2

Q1. What are the Conditional Operators in Java?

AND = &&
OR = ||
Ternary ?:

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

Based on the number of operands, operators in Java can be categorized into three types:

1. Unary Operators: Unary operators act on a single operand. They can be used to perform operations such as negation, increment, decrement, and logical complement. Some examples of unary operators are:
   - Unary minus (-): Changes the sign of a numeric operand.
   - Unary plus (+): Indicates the sign of a numeric operand.
   - Increment (++): Increases the value of a variable by 1.
   - Decrement (--): Decreases the value of a variable by 1.
   - Logical complement (!): Inverts the boolean value of an operand.
2. Binary Operators: Binary operators act on two operands. They can perform arithmetic, relational, logical, assignment, and other operations. Some examples of binary operators are:
   - Addition (+), subtraction (-), multiplication (*), and division (/): Perform arithmetic operations.
   - Greater than (>), less than (<), equal to (==), and not equal to (!=): Compare values.
   - Logical AND (&&), logical OR (||): Perform logical operations.
   - Assignment (=): Assign a value to a variable.
3. Ternary Operator: The ternary operator (?:) is a special operator that takes three operands. It is the only ternary operator in Java and is used for conditional operations. The syntax of the ternary operator is condition ? expression1 : expression2. It evaluates the condition and returns expression1 if the condition is true, otherwise it returns expression2.

Q3. What is the use of Switch case in Java programming?

```
Switch(case){
case 1: {}
Case 2:{}
}
```

Q4. What are the conditional statements and use of conditional statements in Java?

The three types of conditional statements in Java are:

4. if statement: It executes a block of code if the given condition is true.
5. if-else statement: It executes one block of code if the condition is true, and another block of code if the condition is false.
6. switch case statement: It selects one of many code blocks to be executed based on the value of a variable or expression.

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

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

```
}
```

Q6. How do you compare two strings in Java?

```
Using equals() method:
String str1 = "Hello";
String str2 = "World";

if (str1.equals(str2)) {
   // Strings are equal
} else {
   // Strings are not equal
}
2 : compareTo
str1.compareTo(str2);
```

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

```
StringBuffer sf = new StringBuffer("hi");
StringBuilder sb = new StringBuilder("Hello");
sb.append(" World");
sb.insert(5, ",");
sb.delete(5, 6);

String result = sb.toString();
System.out.println(result); // Output: Hello, World
```

Q8. Write a program to sort a String Alphabetically:

```
public class StringSort {
   public static void main(String[] args) {
      String str = "programming";
      char[] chars = str.toCharArray();

      Arrays.sort(chars);

      String sortedStr = new String(chars);
      System.out.println("Sorted String: " + sortedStr);
   }
}
```

Q9. Write a program to check if the letter 'e' is present in the word 'Umbrella':

```
public class CheckLetter {
   public static void main(String[] args) {
      String word = "Umbrella";
      boolean isPresent = false;

      for (int i = 0; i < word.length(); i++) {
         if (word.charAt(i) == 'e') {
            isPresent = true;
            break;
         }
      }
```

```
        if (isPresent) {
            System.out.println("'e' is present in the word.");
        } else {
            System.out.println("'e' is not present in the word.");
        }
    }
}
```

Q10. Where exactly is the string constant pool located in the memory?

In Java, the string constant pool is a part of the runtime constant pool, which is a special area of memory allocated by the JVM (Java Virtual Machine). The string constant pool stores literal string values (strings created using double quotes "") and string literals used in the program.

The string constant pool is located in the heap memory, which is the runtime data area where objects are allocated. It is a special area within the heap that stores string objects to optimize memory usage.

The string constant pool is shared among multiple string objects, meaning that if multiple string objects have the same value, they will refer to the same memory location in the string constant pool. This allows string literals to be compared using reference equality (==) instead of comparing their content.

By keeping string literals in a separate pool, Java can optimize memory usage by reusing common string values and reducing the number of duplicate string objects created in memory.