
 }  
 */\*\*  
 \* Возвращает mySqrt из двух double значений  
 \*/* public static double mySqrt(double num1, double num2) {  
 return num2\*num1;  
 }  
}

**Скриншот выполнения программы**

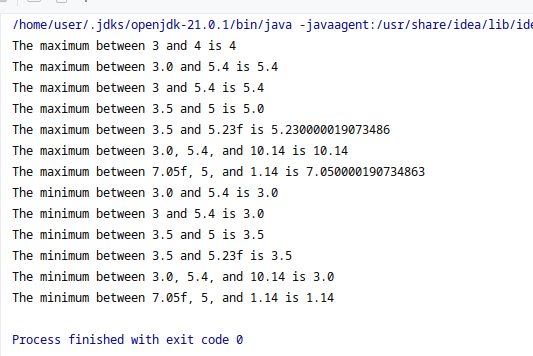


Рисунок 6 – результат работы программы TestMethodOverloading

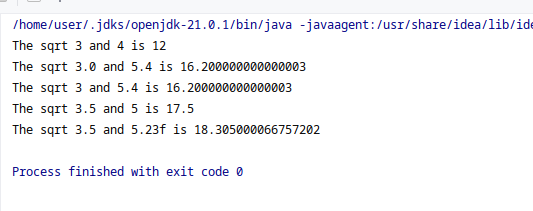


Рисунок 7 – результат работы программы TestSqrt

**Контрольные вопросы:**

1. Что такое перегрузка методов?

**Перегрузка методов** (method overloading) в Java — это возможность создавать несколько методов с одинаковым именем, но с разными параметрами (типами и/или количеством). Это позволяет использовать один и тот же метод для различных типов данных.

1. Что такое сигнатура методов?

**Сигнатура метода** (method signature) — это уникальная комбинация имени метода и его параметров (типы, порядок и количество). Сигнатура используется компилятором для определения, какой именно метод нужно вызвать.

.

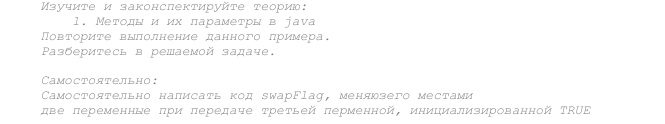
**Глоссарий**

**TestPassByValue**

**Цель работы**

Изучить методы и их параметры в java

**Задания:**



**Код программы:**

package Task4;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class TestPassByValue {  
 */\*\* Main method \*/* public static void main(String[] args) {  
 *// Объявляем и инициализируем переменные* int num1 = 1;  
 int num2 = 2;  
 boolean flag;  
 Scanner inputFlag = new Scanner(System.*in*);  
 int intFlag = inputFlag.nextInt();  
 if (intFlag == 0)  
 flag = false;  
 else  
 flag = true;  
 System.*out*.println("Перед вызовом метода swap, num1 = " + num1 + " и num2 = " + num2);  
 int[] output = *swap*(num1, num2, flag);  
 num1 = output[0];  
 num2 = output[1];  
 System.*out*.println("После вызова метода swap method, num1 = " + num1 + " и num2 = " + num2);  
 }  
 */\*\* Swap two variables \*/* public static int[] swap(int n1, int n2, boolean flag) {  
 if (!flag)  
 System.*out*.println("Флаг false, операция не выполнена");  
 else {  
 System.*out*.println("\tВнутри метода swap");  
 System.*out*.println("\t\tПеред перестановкой n1 = " + n1 + " n2 = " + n2);  
 *// Swap n1 with n2* int temp = n1;  
 n1 = n2;  
 n2 = temp;  
 System.*out*.println("\t\tПосле перестановки n1 = " + n1 + " n2 = " + n2);  
 }  
 int[] output = {n1, n2};  
 return output;  
 }  
}

**Скриншот выполнения программы**

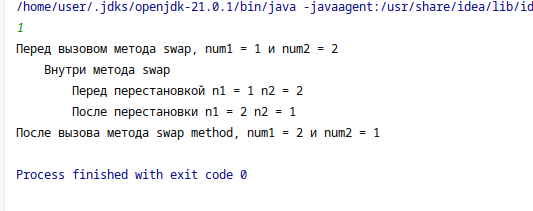


Рисунок 8 – результат работы программы TestPassByValue

**Контрольные вопросы:**

1. Как передать данные в метод?

