Topic Name- Variables Method Operator Control Statement Looping Statement Jumping statement

Q1. Difference between Parameters & Arguments?

|  |  |
| --- | --- |
| Parameter | Argument |
| The parameter is the variable inside the definition of the function. | An argument is an instance that appears in function calls. |
| A parameter is also called a *formal parameter* or *formal argument* | An argument is often called *actual arguments* or *actual parameters*. |
| A parameter has a name, a data type, and a calling mechanism (call by reference or call by value). | An argument is an expression that does not have any name, but it can be a variable, a constant, or a literal. |
| In parameters, to get value from arguments we utilize the local variable in function declaration. | Arguments are used to transfer the value from calling function to the receiving function |

Q2. Is java call by value or call by reference?

* There is only call by value in java, not call by reference. If we call a method passing a value, it is known as call by value.
* Call by Value means calling a method with a parameter as value. Through this, the argument value is passed to the parameter.
* While Call by Reference means calling a method with a parameter as a reference. Through this, the argument reference is passed to the parameter.

Q3. What is recursion ?

* Recursion in java is a process in which a method calls itself continuously. A method in java that calls itself is called recursive method.
* It makes the code compact but complex to understand.

Syntax:-

returntype methodname(){

//code to be executed

methodname();//calling same method

}

Example :-

**public** **class** RecursionExample1 {

**static** **void** p(){

System.out.println("hello");

p();

}

**public** **static** **void** main(String[] args) {

p();

}

}

Output:-

hello

hello

…….

Java.lang.StackOverflowError

Q4. What is difference between & and && ?

|  |  |
| --- | --- |
| **& Operator** | **&& Operator** |
| It is a bitwise AND operator. | It is a logical AND operator. |
| It evaluates both the left and right side of the given expression. | It only evaluates the left sides of the given expression |
| It operates on Boolean data types as well as on bits. | It operates only on Boolean dataty |
| Used to check logical condition and also used to mask off certain bits such as parity bits. | Used only to check the logical conditions. |
| Example- z = x & y | Example - if (y > 1 && y > x) |

Q5. Default value of local variables & global variables ?

* There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.
* global and static variables have '0' as their default values.

Q6. What is variables? Types of variable?

* A variable is a container which holds the value while the [Java program](https://www.javatpoint.com/simple-program-of-java) is executed.
* A variable is assigned with a data type.
* Variable is a name of memory location.
* There are three types of variables in java: local, instance and static.

1) Local Variable🡪

* A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
* A local variable cannot be defined with "static" keyword.

#### 2) Instance Variable

* A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as [static](https://www.javatpoint.com/static-keyword-in-java).
* It is called an instance variable because its value is instance-specific and is not shared among instances.

#### 3) Static variable

* + - A variable that is declared as static is called a static variable. It cannot be local.
    - You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Q7. Ternary operator ?

* In Java, the **ternary operator** is a type of Java conditional operator.
* The meaning of **ternary** is composed of three parts. The **ternary operator (? :)** consists of three operands. It is used to evaluate Boolean expressions.
* The operator decides which value will be assigned to the variable.
* It is the only conditional operator that accepts three operands.
* It can be used instead of the if-else statement. It makes the code much more easy, readable, and shorter.

Syntax:

variable = (condition) ? expression1 : expression2

The above statement states that if the condition returns **true, expression1** gets executed, else the **expression2** gets executed and the final result stored in a variable.

Q8. Can we declare main() Method As Private Or Protected Or With No Access Modifier?

* Answer is Yes.
* But technically, the class will compile successfully but will get run time error as no main method found.

Q8. Will program run if I declare main method as following?

**public** **static** **void** main(String[] args) -----YES

**static** **public** **void** main(String[] args)-----YES

main **static** **public** **void** (String[] args)-----NO

(String[] args) **public** **static** **void** main -----NO

**void** main **static** **public** (String[] args) -----NO

**void** **public** **static** main (String[] args) -----NO

Q8. What is Java String compareTo() method?

* The Java String class compareTo() method compares the given string with the current string lexicographically.
* It returns a positive number, negative number, or 0.
* It compares strings on the basis of the Unicode value of each character in the strings.
* This method is case sensitive.
* **if** s1 > s2, it returns positive number
* **if** s1 < s2, it returns negative number
* **if** s1 == s2, it returns 0

**public** **class** CompareToExample{

**public** **static** **void** main(String args[]){

String s1="hello";

String s2="hello";

String s3="meklo";

String s4="hemlo";

String s5="flag";

System.***out***.println(s1.compareTo(s2));

//0 because both are equal

System.***out***.println(s1.compareTo(s3));

//-5 because "h" is 5 times lower than "m"

System.***out***.println(s1.compareTo(s4));

//-1 because "l" is 1 times lower than "m"

System.***out***.println(s1.compareTo(s5));

//2 because "h" is 2 times greater than "f"

}

}

When we compare two strings in which either first or second string is empty, the method returns the length of the string. So, there may be two scenarios:

* If **first** string is an empty string, the method returns a **negative**
* If **second** string is an empty string, the method returns a **positive** number that is the length of the first string.