Assignment-02

Q1. What are the Conditional Operators in Java?

Answer:

There are three types of the conditional operator in Java:

- Conditional AND (&&)
- Conditional OR (||)
- Ternary Operator (? :)

Conditional AND

- The operator is applied between two Boolean expressions. It is denoted by the two AND operators (&&). It returns true if and only if both expressions are true, else returns false.

Conditional OR

- The operator is applied between two Boolean expressions. It is denoted by the two OR operator (||). It returns true if any of the expression is true, else returns false.

Ternary Operator

- The meaning of ternary is composed of three parts. The ternary operator (? :) consists of three operands. It is used to evaluate Boolean expressions. The operator decides which value will be assigned to the variable. It is the only conditional operator that accepts three operands. It can be used instead of the if-else statement. It makes the code much more easy, readable, and shorter.
  Syntax: variable = (condition) ? expression1 : expression2
- The above statement states that if the condition returns true, expression1 gets executed, else the expression2 gets executed and the final result stored in a variable.

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

Answer:

There are three types of operators based on the number of operands.

- Unary operator: If an operator takes one operand, it is called a unary operator.
  E.g.: ++, --, ! etc.
- Binary operator: if it takes two operands, it is called a binary operator.
  E.g.: =, ==, <, > etc.
- Ternary operator: if it takes three operands, it is called a ternary operator.
  E.g.: ? :

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

Answer:

The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement. The switch statement works with byte, short, int, long, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, we can use strings in the switch statement.

- The switch case in Java works like an if-else ladder, i.e., multiple conditions can be checked at once. Switch is provided with an expression that can be a constant or literal expression that can be evaluated. The value of the expression is matched with each test case till a match is found. If there is no match, the default keyword, if specified- the associated code executes. Otherwise, the code specified for the matched test case is executed.

Syntax:

```
switch(expression){
    case value1:
        //code to be executed;
        break;  //optional
    case value2:
        //code to be executed;
        break;  //optional
       ......
    default:
     code to be executed if all cases are not matched;
}
```

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

Answer:

Java has the following conditional statements:

- **if Statement:** Use the if statement to specify a block of Java code to be executed if a condition is true.
  Syntax:
  ```
  if (condition) {
      // block of code to be executed
  }
  ```
- **If-Else Statement:** The if clause evaluates the expression. If it comes out as true, statements under if block gets executed. Else, statements under the else block get executed.
  Syntax:

```
if (condition)
{
        // block of code to be executed }
else
{
        // block of code to be executed
}
```

- **Nested If-Else Statements:** The nested if-else statements are statements that incorporate more than one if-else clause.
  Syntax:

```
if (condition)
{
        //Statements set 1
}
else if (condition 2)
{
        //Statements set 2
}

. . .

else
{
        //Statements to be executed
}
```

- **Switch Statement:** The switch case in Java works like an if-else ladder, i.e., multiple conditions can be checked at once. Switch is provided with an expression that can be a constant or literal expression that can be evaluated. The value of the expression is matched with each test case till a match is found. If there is no match, the default keyword, if specified- the associated code executes.

Otherwise, the code specified for the matched test case is executed.

Syntax:

```
switch(expression){
case value1:
        //code to be executed;
         break;  //optional
case value2:
        //code to be executed;
        break;  //optional
......
default:
        //code to be executed if all cases are not
        matched;
}
```

Q5. What is the syntax of if else statement?

Answer:

- **if Statement:** Use the if statement to specify a block of Java code to be executed if a condition is true.

  Syntax:

  ```
  if (condition) {
          // block of code to be executed
  }
  ```

- **If-Else Statement:** The if clause evaluates the expression. If it comes out as true, statements under if block gets executed. Else, statements under the else block get executed.

  Syntax:

  ```
  if (condition)
  {
          // block of code to be executed
  ```

```
}
else
{
        // block of code to be executed
}
```

- **Nested If-Else Statements:** The nested if-else statements are statements that incorporate more than one if-else clause.

  Syntax:

  ```
  if (condition)
  {
          //Statements set 1
  }
  else if (condition 2)
  {
          //Statements set 2
  }
  . . .
  else
  {
          //Statements to be executed
  }
  ```

Q6. How do you compare two strings in Java?

Answer:

1. By Using equals() Method

   The String class equals() method compares the original content of the string. It compares values of string for equality.

   - public boolean equals(Object another)

     compares this string to the specified object.

- public boolean equalsIgnoreCase(String another) compares this string to another string, ignoring case.

