FULL STACK DEVELOPMENT – WORKSHEET 5

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));
    }
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q1.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q1
abcd abc false
abcd abcd true
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

String class is immutable while the Stringbuffer class is mutable which means the content can be changed without creating. It uses heap memory hence all the objects point towards the single reference. Hence if we change the content it reflects to all the objects. While for String it is not the same and it uses individual objects for a location. Hence if we change s1, s2 was not updated.

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);
	}
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q2.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q2
String
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

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();
        }
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q3.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q3
a
b
c
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

Since all the methods are default constructors and are inherited, when object for the latest class is created all the parent class constructors are invoked along with the constructor for the last child class.

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();
        }
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q4.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q4
20
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

Though we have num assigned value 100, we are passing another parameter as num with value 2 from main method. This value is assigned to the "num" variable in the method parameter. Later the value of "num" in class with value is over written by the product of the parameter value and 10 i.e., 2*10. Hence the result is 20.

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);
        }
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q5.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q5
4
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

Substring is a method that returns the String type value from the main string. Here s1 is not a string type, it is a String Builder class. Hence the index of s2 in s1 is still at 4. Java**(L)**ove.

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();
	}
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q6.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q6
Writing book
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

Here the object is created with Author as refrence hence the extended method in Author class is called.

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");
	}
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q7.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q7
Not equal
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

We are comparing the object references, not the values. Hence they are not equal.

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");
	}
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q8.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q8
First statement of try block
15
finally block
Main method
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

Because 15 is an integer returned by dividing 45 by 3. The division was successful. Hence no exceptions were caught.

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();
	}
}
```

Output:

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q9.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q9
Constructor Called
Constructor Called
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

Constructors are invoked immediately the objects are created

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);
        }
}
```

Output

```
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> javac Q10.java
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5> java Q10
Static Block 1
Static Block 2
Value of num = 100
Value of mystr = Constructor
PS C:\Users\suhaila\iCloudDrive\Development & Coding\FlipRobo\Worksheet 5>
```

Static blocks do not require object to invoke. Hence the Statements are printed. Then the constructor Is invoked with the values being printed. The value of number and string are not changed.