Operators and Loops

Assignment Questions

Assignment Question

1. **What are the Conditional Operators in Java ?**

   Ans :

- **There is only one conditional operator in Java, which is the ternary operator**
- **The ternary operator takes three operands and returns a value based on a condition, the syntax of the ternary operator is as follows :**

condition ? expression 1 : expression 2 ;

If the condition is true then the expression 1 is executed, if the condition is false then the expression 2 is executed.

- **Ex :**

```
class TernaryOperator
{
  public static void main(String[] args)
    {
        int a=30;
        int b=40;

        String result = (a<b)? “a is smaller number” : “b is smaller number” ;
        System.out.println (result);
    }
}
```

In this program the output is the ‘a is smaller number’ because the condition is true.

2. **What are the types of operators based on the number of operands ?**

   Ans :

a. Unary Operators :

These operators work with a single operand.

Unary Plus (+) : Represents positive values (e.g., +5).
Unary Minus (-) : Negates the value (e.g., -5).
Increment (++) and Decrement (--) : Increase or decrease the value by 1.
Logical Complement (!) : Flips the logical state (true becomes false and vice versa).
Bitwise Complement (~) : Inverts the bits of an integer.
Binary Operators : These operators work with two operands.

b. Arithmetic Operators: Perform arithmetic operations.

Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo/Remainder (%)

c. Relational Operators: Compare two values and return a boolean result.

Equal to (==)
Not equal to (!=)
Greater than (>)
Less than (<)
Greater than or equal to (>=)
Less than or equal to (<=)

d. Logical Operators: Perform logical operations on boolean values.

Logical AND (&&)
Logical OR (||)

e. Assignment Operators: Assign a value to a variable.

Simple assignment (=)
Compound assignment operators (e.g., +=, -=, *=)

3. What is the use of Switch case in Java programming ?

And :

- **The switch statement in Java is a control flow statement that allows you to select one of many code blocks to be executed based on the value of a variable or an expression.**
- **It provides an alternative to using multiple if-else statements when you have a limited number of possible values to compare**
- **The switch statement is useful when you have a limited set of values to compare against a single variable or expression. It can make your code more readable and concise compared to using multiple if-else statements**

Ex :

```
class SwitchCase
{
  public static void main(String[] args)
   {
      Int num = 2;
      switch(num)
     {
     case 1 : System.out.println(“case 1”);
     break;
     case 2 : System.out.println(“case 2”);
     break;
     case 3 : System.out.println(“case 3”);
     break;
     default : System.out.println(“default”);
     break;
     }
  }
}
```

The output of these program is ‘case 2’

4. What are the priority levels of arithmetic operation in Java ?

Ans :

- **In Java, the following are the priority levels of arithmetic operations, from highest to lowest:**

  Parentheses () and Array subscript []
  Unary operators -, +, ~, !
  Multiplication, division, and modulus *, /, %
  Addition and subtraction +, -
  Bitwise operators &, |, ^, >>, <<, >>>
  Relational operators <, <=, >, >=, ==, !=
  Logical operators &&, ||
  Ternary operator ? :

- **In other words, operators with higher precedence are evaluated before operators with lower precedence. For example, in the expression 5 * 3 + 2, the multiplication is performed first, followed by the addition. The result is 17.**

- **If there are operators with the same precedence, they are evaluated left-to-right.**

5. What are the conditional Statements and use of conditional statements in Java ?

Ans :

Conditional statements in Java are used to control the flow of execution based on certain conditions. They allow the program to make decisions and execute different blocks of code depending on whether a condition is true or false. There are three main types of

conditional statements in Java :

a. **if statement :**

The if statement is the most basic conditional statement. It executes a block of code if a specified condition is true. If the condition is false, the block is skipped

```
if (condition)
{
   // executed when the condition is true
}
```

b. **if-else statement :**

The if-else statement provides an alternative block of code to be executed if the condition in the if statement is false .

```
if (condition)
{
   // executed when the condition is true
}

else
{
   // executed when condition is false
}
```

c. **if-else-if ladder :**

The if-else-if ladder allows you to test multiple conditions and execute different code blocks based on which condition is true. It provides a series of if-else statements, where each else-if statement is evaluated only if the preceding conditions are false

```
if (condition1)
{
   // executed if condition 1 is true
}
else if (condition2)
{
   // executed if condition 2 is true
}
else if (condition3)
{
   // executed if condition 3 is true
}
else
{
   // executed if all conditions are false
}
```

6. What is the syntax of if else statement ?

Ans :

- **The syntax of if else statement is as follows :**

```
if (condition)
{
  //executed if the condition is true
}
else
{
  //executed if the condition is false
}
```

- **The if-else statement allows the program to execute different blocks of code based on the result of the condition. If the condition is true, the code block following the "if" is executed, and if the condition is false, the code block following the "else" is executed.**

7. What are the 3 types of iterative statements in java ?

Ans :

- **In Java, there are three types of iterative statements, also known as loops, which are used to execute a block of code repeatedly :**

- **for loop :**

  The for loop is used when you know the number of iterations in advance. It consists of an initialization, a condition, an update, and a loop body.

```
for (initialization; condition; update)
{
    // code block to be executed
}
```

- **while loop :**

  The while loop is used when the number of iterations is not known in advance, and the loop continues as long as a specified condition is true.

```
while (condition)
{
    // code block to be executed
}
```

- **do-while loop:**

  The do-while loop is similar to the while loop, but the condition is evaluated after executing the loop body. This guarantees that the loop body is executed at least once.

```
do
{
   // code block to be executed
} while (condition);
```

- **All three types of loops allow you to repeat a block of code until a certain condition is met**

8. Write the difference between for loop and do-while loop ?

Ans :

- **Initialization and Increment/Update :**

  In a for loop, you can specify the initialization of the loop variable(s) and the increment/update statement within the loop syntax itself. These statements are executed before each iteration of the loop.

  In a do-while loop, the initialization and increment/update statements need to be placed outside the loop. They are executed within the loop body after each iteration.

- **Condition Evaluation :**

  In a for loop, the condition is evaluated before each iteration. If the condition is false, the loop is not executed at all.

  In a do-while loop, the condition is evaluated after each iteration. This means that the loop body is executed at least once, regardless of the condition's initial state.

- **Loop Control :**

  In a for loop, you have more control over the loop because you explicitly define the initialization, condition, and update statements within the loop syntax. This makes it easier to manage the loop variables and control the loop flow.

  In a do-while loop, the loop control is more limited because the condition is checked after each iteration. However, the do-while loop is useful when you want to ensure that the loop body is executed at least once, regardless of the condition.

9. Write a program to print numbers from 1 to 10.

Ans :

```
class Count
{
public static void main(String[] args)
  {
     for (int i=0; i<10; i++)
     {
       System.out.println(i+1);
    }
  }
}
```