Assignment Questions 2

💡 **Q1. What are the Conditional Operators in Java?**

**Ans.:** Conditional operators are used to perform logical operations and comparisons. They are used to check the truth value of an expression and return a boolean value.

The conditional operators in Java are:

* Logical AND (&&): Returns true if both operands are true, otherwise false.
* Logical OR (||): Returns true if at least one of the operands is true, otherwise false.
* Logical NOT (!): Reverses the logical state of its operand. If the operand is true, it returns false, and if the operand is false, it returns true.
* Equality (==): Returns true if the two operands are equal, otherwise false.
* Inequality (!=): Returns true if the two operands are not equal, otherwise false.
* Relational Operators: These include greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=). These operators are used to compare values and return a boolean result.

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

Ans.: Based on the number of operands, operators in Java can be categorized as follows:

* Unary Operators: These operators work with a single operand. Examples include unary plus (+), unary minus (-), increment (++), decrement (--), logical NOT (!), etc.
* Binary Operators: These operators work with two operands. Examples include arithmetic operators like addition (+), subtraction (-), multiplication (\*), division (/), modulus (%), etc. Other binary operators include assignment operators (=), logical operators (&&, ||), relational operators (==, !=, >, <, >=, <=), etc.
* Ternary Operator: The ternary operator (?:) is the only operator in Java that takes three operands. It is used as a shorthand for the if-else statement and helps in making decisions based on a condition.

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

Ans.: The switch case is a control statement in Java used as provides an alternative to multiple if-else statements when there are multiple possible matches. The switch expression is evaluated once, and its value is compared with the values of each case. If a match is found, the corresponding block of code is executed. The break statement is used to exit the switch statement after a match is found. If no match is found, the code inside the default block is executed.

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

Ans.: Conditional statements are used in Java to control the flow of execution based on certain conditions. They allow the program to make decisions and perform different actions based on the outcome of these conditions.

Java has three main types of conditional statements:

* If statement: It checks a condition and executes a block of code if the condition is true. It can be followed by an optional else statement to specify an alternative block of code to execute if the condition is false.
* If-else statement: It checks a condition and executes one block of code if the condition is true, and another block of code if the condition is false. If the condition in the if statement is true, the code inside the if block is executed. Otherwise, the code inside the else block is executed.
* Nested If-else statement: This statement allows you to check multiple conditions and execute different blocks of code based on the conditions. It starts with an if statement, followed by zero or more else if statements, and ends with an optional else statement. The code block associated with the first true condition is executed, and if none of the conditions are true, the code inside the else block is executed.

💡 **Q5.What is the syntax of if else statement?**

Ans.: The syntax of the if-else statement in Java is :

if (condition) {

// code to be executed if condition is true

} else {

// code to be executed if condition is false

}

💡 **Q6.How do you compare two strings in Java?**

Ans.: There are Three methods to compare strings as:

1. By using **equals()**

To check if two strings have the same content, you should use the **equals()** method.This checks the strings for equality considering ‘case’. That means it is case sensitive.

Example:

String str1 = "Name";

String str2 = "name";

boolean isEqual = str1.equals(str2); // Case-sensitive comparison

System.out.println(isEqual); // false

1. By using **equalsIgnoreCase()**

This method is used to ignore case differences.

Example:

String str1 = "Name";

String str2 = "name";

boolean isEqualIgnoreCase = str1.equalsIgnoreCase(str2); // Case-insensitive comparison

System.out.println(isEqualIgnoreCase); // true

1. By using **compareTo():**

The compareTo() method compares two strings lexicographically.The comparison is based on the Unicode value of each character in the strings.The method returns 0 if the string is equal to the other string. A value less than 0 is returned if the string is less than the other string (less characters) and a value greater than 0 if the string is greater than the other string (more characters).

String Str1 = "Hello";

String Str2 = "Hello25152";

System.out.println(Str1.compareTo(Str2)); //-5

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

Ans.: Mutable means changing over time or that can be changed. In a mutable string, we can change the value of the string and JVM doesn’t create a new object. In a mutable string, we can change the value of the string in the same object.

To create a mutable string in java, Java has two classes StringBuffer and StringBuilder where the String class is used for the immutable string

public class MutableString

{

public static void main (String[] args)

{

StringBuffer str1 = new StringBuffer("I am");

StringBuilder str2 = new StringBuilder("Learning");

System.out.println("Value of str1 before change :" + str1);

System.out.println("Value of str2 before change :" + str2);

str1.append("Aryan");

str2.append(" Java");

System.out.println("Value of str1 after change :" + str1);

System.out.println("Value of str2 after change :" + str2);

}

}  
  
The output of the code will be:

Value of str1 before change :I am

Value of str2 before change :Learning

Value of str1 after change :I am Aryan

Value of str2 after change :Learning Java

💡 **Q8.Write a program to sort a String Alphabetically**

Ans.:

public class StringSorter {

public static void main(String[] args) {

String input = "study";

char[] charArray = input.toCharArray();

Arrays.sort(charArray);

String sortedString = new String(charArray);

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

}

}

The output will be:

dusty

💡 **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";

char target = 'e';

boolean isPresent = false;

for (char c : word.toCharArray()) {

if (c == target) {

isPresent = true;

break;

}

}

if (isPresent) {

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

} else {

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

}

}

}

💡 **Q10.Where exactly is the string constant pool located in the**

**memory?**

Ans.: The String constant pool is a special memory area. When we declare a String literal, the JVM creates the object in the pool and stores its reference on the stack. Before creating each String object in memory, the JVM performs some steps to decrease the memory overhead.

The String constant pool uses a Hashmap in its implementation. Each bucket of the Hashmap contains a list of Strings with the same hash code. In earlier versions of Java, the storage area for the pool was a fixed size and could often lead to the “Could not reserve enough space for object heap” error. Stack and heap have different characteristics to store and access data. From memory allocation to its access and availability, a heap is the most suitable area to store the String constant pool.