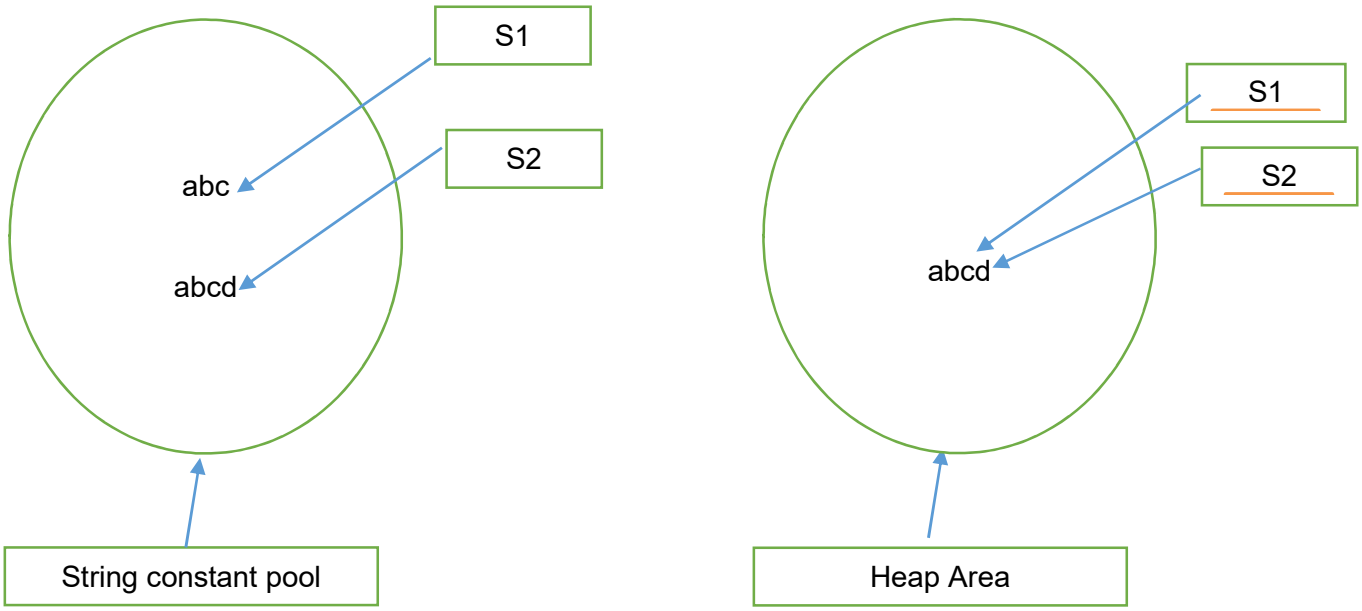
FULL STACK DEVELOPMENT – WORKSHEET 5

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));
        }
}
```

Explanation:



Strings are Immutable so we can't changed string by concanating .
If we concanate strings then another string object created.

In java , we can make strings are mutable by using StringBuffer or StringBuilder class.
So here we can edit or changed strings using append method.

So Output is:

```
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);
        }
}
```

Explanation: In this program there are two overloaded method named as FlipRobo. In first 'FlipRobo' method we passed 'String' parameter and in a second 'FlipRobo' method we passed 'Object' parameter. And main method calls the 'FlipRobo' method with 'null' as an argument.

When a method is called with 'null' as an argument , the java compiler tries to determine the most specific matching method based on the available method signatures. In this case both method can accept 'null' as argument, so there is an ambiguity in choosing the appropriate method.

To resolve the ambiguity, the java compiler uses the most specific method. In this case , 'FlipRobo(String s)' method is more specific .

So , therefore the 'FlipRobo(String s)' method is selected and output will be:

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();
   }
}
```

Explanation:

In this java program, Inheritance concept is used

'Second class' inherited 'First class' by using 'extends' keyword
Means all the properties of 'First class' is acquired by the 'Second class'.
Similarly 'Third class' inherited 'Second class' .
So ultimately Third class acquired all the properties of second and first classes.
Also Created constructor for every class. As we created constructor for every class no need to created object of the derived class.it invoked implicitily.

In the main method we created object of derived class which is 'Third class'.
So Expected output will be

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();
   }

}
```

Explanation: In this Java Program , we have a class named "Calculator" with three methods : 'calc' , 'printNum' , 'main'.

The 'calc' method takes an 'int' parameter named as 'num'. Inside the method, the instance variable 'this.num' is assigned the value of 'num' multiplied by 10. The 'this' keyword refers to the current instance of the class allowing to access the instance variable.
The 'printNum' method simply prints the value of the instance variable 'num'.

In the 'main' method , an object 'obj' of the 'Calculator' class is created . The 'calc' method is called on the 'obj' instance, passing the value 2 as an argument . This sets the value of the instance variable 'num' to 20. Then , the 'printNum' method is called on 'obj', which prints the value of 'num' (20).

Therefore, the output of the program will be 20

This is because the 'calc' method modifies the value of the instance variable 'num' and the 'printNum' method prints the updates value of 'num' within the same instance of the 'Caluculator' class.

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);

   }

}
```

Explanation:

In this is program we have a class named as 'Test' with 'main' method.
We created a String object by using 'StringBuffer class' & s1 variable is refered to that object and we initizlized that object with 'java' literal.

