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

In jave, String is immutable and string buffer is mutable . so string s1 and s2 both points towards same string "**abc**" at first, after adding **d** => s1 become **abcd** and s2 become **abc.** Hence both are not equal and the result is **false.**

But, in case of string buffer, both sb1 and sb2 both points to same object. As they are mutable change in one string automatically reflects in another as well. So both string buffer points the same even after change => **abcd.** Hence result is **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. String , As null argument passed, so automatically Fliprobo(String s) method will invoke by default. And hence will print 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

As class Third inherit class Second, which inherit class First. with instantiation of class Third object, automatically constructors of class First and Second also invoke and hence we receive output of all three constructors.

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
Here, the class variable name ‘num’ is same as calc() method local variable name(num). so for referencing class instance variable from calc() method, **this keyword** is used. So the output of *this.num=num*10* will return the value 2*10=20 in output.

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, 'append' command will concatenate s1 and s2 which results=> s1="**JavaLove**", and foundAt will return the first occurrence index of s2 (**Love**) in concatenated string s1. Which starts from letter "**L"** which is at index **4,** hence it will provide 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. Writing book
From the line Author a =new Programmer(); here, object of Author class a is created which is referenced by programmer class. (Dynamic method Dispatch). In such case, the method of object class (Author will invoke). Hence we get the output of this method as 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

As, here both s1 and s2 are not pointing toward same string, although the content is same “FlipRobo” . but they are different object locations hence results in Not equal output.

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
```

In the try block as we can see, there is no exception occur, hence it will print the line “First statement of try block” and then value of num **15** in next line. And as we know, the later program will skip catch block as there is no exception and finally block will execute . which print two lines=> “Finally block” and “ 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

here one static block is present in line 8. As we know static blocks automatically execute once the class is load in memory. And constructor execute only after creation of object or instantiation. Hence first static block will run; which create object **a,** which will invoke constructor FlipRobo() corresponding to this object **a.** which will print "**constructor called"** in line 1.

later with execution of line 12, Object **b** will created so constructot FlipRobo() will automatically invoke corresponding to this object **b.** and it will print "**constructor called"** again.

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

Static Blocks are automatically executed once the class is loaded in the memory, and they are executed in the same sequence as they are written in the program. And constructor is called after the instantiation of object is created. Hence first static blocks are called after that constructor is called. Hence we get outputs corresponding to the static block 1 and then static block 2 and later we get outputs from corresponding constructor FlipRobo().