1. In Java, conditional operators are used to perform conditional or ternary operations.

Ternary Operators (? :) : The ternary operator is the only conditional operator that Java has. It takes three operands and evaluates a boolean expression, returning one of two possible values based on the result of the evaluation. It’s expression is:

condition ? value1 : value2

There are two more conditional operators in java which are conditional OR and conditional AND.

2. 1. Unary Operators: These include increment, decrement, logical complement, unary plus, unary minus, bitwise complement operators.

2. Binary Operators: These include arithmetic operators (Addition, subtraction, multiplication, division, modulus) and relational operators (Equality, inequality, greater than, less than).

3. Logical Operators: These include logical OR and logical AND operators.

4. Assignment Operators: These include assignment and compound assignment operators.

5. Bitwise Operators: These include bitwise OR, AND, XOR operators.

3. The switch statement in Java is used to make decisions based on the value of a variable or expression. It provides a convenient way to select one of many code blocks to execute, depending on the value of a given expression. The switch statement in Java is a powerful control flow mechanism that allows you to select one of multiple code blocks to execute based on the value of an expression. It provides a structured and efficient way to handle different cases and improve code readability.

4. Conditional statements in Java are used to control the flow of execution based on certain conditions. They allow you to make decisions and perform different actions depending on the evaluation of a boolean expression. The main conditional statements in Java are:

a) IF Statement

b) IF-ELSE Statement

c) IF-ELSE-IF Statement

5. The if-else statement extends the if statement by providing an alternative block of code to execute if the condition is false. The syntax of an if-else statement is as follows:

if (condition) {

// Code to be executed if the condition is true

} else {

// Code to be executed if the condition is false

}

6. In Java, you can compare two strings using the **equals()** method or the **compareTo()** method.

a) 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 characters in the same order, and **false** otherwise.

b) Using the **compareTo()** Method: The **compareTo()** method is used to compare two strings lexicographically. It returns an integer value indicating the relationship between the strings. If the strings are equal, it returns **0**. If the calling string is lexicographically less than the specified string, it returns a negative value. If the calling string is lexicographically greater, it returns a positive value.

7. In Java, the **String** class is immutable, which means that once a string object is created, its value cannot be changed. However, there is another class called **StringBuilder** that provides mutable string operations.

The **StringBuilder** class is part of the Java Standard Library and is used when you need to perform multiple modifications to a string without creating new string objects at each step. It allows you to append, insert, delete, or replace characters in a mutable sequence of characters. Here's an example that demonstrates the usage of **StringBuilder**:

StringBuilder stringBuilder = new StringBuilder("Hello");

// Append characters to the string

stringBuilder.append(" World");

// Insert characters at a specific position

stringBuilder.insert(5, ", Java");

// Replace characters

stringBuilder.replace(7, 12, "OpenAI");

// Delete characters

stringBuilder.delete(13, 19);

String finalString = stringBuilder.toString();

System.out.println(finalString);

Output: Hello, OpenAI

8. import java.util.Arrays;

public class AlphabeticalSort {

public static void main(String[] args) {

String input = "openai";

// Convert the string to an array of characters

char[] chars = input.toCharArray();

// Sort the array of characters

Arrays.sort(chars);

// Create a new string from the sorted array of characters

String sortedString = new String(chars);

System.out.println("Original String: " + input);

System.out.println("Sorted String: " + sortedString);

}

}

Output: Original String: openai

Sorted String: aeiopn

9. public class LetterCheck {

public static void main(String[] args) {

String word = "Umbrella";

char target = 'e';

// Convert the word to lowercase (for case-insensitive comparison)

String lowercaseWord = word.toLowerCase();

// Check if the target character is present in the word

boolean isPresent = false;

for (int i = 0; i < lowercaseWord.length(); i++) {

if (lowercaseWord.charAt(i) == target) {

isPresent = true;

break;

}

}

// Print the result

if (isPresent) {

System.out.println("'" + target + "' is present in the word '" + word + "'.");

} else {

System.out.println("'" + target + "' is not present in the word '" + word + "'.");

}

}

}

Output: 'e' is present in the word 'Umbrella'.

10. The string constant pool is located in the Metaspace, which is a separate memory area outside the Java Heap. The string constant pool in Java is a special area of memory that stores string literals. It is a part of the Java Heap, which is the runtime data area where objects are allocated.