Assignment No .2

Q1.Answer:

The conditional operators in Java are as follows:

1. Conditional AND (&&): This operator returns true if both of its operands are true; otherwise, it returns false.
2. Conditional OR (||): This operator returns true if at least one of its operands is true; otherwise, it returns false.
3. Conditional NOT (!): This operator reverses the logical state of its operand.

These operators are used for logical and comparison operations in Java and are commonly used in conditional statements and expressions.

Q2.Answer:

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

1. Unary Operators: Unary operators act on a single operand. They perform operations such as negation, increment, decrement, logical complement, etc.
2. Binary Operators: Binary operators operate on two operands. They perform operations such as addition, subtraction, multiplication, division, assignment, comparison, logical operations, etc.
3. Ternary Operator: The ternary operator is the only operator in Java that takes three operands. It evaluates a Boolean expression and returns one of two possible values based on the result of the expression.

These operator types provide different ways to manipulate and operate on values in programming languages.

Q3. Answer:

The switch statement in Java is used for decision-making based on multiple possible values of a single variable or expression. It provides an alternative to using multiple if-else statements when you have a specific set of values to compare against.

The main purpose of the switch statement is to simplify the code and improve its readability by grouping related cases together. It allows you to write more concise and organized code when dealing with multiple conditions.

The switch statement works as follows:

1. The expression (a variable or an expression that evaluates to a value) is evaluated.
2. The value of the expression is compared to the values specified in each case.
3. If a case value matches the value of the expression, the corresponding code block is executed.
4. The break statement is used to exit the switch statement after executing the code block of a matching case. This prevents fall-through to the next case.
5. If no case value matches the expression, an optional default case can be specified. The code block under the default case is executed when none of the case values match.

Q4.Answer:

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

The main conditional statements in Java are:

1. if statement: The if statement is used to execute a block of code only if a specified condition is true. It allows you to perform an action or a set of actions when a certain condition is met.

Example:

```
int number = 5;

if (number > 0) {

    System.out.println("The number is positive.");

}
```

2. **if-else statement**: The if-else statement extends the if statement by providing an alternative block of code to execute if the condition in the if statement is false. It allows you to perform one set of actions when the condition is true and a different set of actions when it is false.

Example:

```
int number = 5;

if (number > 0) {

    System.out.println("The number is positive.");

} else {

    System.out.println("The number is non-positive.");

}
```

3. if-else if-else statement: The if-else if-else statement allows you to test multiple conditions in sequence and execute different blocks of code based on the first condition that evaluates to true. It is useful when you have multiple mutually exclusive conditions.

Example:

```
int number = 5;

if (number > 0) {

    System.out.println("The number is positive.");

} else if (number < 0) {

    System.out.println("The number is negative.");

} else {

    System.out.println("The number is zero.");

}
```

Conditional statements are used in Java to control the program's behavior based on specific conditions. They enable you to create dynamic and responsive programs that can handle different scenarios and make decisions at runtime. By utilizing conditional statements, you can create logic that executes different blocks of code based on the fulfilment of specific conditions, enhancing the flexibility and control of your Java programs.

Q.5 Answer :

```
if (condition) {

   // True block of condition

} else {

   // False block of condition

}
```

Q.6 Answer :

```
String str1 = "Hello";

String str2 = "World";


if (str1.compare(str2)) {

    System.out.println("The strings are equal.");

} else {

    System.out.println("The strings are not equal.");

}
```

Q.7Answer:

Mutable String

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 String".

Example: StringBuffer, StringBuilder.

```
StringBuffer sb = new StringBuffer(19);

System.out.println(sb.capacity());//19
```

Q8. Answer:

```
public class StringSorter {
    public static void main(String[] args) {
        String input = "GokulDham";
        String sortedString = sortStringAlphabetically(input);
        System.out.println("Sorted string: " + sortedString);
    }
    public static String sortStringAlphabetically(String input) {
        char[] charArray = input.toCharArray();
        for (int i = 0; i < charArray.length - 1; i++) {
            for (int j = i + 1; j < charArray.length; j++) {
                if (charArray[i] > charArray[j]) {
                    char temp = charArray[i];
                    charArray[i] = charArray[j];
                    charArray[j] = temp;
                }
            }
        }
        return new String(charArray);
    }
}
```

Q9. Answer:

```
public class LetterCheck {
    public static void main(String[] args) {
        String word = " bumpers  ";
        boolean isLetterPresent = checkLetter(word, 'e');
        if (isLetterPresent) {
            System.out.println("The letter 'e' is present in the word.");
        } else {
            System.out.println("The letter 'e' is not present in the word.");
        }
    }

    public static boolean checkLetter(String word, char letter) {
        for (int i = 0; i < word.length(); i++) {
            if (word.charAt(i) == letter) {
                return true;
            }
        }
        return false;
    }
}
```

Q10. Answer:

In the Java programming language, the string constant pool is a part of the Java heap memory. More specifically, it is a section within the runtime constant pool, which is a runtime representation of the constant pool in the class file format.

To provide a bit more detail, the Java heap is the runtime data area where objects are allocated, including string objects. Within the heap, the string constant pool is a special area that stores unique string literals, such as string literals declared in your code or string literals loaded from compiled class files.

The exact location of the string constant pool within the heap may vary based on the JVM implementation. It is managed by the JVM and is not directly accessible or manipulable by Java programs. The JVM handles the creation, storage, and retrieval of string literals in the string constant pool during runtime.