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`

Here String s1 is created as String literal way. String is immutable and can't be changed once created. So String s2 is assigned with the value of s1, i.e. "abc". So Now the s1 +=d is performed on s1. So "d" is added to s1 and also stored in s1. And new value " abcd" is stored in s1. So in next print line it is printed s1=abcd, now in s2 there is already "abc", and no operation on s2. So s2 get printed as "abc" as it is. And so s1 is not equal to s2, so false.
StringBuffer is String Class that is growable and writable, so when "d" is appended in sb1, it also gets reflected in sb2, without creating new object for sb1.So sb1 is equal to sb2.

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,
In method Overloading, it allows same name method, and run the method that matches with given parameter. Here null is taken as String parameter and not the Object, So the method with String S is

running, which prints “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
Here First, Second & Third are constructors, as the name is same as their class name and there is no return type. They are no argument constructors. Now the constructor is called when object of the class is created, so when Third c = new Third(); is written, it calls constructor(block of code) Third, which in inherited second, and second inherited First, So it is printed as a,b & 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
Method calc is called with argument as 2, which gives this.num = 2 * 10 = 20,
Now num has become 20(current instance), instead of 100(local variable). It is because of this keyword. And calling printNum gives 20 printed on console. Here “this” is reference variable, that refers to current object on which method is being invoked.

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 foundAt integer is assigned as s1.indexOf(s2), indexOf returns the number of index of passed parameter string(it is s2 here),
Now StringBuilder is again mutable, so s1.append(s2) makes s1 = JavaLove,
S1.substring(4) means starting index of subString is 4,
S0 foundAt becomes 4 here.

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
Static method can't be overridden, and "a" is Author referenced type, so Author class method is called, which prints "Writing book".

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 new keyword is used to for creating String. So for both s1 & s2, new objects are created in heap memory area, evenif the value of string is same. "==" operator matches references and here s1 & s2 refers to different objects, so s1==s2 results in false. And else part is executed.

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)
		{
				{
		}
		finally		}
	}
}
```

```
System.out.println(
"FlipRobo caught
Exception");          System.out.println("finally block");

                System.out.println("Main method");
```



```
        }
}
```

Ans 15

Finally block

Main method

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

Everytime object of constructor class is instantiated, method is called, 1st print for “a” and second for “b”

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
Object of class is created, constructor method is called. Static block is executed when class is loaded in memory. So after object "a" created for class FlipRobo, static block is called, which prints "Static block 1 2" and then value of num & mystr : as in Constructor : 100 & constructor.
Static Methods can access class variables without using object of the class. Since constructor is called when a new instance is created so firstly the static blocks are called and after that the constructor is called.

```
    }
}
```