Example:

```
class CompareString{

public static void main(String args[]){

        String s1="Sachin";

        String s2="SACHIN";

        System.out.println(s1.equals(s2));//false
        System.out.println(s1.equalsIgnoreCase(s2));

                //true

        }

}
```

2. By Using == operator
The == operator compares references not values.

Example:

```
class CompareString {
  public static void main(String args[]){
        String s1="Sachin";
        String s2="Sachin";
        String s3=new String("Sachin");
        System.out.println(s1==s2);//true
        System.out.println(s1==s3);//false
   }
}
```

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

Answer:

- 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.
  Example:

```
public class MutableString{
        public static void main (String[] args){
        StringBuffer str1 = new StringBuffer("Sachin");
        StringBuilder str2 = new StringBuilder("Virat");

        System.out.println("Value of str1 before change :" +
        str1);
        System.out.println("Value of str2 before change :" +
        str2);

        str1.append(" tendulkar ");
        str2.append(" kohli");

        System.out.println("Value of str1 after change :" +
        str1);
        System.out.println("Value of str2 after change :" +
        str2);
        }
}
```

Output: Value of str1 before change : Sachin

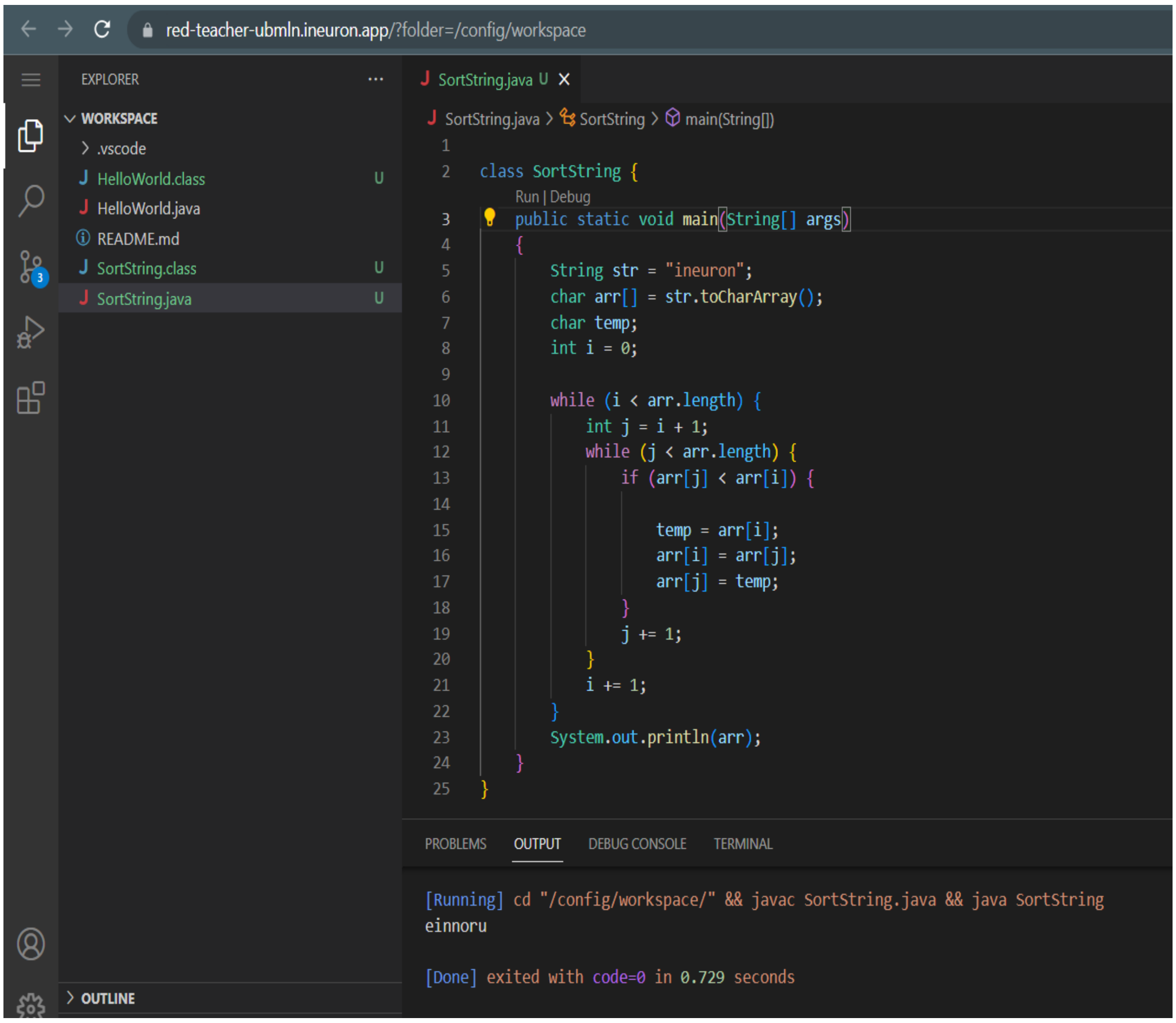
Value of str2 before change : Virat

Value of str1 after change : Sachin tendulkar

Value of str2 after change : Virat kohli

Q8. Write a program to sort a String Alphabetically

Answer:

