ASSIGNMENT-2(JAVA CORE MODULE)

Q. What are the Conditional Operators in Java?

Ans- In Java, conditional operators are used to evaluate Boolean expressions and make decisions based on the result. They allow you to create conditional statements and control the flow of your program. The conditional operators in Java are:

1. Equality Operators:

   - == (equal to): Compares if two operands are equal in value.

   - != (not equal to): Compares if two operands are not equal in value.

   These operators are commonly used to compare primitive data types and object references for equality.

2. Relational Operators:

   - > (greater than): Checks if the left operand is greater than the right operand.

   - < (less than): Checks if the left operand is less than the right operand.

   - >= (greater than or equal to): Checks if the left operand is greater than or equal to the right operand.

   - <= (less than or equal to): Checks if the left operand is less than or equal to the right operand.

   These operators are primarily used for numerical comparisons but can also be used with other compatible types.

3. Logical Operators:

   - && (logical AND): Evaluates two Boolean expressions and returns true if both expressions are true.

   - || (logical OR): Evaluates two Boolean expressions and returns true if at least one expression is true.

   - ! (logical NOT): Reverses the logical state of a Boolean expression.

   Logical operators are often used to combine multiple conditions and create complex Boolean expressions.

4. Conditional Operator (Ternary Operator):

   - ?: (conditional operator): Also known as the ternary operator, it is the only ternary operator in Java.

   - It is used to evaluate a condition and return one of two expressions based on the result of the condition.

- Syntax: condition ? expression1 : expression2

- If the condition is true, expression1 is evaluated and returned; otherwise, expression2 is evaluated and returned.

The conditional operator is a concise way to write simple conditional statements.

These operators are used in conditional statements, such as if-else statements, loops, and other control flow structures, to make decisions and control program execution based on certain conditions.

Q2. What are the types of operators based on the number of operands?

Ans- Operators in programming languages can be classified based on the number of operands they work with. The three main categories of operators based on the number of operands are:

1. Unary Operators:

- Unary operators work with a single operand.
- They perform operations on a single value or variable.
- Examples of unary operators in Java include:
  - Unary plus (+): Represents the positive value of a number.
  - Unary minus (-): Negates a numeric value.
  - Increment (++): Increases the value of a variable by 1.
  - Decrement (--): Decreases the value of a variable by 1.
  - Logical NOT (!): Reverses the logical state of a Boolean value.

2. Binary Operators:

- Binary operators work with two operands.
- They perform operations that involve two values or variables.
- Examples of binary operators in Java include:
  - Arithmetic operators: Addition (+), subtraction (-), multiplication (*), division (/), modulus (%).
  - Relational operators: Greater than (>), less than (<), greater than or equal to (>=), less than or equal to (<=), equal to (==), not equal to (!=).
  - Logical operators: Logical AND (&&), logical OR (||).
  - Assignment operator (=): Assigns a value to a variable.

3. Ternary Operator:

- The ternary operator is the only operator in Java that takes three operands.
- It is a conditional operator that evaluates a condition and returns one of two expressions based on the result.
- The syntax of the ternary operator is: condition ? expression1 : expression2.
- If the condition is true, expression1 is evaluated and returned; otherwise, expression2 is evaluated and returned.

These different types of operators provide the means to perform a wide range of operations in programming, from basic arithmetic calculations to complex logical evaluations and conditional assignments.

Q3. What is the use of Switch Case in Java Programming?

Ans- The switch-case statement in Java is a control flow statement that provides an alternative way to write multiple if-else-if statements when dealing with multiple possible conditions. It allows the program to evaluate an expression and execute different blocks of code based on the value of that expression.

```
switch (expression) {
   case value1:
      // Code to execute if expression equals value1
      break;
   case value2:
      // Code to execute if expression equals value2
      break;
   // Additional cases...
   default:
      // Code to execute if expression doesn't match any case
}
```

Q4. What are the conditionals Statements and use of conditional statements in java?

Ans- Conditional statements in Java allow the program to make decisions and control the flow of execution based on certain conditions. They enable the program to choose between different paths or perform different actions based on whether a condition is true or false.

The use of conditional statements in Java allows for selective execution of code blocks based on specific conditions or values. They enable programs to handle different scenarios, make decisions, validate inputs, implement branching logic, and control the program's flow and behavior. Conditional statements play a crucial role in implementing logic and ensuring the desired behavior of Java programs.

Q5. What is the syntax of if-else statement?

