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:-

he output of the program will be:

abcd abc false

abcd abcd true

Explanation:

In the first part of the code, two String variables, s1 and s2, are created and assigned the value "abc". Initially, both s1 and s2 are pointing to the same object in memory.

When we concatenate "d" to s1 using the += operator, a new String object is created with the value "abcd" and assigned to s1. However, s2 still points to the original object "abc". Therefore, when we print

out s1, s2, and the comparison (s1 == s2), we get the output "abcd abc false".

In the second part of the code, two StringBuffer objects, sb1 and sb2, are created and assigned the value "abc". Initially, both sb1 and sb2 are pointing to the same object in memory.

When we use the append() method to concatenate "d" to sb1, the original object "abc" is modified. Since sb2 is also pointing to the same object, both sb1 and sb2 now have the value "abcd". Therefore, when we print out sb1, sb2, and the comparison (sb1 == sb2), we get the output "abcd abcd true".

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:-

The output of the program will be:

String

Explanation:

In this code, there are two overloaded methods with the name "FlipRobo". One method takes a String parameter and the other method takes an Object parameter.

In the main method, we call the FlipRobo method with a null argument. Since null can be interpreted as either a String or an Object, the compiler will prioritize a specific parameter type over a more general parameter type.

In this case, the method with the String parameter is more specific than the method with the Object parameter. Therefore, the method FlipRobo(String s) will be called and the output "String" will be printed

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:-

The output of the program will be:

a
b
c

Explanation:

In this code, there are three classes: First, Second, and Third. Each class has a constructor that prints a letter: "a" for First, "b" for Second, and "c" for Third.

In the MainClass, we create an instance of the Third class by calling its constructor

with the "new" keyword. When we create an instance of the Third class, it inherits the constructors and behavior from its superclass (Second), which in turn inherits from its superclass (First).

Therefore, when we create an instance of the Third class, the constructors of all three classes will be called in order: First, then Second, then Third. This results in the output "a", "b", "c" being printe

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:-

The output of the program will be:

20

Explanation:

In this code, there is a Calculator class with an instance variable `num` and two methods: `calc` and `printNum`.

In the `calc` method, the parameter `num` is multiplied by 10 and assigned to the instance variable `num` of the Calculator object.

In the `printNum` method, the value of the instance variable `num` is printed.

In the `main` method, we create an object of the Calculator class named `obj`. We then call the `calc` method on this object with the argument 2. This means that the `calc` method will assign the value 2 * 10 = 20 to the instance variable `num`.

Finally, we call the `printNum` method on the `obj` object, which will print the value of the instance variable `num`, which 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);
   }
}
```

Ans:-

The output of the program will be:

4

Explanation:

In this code, a StringBuilder object `s1` is created with the initial value "Java" and a String object `s2` is created with the value "Love".

The statement `s1.append(s2)` appends the string "Love" to the StringBuilder `s1`. After this line, the value of `s1` will be "JavaLove".

The statement `s1.substring(4)` returns a new string starting from index 4 of `s1`, which is "Love". However, this returned substring is not stored in any variable, so it does not affect the value of `s1`.

The statement `s1.indexOf(s2)` returns the index of the first occurrence of `s2` (which is "Love") within `s1`. Since "Love" starts at index 4 in `s1`, the value of `foundAt` will be 4.

Finally, the value of `foundAt` is printed, 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:-

The output of the program will be:

"Writing book"

Explanation:

In this code, there are three classes: Writer, Author (which extends Writer), and Programmer (which extends Author).

The Writer class has a static method named "write" that simply prints "Writing..." to the console.

The Author class overrides the "write" method and prints "Writing book" to the console.

The Programmer class also overrides the "write" method and prints "Writing code" to the console.

In the main method, an object of type Author is created but is assigned a Programmer object. This is allowed because Programmer is a subclass of Author.

When the write() method is called on the object a, it invokes the overridden write() method of the Programmer class because the actual runtime type of the object is Programmer. Therefore, "Writing code" 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:-

The output of the program will be:

"Not equal"

Explanation:

In this code, two String objects `s1` and `s2` are created using the `new` keyword. Although the content of `s1` and `s2` is the same ("FlipRobo"), they are two separate objects in memory.

The statement `s1 == s2` compares the references of the two String objects. Since `s1` and `s2` are different objects in memory, the comparison will return false.

Therefore, the output will be "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");
	}
}
```

Ans:-

Output:

First statement of try block

15

finally block

Main method

Explanation:

In this code, the try block contains two statements. The first statement prints "First statement of try block". The second statement calculates

the result of the division 45/3, which is 15, and prints it.

Since there are no exceptions thrown in the try block, the catch block is not executed. However, the finally block is always executed, regardless of whether an exception is thrown or not. In this case, it prints "finally block".

After the finally block, the program continues executing the next statement, which is printing "Main method". Therefore, the output is "Main method".

Note: Since there are no exceptions thrown in the try block, the catch block is not executed. If an exception is thrown, it will be caught by the catch block and the code inside the catch block will be executed.

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:-

The output of the program will be:

"constructor called"

Explanation:

In this code, the class `FlipRobo` has a constructor that prints "constructor called" when it is called.

At line 8, a static variable `a` of type `FlipRobo` is declared and initialized with a new instance of `FlipRobo`. This means that the constructor is called and it prints "constructor called".

In the `main` method, at line 12, a variable `b` of type `FlipRobo` is declared but not initialized. Then, at the next line, it is initialized with a new instance of `FlipRobo`. This also calls the constructor and it prints "constructor called".

Therefore, the output of the program is "constructor called" because the constructor is called twice when initializing `a` and `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:-

Output:

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

Explanation:

In this code, there are two static blocks that are executed before the `main` method.

The first static block initializes `num` to 68 and `mystr` to "Block1".

The second static block is executed after the first static block and initializes `num` to 98 and `mystr` to "Block2".

In the `main` method, an instance of `FlipRobo` class is created using the constructor. The constructor sets `num` to 100 and `mystr` to "Constructor".

Therefore, when printing `a.num`, it will output `100` because that's the value set by the constructor.

Similarly, when printing `a.mystr`, it will output `Constructor` because that's the value set by the constructor.