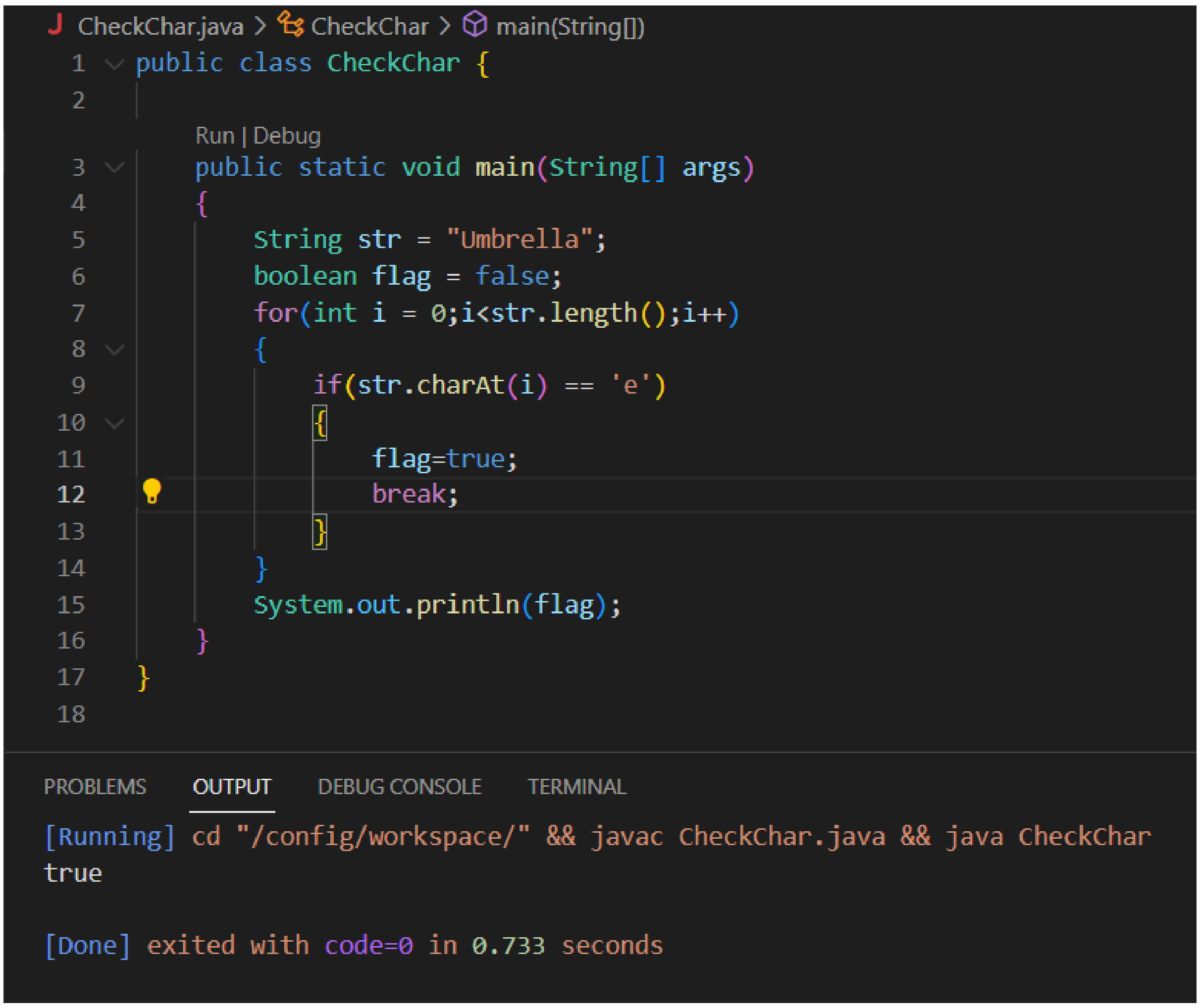


```java

class SortString {
    public static void main(String[] args)
    {
        String str = "ineuron";
        char arr[] = str.toCharArray();
        char temp;
        int i = 0;

        while (i < arr.length) {
            int j = i + 1;
            while (j < arr.length) {
                if (arr[j] < arr[i]) {

                    temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
                j += 1;
            }
            i += 1;
        }
        System.out.println(arr);
    }
}
```

```
[Running] cd "/config/workspace/" && javac SortString.java && java SortString
einnoru

[Done] exited with code=0 in 0.729 seconds
```

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

Answer:



```java
public class CheckChar {

    public static void main(String[] args)
    {
        String str = "Umbrella";
        boolean flag = false;
        for(int i = 0;i<str.length();i++)
        {
            if(str.charAt(i) == 'e')
            {
                flag=true;
                break;
            }
        }
        System.out.println(flag);
    }
}
```

```
[Running] cd "/config/workspace/" && javac CheckChar.java && java CheckChar
true

[Done] exited with code=0 in 0.733 seconds
```

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

Answer:

A string constant pool (SCP) is a separate place in the heap memory where the values of all the strings which are defined in the program are stored. When we declare a string, an object of type String is created in the stack, while an instance with the value of the string is created in the heap. On standard assignment of a value to a string variable, the variable is allocated stack, while the value is stored in the heap in the string constant pool.