# FULL STACK DEVELOPMENT – WORKSHEET 5

## FIND OUTPUT OF THE PROGRAMS WITH EXPLANATION

**Q1.//Stringbuffer**

```
public class Main
{
	public static void main(String args[])
	{
		String s1 = "abc";
		String s2 = s1;
		s1 += "d";
		System.out.println(s1 + " " + s2 + " " + (s1 == s2));
		StringBuffer sb1 = new StringBuffer("abc");
		StringBuffer sb2 = sb1;
		sb1.append("d");
		System.out.println(sb1 + " " + sb2 + " " + (sb1 == sb2));
	}
}
```

Ans: abcd abc false
abcd abcd true

When you compare s1 and s2 using (s1 == s2), it returns false because s1 and s2 are referencing different string objects.
When you compare sb1 and sb2 using (sb1 == sb2), it returns true because sb1 and sb2 are referencing the same StringBuffer object

**Q2.// Method overloading**

```
public class Main
{
	public static void FlipRobo(String s)
	{
		System.out.println("String");
	}
	public static void FlipRobo(Object o)
	{
		System.out.println("Object");
	}

	public static void main(String args[])
	{
		FlipRobo(null);
	}
}
```

Ans: String

When you call FlipRobo(null) in the main method, the Java compiler selects the most specific method that can handle the given argument. In this case, the most specific method is FlipRobo(String s) because null is compatible with String

**Q3.**

```
class First
{
   public First() { System.out.println("a"); }
}

class Second extends First
{
   public Second() {  System.out.println("b"); }
}

class Third extends Second
{
   public Third()  {  System.out.println("c"); }
}
public class MainClass
{
   public static void main(String[] args)
   {
      Third c = new Third();
   }
}
```

Ans: a
b
c

The constructors are called in the order of inheritance, from the most base class (First) to the most derived class (Third), resulting in the output "a," "b," and "c."

**Q4.**

```
public class Calculator
{
   int num = 100;
   public void calc(int num) { this.num = num * 10; }
   public void printNum()     { System.out.println(num); }

   public static void main(String[] args)
   {
      Calculator obj = new Calculator();
      obj.calc(2);
      obj.printNum();
   }
}
```

Ans: 20

The function calc() is called with an argument to by the object 'obj'. Hence, the method is executed and the value of num is updated and printed from the printNum() function.

**Q5.public class Test**

```
{
   public static void main(String[] args)
   {
      StringBuilder s1 = new StringBuilder("Java");
      String s2 = "Love";
      s1.append(s2);
      s1.substring(4);
      int foundAt = s1.indexOf(s2);
      System.out.println(foundAt);
   }
}
```

Ans: 4

Here, the indexOf() method is called on s1 to find the first occurrence of the string "Love" within s1. It returns the index of the first occurrence, which is 4.

**Q6. class Writer**

```
{
        public static void write()
        {
                System.out.println("Writing...");
        }
}
class Author extends Writer
{
        public static void write()
        {
                System.out.println("Writing book");
        }
}

public class Programmer extends Author
{
        public static void write()
        {
                System.out.println("Writing code");
        }

        public static void main(String[] args)
        {
                Author a = new Programmer();
                a.write();
        }
}
```

Ans: Writing book

The a.write(); line calls the write method of the Author class because the reference type Author determines the method to be called. So, "Writing book" is printed to the console.

**Q7.class FlipRobo**

```
{
        public static void main(String args[])
       {
               String s1 = new String("FlipRobo");
               String s2 = new String("FlipRobo");
               if (s1 == s2) System.out.println("Equal");
                else System.out.println("Not equal");
               }
 }
```

Ans: Not equal

Here, s1 and s2 will reference two different String objects with the same content "FlipRobo." 's1==s2' checks if the 2 strings are in the same memory location, which they are not. They are separate string objects. Hence false.

**Q8.class FlipRobo**

```
{
        public static void main(String args[])
       {
               try {
                       System.out.println("First statement of try block");
                       int num=45/3; System.out.println(num);
               }
               catch(Exception e)
              {
                       System.out.println("FlipRobo caught Exception");
               }
               finally
              {
                       System.out.println("finally block");
               }
              System.out.println("Main method");
               }
      }
 }
```

Ans: First statement of try block
15
finally block
Main method

Here, the try block executes first hence "First statement of try block" prints. Then it does 45/3 and then prints "15". Since there is no exceptions, the catch block does not execute and it straight away goes to finally block which executes always.
Then "finally block". After the finally block executes, the control goes back to the method hence "Main method" is then printed

**Q9.class FlipRobo**
```
 {
 // constructor
  FlipRobo()
{
        System.out.println("constructor called");
 }
 static FlipRobo a = new FlipRobo(); //line 8
 public static void main(String args[])
{
 FlipRobo b; //line 12
b = new FlipRobo();
 }
 }
```

Ans: constructor called
constructor called

When the class FlipRobo is loaded, static variables are initialized. This means that the constructor of FlipRobo will be called during class initialization with variable "a" initialized, hence "constructor called" is displayed once. Then we initialise a fliprobo variable b in the main method which leads to another constructor call. hence "constructor called constructor called"

**Q10.class FlipRobo**
```
{
static int num;
static String mystr;
 // constructor
 FlipRobo()
{
 num = 100;
mystr = "Constructor";
 }
// First Static block static
{
 System.out.println("Static Block 1");
num = 68;
mystr = "Block1";
}
 // Second static block static
 {
 System.out.println("Static Block 2");
 num = 98;
 mystr = "Block2";
 }
 public static void main(String args[])
 {
 FlipRobo a = new FlipRobo();
 System.out.println("Value of num = " + a.num);
 System.out.println("Value of mystr = " + a.mystr);
}
```

Ans: Static Block 1
Static Block 2
Value of num = 100
Value of mystr = Constructor

When the class FlipRobo is loaded, static variables and static blocks are initialized in the order they appear in the code, ie, "Static Block 1" is printed and then "Static Block 2" is printed. After the instance is created, you print the values of num and mystr using the reference variable a. The value of num is 100 because the constructor set it to 100. The value of mystr is "Constructor" because the constructor set it to "Constructor."

```
        }
}
```