Ans- 1. if Statement:

- The if statement is the most basic conditional statement in Java.
- It evaluates a Boolean expression and executes a block of code if the expression is true.
- Syntax:

```java
if (condition) {
    // code to execute if the condition is true
}
```

2. if-else Statement:

- The if-else statement extends the if statement by providing an alternative block of code to execute if the condition is false.
- It allows the program to choose between two different paths based on the Boolean expression.
- Syntax:

```java
if (condition) {
    // code to execute if the condition is true
} else {
    // code to execute if the condition is false
}
```

3. if-else-if Ladder:

- The if-else-if ladder is used when multiple conditions need to be evaluated in a sequence.
- It allows the program to choose from multiple options based on the first condition that evaluates to true.
- Syntax:

```java
if (condition1) {
    // code to execute if condition1 is true
```

```java
   } else if (condition2) {
       // code to execute if condition2 is true
   } else if (condition3) {
       // code to execute if condition3 is true
   } else {
       // code to execute if all conditions are false
   }
```

4. switch-case Statement:

  - The switch-case statement provides a concise way to handle multiple conditions with multiple possible values.

  - It evaluates an expression and executes different blocks of code based on the matching case.

  - Syntax:

```java
   switch (expression) {
       case value1:
           // code to execute if expression equals value1
           break;
       case value2:
           // code to execute if expression equals value2
           break;
       // Additional cases...
       default:
           // code to execute if expression doesn't match any case
   }
```

Q6. How Do you Compare two Strings in Java?

Ans- In Java, you can compare two strings using various methods and operators. Here are some common approaches to compare strings in Java:

1. Using the equals() Method:

- The `equals()` method is used to compare the contents of two strings for equality.
- It returns `true` if the strings have the same sequence of characters, and `false` otherwise.
- Syntax:

```java
String str1 = "Hello";
String str2 = "World";
if (str1.equals(str2)) {
    // Strings are equal
} else {
    // Strings are not equal
}
```

2. Ignoring Case Sensitivity:

- If you want to compare strings while ignoring case sensitivity, you can use the `equalsIgnoreCase()` method instead of `equals()`.
- The `equalsIgnoreCase()` method compares the contents of two strings while disregarding the case of the characters.
- Syntax:

```java
String str1 = "Hello";
String str2 = "hello";
if (str1.equalsIgnoreCase(str2)) {
    // Strings are equal (ignoring case)
} else {
    // Strings are not equal (ignoring case)
}
```

3. Using the compareTo() Method:

- The `compareTo()` method is used to perform lexicographic comparison of two strings.
- It returns an integer value indicating the relative order of the strings.

- If `str1` is less than `str2`, a negative value is returned. If `str1` is greater, a positive value is returned. If they are equal, `0` is returned.

- Syntax:

```java
String str1 = "Apple";
String str2 = "Banana";
int comparisonResult = str1.compareTo(str2);
if (comparisonResult < 0) {
    // str1 is less than str2
} else if (comparisonResult > 0) {
    // str1 is greater than str2
} else {
    // str1 and str2 are equal
}
```

4. Using the == Operator:

- The `==` operator is used to compare the references of two string objects.
- It checks if the two string references point to the same memory location.
- However, it does not compare the actual content of the strings.
- Syntax:

```java
String str1 = "Hello";
String str2 = "Hello";
if (str1 == str2) {
    // Strings are equal
} else {
    // Strings are not equal
}
```

Q7. What is Mutable Strings in Java Explain with an example?

Ans- Once if we create a string, on that string if we try to perform any operation and if those changes get reflected in the same object then such strings are called mutable strings.

Example- StringBuffer, StringBuilder.

Q8. Write a program to sort a string alphabetically.

Ans-

```
public class sortString {
   public static void main(String[] args) {
      String str = "amit";
      char[] charArray = str.toCharArray();
      int n = charArray.length;
      for (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < n - i - 1; j++) {
            if (charArray[j] > charArray[j + 1]) {
               // Swap characters if they are in the wrong order
               char temp = charArray[j];
               charArray[j] = charArray[j + 1];
               charArray[j + 1] = temp;
            }
         }
      }
      String sortedStr = new String(charArray);
      System.out.println("Original String: " + str);
      System.out.println("Sorted String: " + sortedStr);
   }
}
```

Q9. Write a program to check if the letter ‘e’ is present in the word ‘Umbrella’?

Ans- // Online Java Compiler

```
// Use this editor to write, compile and run your Java code online


class HelloWorld {
    public static void main(String[] args) {
        String s= "Umbrella";
        char[] ch= s.toCharArray();
        for(int i=0; i<ch.length; i++)
        {
            if(ch[i]=='e')
            {
                System.out.print("e is present in Umbrella");
                break;
            }
        }
    }
}
```

Q 10. Where exactly is the string constant pool located in the memory?

Ans- In Java, the string constant pool is a special area of memory that stores literal string objects. It is part of the runtime constant pool, which is a section of memory associated with each class and loaded by the JVM when a class is loaded. The string constant pool is typically located in the non-heap memory area called the "Method Area" or "PermGen" (Permanent Generation) in older versions of Java.