В Java данные передаются в метод через параметры. Параметры указываются в круглых скобках после имени метода.

1. Может ли метод возвращать несколько значений?

В Java метод может возвращать только одно значение. Однако есть несколько способов обойти это ограничение:

**Использование объекта**: Создайте класс, который будет содержать все значения, которые вы хотите вернуть.

**Использование массива или коллекции**: Вы можете вернуть массив или коллекцию, содержащую несколько значений.

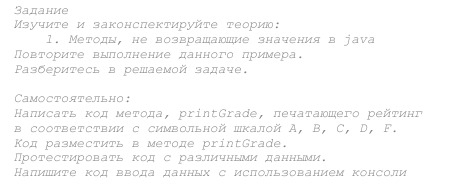
**Глоссарий**

**VoidMethodTest**

**Цель работы**

Изучить методы, не возвращающие значения

**Задания:**



**Код программы:**

package Task5;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class VoidMethodTest {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Enter grade score: ");  
 double score = input.nextDouble();  
 System.*out*.print("The grade is ");  
 *printGrade*(score);  
  
 }  
 public static void printGrade(double score) {  
 if (score >= 90) {  
 System.*out*.print("A");  
 } else if (score >= 80) {  
 System.*out*.print("B");  
 } else if (score >= 70) {  
 System.*out*.print("C");  
 } else if (score >= 60) {  
 System.*out*.print("D");  
 } else if (score >= 50) {  
 System.*out*.print("E");  
 } else {  
 System.*out*.print("F");  
 }  
 }  
}

**Скриншот выполнения программы**

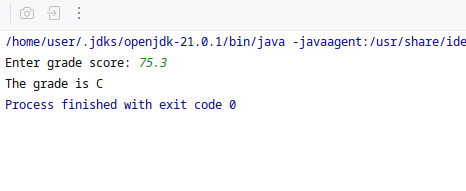


Рисунок 9 – результат работы программы VoidMethodTest

**Контрольные вопросы:**

1. Какое ключевое слово в объявлении отличает методы,   возвращающие и не возвращающие значения?

Методы, которые **не возвращают значения**, объявляются с ключевым словом void. Это означает, что метод не возвращает никакого значения.

Методы, которые **возвращают значение**, объявляются с указанием типа возвращаемого значения (например, int, double, String и т.д.). В этом случае метод должен содержать оператор return, который возвращает значение указанного типа

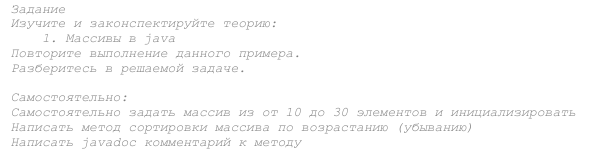
**Глоссарий**

**Array**

**Цель работы**

Изучить массивы в java

**Задания:**



**Код программы:**

package Task6;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class Array {  
  
 private int[] array;  
  
 */\*\*  
 \* Конструктор для создания и инициализации массива.  
 \* @param size размер массива (от 10 до 30 элементов)  
 \*/* public Array(int size) {  
 if (size < 10 || size > 30) {  
 throw new IllegalArgumentException("Размер массива должен быть от 10 до 30 элементов.");  
 }  
 array = new int[size];  
 for (int i = 0; i < size; i++) {  
 array[i] = (int) (Math.*random*() \* 100); *// Инициализация случайными значениями от 0 до 99* }  
 }  
  
 */\*\*  
 \* Метод для сортировки массива по возрастанию.  
 \*/* public void sortAscending() {  
 java.util.Arrays.*sort*(array);  
 }  
  
 */\*\*  
 \* Метод для сортировки массива по убыванию.  
 \*/* public void sortDescending() {  
 java.util.Arrays.*sort*(array);  
 reverseArray();  
 }  
  
 */\*\*  
 \* Метод для вывода массива.  
 \*/* public void printArray() {  
 for (int num : array) {  
 System.*out*.print(num + " ");  
 }  
 System.*out*.println();  
 }  
  
