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: Explanation**:**

- In the first case with String, since strings are immutable, modifying s1 creates a new object, leaving s2 pointing to the original object.
- In the second case with StringBuffer, since StringBuffer is mutable, modifying sb1 also affects sb2 since both reference the same object.

Output:

abcd abc false

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: Explanation:

- Java selects the String version of the method because it is more specific when resolving overloaded methods with null as an argument.

Output:

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: Explanation:

The output follows the order of constructor calls in the class hierarchy: first First, then Second, and finally Third. Each constructor prints its respective character.

Output:

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: Explanation:

1. The instance variable num is initially set to 100.

2. The calc(2) method updates the instance variable num to 20 (2 * 10).

3. The printNum method then prints this updated value, resulting in the output 20.

Output:

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: Explanation:

1. The StringBuilder s1 is initially "Java".

2. After appending "Love", s1 becomes "JavaLove".

3. The substring(4) method does not alter s1 because the result is not stored or used.

4. The indexOf(s2) method finds the substring "Love" starting at index 4, so the output is 4.

Output:

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: Explanation:

1. The write() method from the Author class is called because static methods are bound to the class type of the reference (Author in this case) rather than the object type (Programmer).

2. As a result, the output is "Writing book".

Output:

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: Explanation

1. The == operator checks if s1 and s2 refer to the same object in memory. Since s1 and s2 are created as separate objects, their references are different, so the condition s1 == s2 is false.

2. Therefore, the output is "Not equal".

Output:

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: Explanation:

1. The try block executes successfully, printing "First statement of try block" and the result of 45 / 3, which is 15.
2. The catch block is skipped since no exception is thrown.
3. The finally block always executes, printing "finally block".
4. The last statement in the main method prints "Main method".

Output:

First statement of try block

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: Explanation:

1. The first "constructor called" is printed when the static variable a is initialized.

2. The second "constructor called" is printed when the object b is created inside the main method.

3. Therefore, the output consists of "constructor called" being printed twice.

Output:

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: Explanation:

1. "Static Block 1" is printed when the first static block executes.
2. "Static Block 2" is printed when the second static block executes.
3. The constructor sets num to 100 and mystr to "Constructor", which are then printed by the main method.

Output:

Static Block 1

Static Block 2

Value of num = 100

Value of mystr = Constructor