Then we take another string object reffered by S2 and initializing by 'Love' literal.

After that we used 'append method to concatenates to two Strings object , so after contatenation s1 object is 'JavaLove'.
Then used 'substring' method to show substring by providing starting index of substring. So here s1.substring(4)= 'Love'.
Then we used 'indexOf(); to find the starting index of any strings
Here s1.indexOf(s2)=4 and this index is stored in data type 'int' 'foundAt' variable.
& prints the 'foundAt' which 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();
        }
```

Explanation:

In this given program , we have three classes : 'Writer' , 'Author' , 'Programmer'. The 'Author' class extends the 'Writer' class, and the 'Programmer' class extends the 'Author' class.

Inside each class , there is a static method named'write(); that prints a specific message.

In the main method of the 'Programmer' class:

'Author a = new Programmer();' : An object 'a' of type 'Author' is created and assigned an instance of the 'Programmer' class. This is because 'Programmer' is a subclass of 'Author', and Java supports polymorphism.

Now, 'a.write()' method is called on the 'a' object. Since 'a's is of type 'Author', that static method 'write()' in the 'Author' class is invoked.

Therefore output will be Will be Writing book.

This is be is because Java resolves static method calls based on the declared type of the variable, not the actual type of the object at runtime. In this case , even though 'a' is referencing instance of the 'Programmer' class. The 'write()' method in the 'Author' class is called because the variable 'a' is of type 'Author'.

Note: Polymorphism applies to instance methods, not static methods. Static methods are resolved at compile-time based on the declared type.

7) 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”); }}
```

Explanation: In the program we have created two ‘String’ objects, ‘s1’ and ‘s2’, are created using the ‘new’ keyword. Each ‘String’ objects is initialized with the same value, which is “FlipRobo”.
The ‘==’ operator used to compare the two ‘String’ objects. However , when comparing objects using the ‘==’ opearator, it checks if the two objects have the same reference , not if they have the same content.

Since ‘s1’ and ‘s2’ are created using the ‘new’ keyword , they refer to two different ‘String’ objects in memory, even though they have the same content . therefore, the ‘if’ condition ‘s1==s2’ will evaluate to ‘false’, and the code will print “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:
```
First statement of try block
15
finally block
Main method
```

Explanation:

1) In this program we used 'try','catch','finally' keyword to handle the exceptions
2) First execute try block prints the "First stamen of the try block" then check mathematical expression if it is with no exception then prints that calculation. Otherwise prints 'catch' block'
3) 'finally' block is executed at any condition , it will not affected by any exception in the program.
4) Then lastly Main method 'print' executed.

Q.9

```
class FlipRobo
{
        FlipRobo()
         {
                 System.out.println("constructor called");
                 }
         static FlipRobo a = new FlipRobo(); //line 8
         public static void main(String args[])
         {
                FlipRobo b;
                b=new FlipRobo();
        }
}
```

Output: constructor called
constructor called

Explanation:

In this program we have class named as "FlipRobo" with user defined constructor with two static variables.
static variable 'a' , it is assigned to new instance of FlipRobo class. Which triggers to be constructor called.As it is static variable it is executed once during the class initialization.

In main method we declared static variable 'b' and it is assigned to instance of FlipRobo class. Which triggers to be constructor called .

Q10.

```
class FlipRobo

{
        static int num;
        static String mystr;
        // Constructor
        FlipRobo()
        {
                num=100;
                mystr="Constructro";
        }

        // 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);
        }
}
```

Explanation: Given program has class named as "FlipRobo" with static variables and static blocks . Here a breakdown of how the code works.

1) The class "FlipRobo" is defined.

2) There are two static variables : 'num' and 'mystr.'

3) The first static block is executed when the class is loaded. It prints "Static Block 1 " and assigns the value 68 to "num" and the value "Block1" to "mystr".

4) The second block is executed after the first static block. It prints "static block 2" assigns the value 98 to "num" and the value "Block 2" to "mystr".

5) The constructor "FlipRobo()" is defined . It assigns the value 100 to "num" and the value "constructor"to "mystr".

6) The main method is declared, which is the entry point of the program.

7) Inside the main method. An object "a" of the "FlipRobo" class is created using the constructor .

8) The value of "num" and "mystr" in the object "a" are printed using the System.out.println statements.

When the program is executed , the following will be the output:
Output:

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