 */\*\*  
 \* Вспомогательный метод для разворота массива.  
 \*/* private void reverseArray() {  
 int n = array.length;  
 for (int i = 0; i < n / 2; i++) {  
 int temp = array[i];  
 array[i] = array[n - 1 - i];  
 array[n - 1 - i] = temp;  
 }  
 }  
  
 public static void main(String[] args) {  
  
 Scanner input = new Scanner(System.*in*);  
 int size = input.nextInt();  
 Array arr = new Array(size);  
 System.*out*.println("Исходный массив:");  
 arr.printArray();  
  
 arr.sortAscending();  
 System.*out*.println("Отсортированный по возрастанию массив:");  
 arr.printArray();  
  
 arr.sortDescending();  
 System.*out*.println("Отсортированный по убыванию массив:");  
 arr.printArray();  
 }  
}

**Скриншот работы программы**

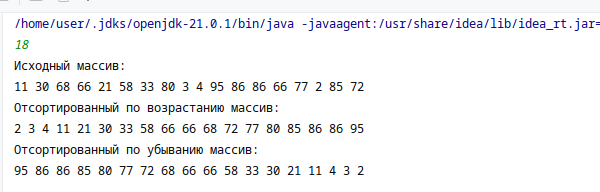


Рисунок 9 – результат работы программы ContinueTest

**Контрольные вопросы:**

Как объявляются массивы?

Какие существуют варианты объявления массивов?

В Java массивы можно объявлять и инициализировать несколькими способами:

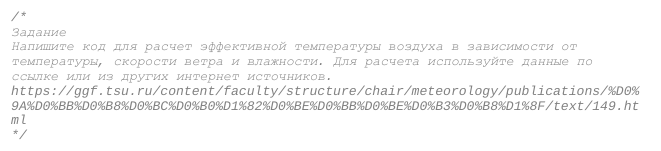
**Объявление массива:**

* 1. Указание типа данных и имени массива. Например, int[] nums; или int nums[];.

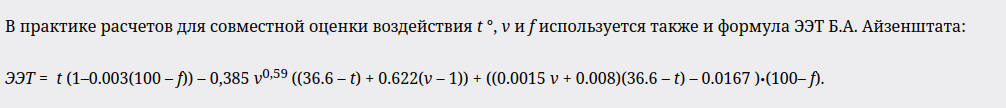
**Инициализация массива:**

* 1. **С указанием размера:** Создание массива с определенным количеством элементов, например, new int[5];.
  2. **С конкретными значениями:** Создание массива и одновременное присвоение значений, например, {1, 2, 3, 4, 5};.
  3. **С использованием ключевого слова** new **и конкретных значений:** Например, new int[] {1, 2, 3, 4, 5};.

**WindChillTemperature**

**Заданиe:**

**Математическая модель:**



**Код программы**:

package Task7;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class WindChillTemperature {  
  
 public static void main(String[] args) {  
  
 Scanner inputTemperature = new Scanner(System.*in*);  
 System.*out*.print("Температура в градусах Цельсия: ");  
 double temperature = inputTemperature.nextDouble(); *// Температура в градусах Цельсия* Scanner inputWindSpeed = new Scanner(System.*in*);  
 System.*out*.print("Скорость ветра в м/с: ");  
 double windSpeed = inputWindSpeed.nextDouble(); *// Скорость ветра в м/с* Scanner inputHumidity = new Scanner(System.*in*);  
 System.*out*.print("Влажность в процентах: ");  
 double humidity = inputHumidity.nextDouble(); *// Влажность в процентах* double effectiveTemperature = *calculateEffectiveTemperature*(temperature, windSpeed, humidity);  
 System.*out*.println("Эффективная температура согласно формуле Б.А.Айзенштата: " + effectiveTemperature + " °C");  
 }  
  
 public static double calculateEffectiveTemperature(double temperature, double windSpeed, double humidity) {  
 *// Пример формулы для расчета эффективной температуры* double EET = temperature \* (1-0.003\*(100-humidity))-0.385\*Math.*pow*(windSpeed, 0.59)\*((36.6-temperature)+0.622\*(windSpeed-1))+((0.0015\*windSpeed+0.008)\*(36.6-temperature)-0.0167)\*(100-humidity);  
 return EET;  
 }  
}

**Скриншот выполнения программы**

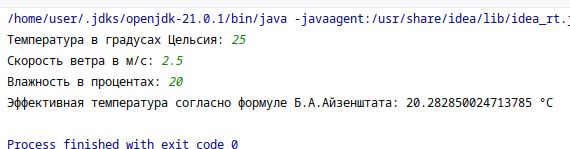


Рисунок 10 – результат работы программы WindChillTemperature