Q1

```
public class Stringbuffer {

public static void main(String args[]) {

        String s1 = "abc";

        String s2 = s1;

        s1 += " d ";

        System.out.println(s1 + "" + s2 + "" + (s1 == s2)); // 6th line

        StringBuffer sb1 = new StringBuffer("abc");

        StringBuffer sb2 = sb1;

        sb1.append("d ");

        System.out.println(sb1 + "" + sb2 + "" + (sb1 == sb2)); // 10th line

        }

}
```

Ans. `abc d abcfalse`

`abcd abcd true`

Expla:-'String s1 = "abc"; ' , A 'String' object 's1' is created with the value "abc"

'String s2 = s1;' 's2' is assigned to reference the same 'String object as 's1'

's1 += "d";' The '+=' operator creates a new 'String' object with the value "abcd" and assign it to 's1'

In line 6th it print "abcd abc false". The '==' operator checks if 's1' and 's2' reference the same object, which is false because 's1' now references a diff object.

'StringBuffer sb1 = new StringBuffer("abc");' a 'StringBuffer' object 'sb1' is created with the value "abc"

'StringBuffer sb2 = sb1;' 'sb2' is assigned to reference the same 'StringBuffer ' object as 'sb1'

'sb1.append("d");' The 'append' method modifies the existing 'StringBuffer' object 'sb1' by adding "d" to it.

In line 10th it print “abcd abcd true”. in this case ‘sb1’ and ‘sb2’ reference the same ‘StringBuffer’ object, and the modification is reflected in both.

The key difference is that ‘String’ objects are immutable, which means any modification creates a new ‘String’ object, while ‘StrngBuffer’ objects are mutatble and can be modified in place.

This is why ‘s1==s2’ is false. But

‘sb1 == sb2’ is true.

Q2.

```
public class MethodOverloading {

	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.

Expla:- when we call ‘FlipRobo(null)’ in the ‘main’ method, there is an ambiguity in selecting the appropriate method because ‘null can be treated as both a ‘String and an ‘Object’. This results in a compilation error.

To resolve the ambiguity, you can cast ‘null’ explicitly to the type you want to use. Here’s how we can modify the code: {FlipRobo((String) null)}

By casting ‘null’ to ‘String’, we ensure that the ‘FlipRobo(String s)’ method is called, and the output will be “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

Expla. This is define a class hierarchy and “MainClass” demonstrate the behavior of constructors and inheritance. When we create an instance of ‘Third’ in the main method it triggers the constructors in the hierarchy in this order First class = ‘a’, Second class= ‘b’, Third class = ‘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.

Expla:- A ‘Calculator’ class is defined with an instance variable ‘num’, two instance methods ‘calc’ and printNum’, and a ‘main’ method to demonstrate the class.

An instance of the ‘Calculator’ class is created with ‘Calculator obj = new Calculator();’.

The ‘calc’ method is called on the ‘obj’ instance with the argument ‘2’.

Inside the ‘calc’ method, the instance variable ‘num’ is set to ‘2 *10’, which is ‘20’.

The ‘printNum’ method is called on the ‘obj’ instance, which prints the value of the ‘num’ instance variable, 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. 10

Explan:- We have a ‘StringBuilder’ name ‘s1’ and a ‘String’ name ‘s2’.

‘StringBuilder s1 = new StringBuilder(Java);’ We create a ‘StringBuilder’ name ‘s1’ and initialize it with the string ‘Java’.

‘String s2 = “Love”;’ we create a ‘String’ name ‘s2’ and initialize it with the string “Love”.

‘s1.append(s2);’ append the content of ‘s2’ (“Love”) to the ‘StringBuilder s1’. So, now ‘s1’ contains “JavaLove”.

‘s1.substring(4);’ we call the ‘substring’ method on ‘s1’ with an argument of 4. this line doesn’t affect the content of ‘s1’.

‘int foundAt = s1.indexOf(s2);’ we attempt to find the index of the string “Love”(the value of ‘s2’) within ‘s1’. Since ‘s2’ is found within ‘s1’, the ‘indexOf’ method returns the index where “Love” starts in ‘s1’. in this case, it strats at index 4 in ‘s1’.

‘System.out.println(foundAt);’ we print the value of ‘foundAt’, 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. Writing book.

Expla:- It’s override each orther. When we create an instance of ‘programmer’ and call the ‘write’ method on it, the method to execute is determined by the reference type, which is ‘author a = new Programmer();’ in this case.

So, when we call ‘a.write();’ it will execute the ‘write’ method from the ‘Author’ class, which print ‘Writing book’.

Q7.

```
public 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.

Expla:- Here we are creating two string objects ‘s1’ and ‘s2’ using the ‘new’ keyword. When we use ‘new’ to create a string object, it will create a new object in the heap memory, even if the content of the strings is the same. So, in the comparison ‘s1 == s2’ we are comparing the memory references of these two string objects, which will not be the same.

Q8.

```
public 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

Explan:- use of try-catch-finally block- The code enters the try block, and print “First Statement of try block”. Then go to next line and divide 45 by 3, which is valid operation, and assigns the result (15) to the variable ‘num’ and print ‘num’ is 15. And There is no exceptions thrown, so the catch block is not executed.

Now Program enters the finally block and prints “finally block” regardless of whether an exception occurred. And then finally it prints “Main method”.

Q9. //Constructor

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

```
            }
        }
```

Ans. Constructor called

Constructor called

Constructor called

Expla:- We define a class ‘FlipRobo’, now we define a constructor for this class, which print “constructor called’ when an object is created.

Now we define a ‘static’ member ‘a’ of type ‘FlipRobo’ and initialize it by creating an instance of ‘FlipRobo’ on line 8th. This means that as soon as the class is loaded, ‘a’ will be called, printing “constructor called”.

In the ‘main’ method we declare a reference variable ‘b’ of type ‘FlipRobo’ on line 12, but we don’t immediately initialize it with an object. Later, on the next line we create an instance of ‘FlipRobo’ and assign it to the variable ‘b’. This will also call the constructor printing “constructor called” again.

Q10.

```
public class Flip {

        static int num;

        static String mystr; // constructor

        Flip(){

        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[]) {
		Flip a = new Flip();
		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