Worksheet-5

Q:1

```
package helloAll;
 //Stringbuffer
public class Main {

    public static void main(String[] args) {
// main program start
        String s1="abc";
        String s2=s1;
        s1+="d";
        System.out.println(s1+ "" + s2+ "" +(s1==s2));
        StringBuffer sb1= new StringBuffer("abc");
//object creating
        StringBuffer sb2=sb1;
        sb1.append("d");
        System.out.println(sb1+""+sb2+""+(sb1==sb2));

    }

}
//end class



/*Output:
abcdabcfalse
abcdabcdtrue*/
```

Explanation : In Java, String is immutable and string buffer is mutable. So string s2 and s1 both pointing to the same string abc. And, after making the changes the string s1 points to abcd and s2 points to abc, hence false. While in string buffer, both sb1 and sb2 both point to the same object. Since string buffer are mutable, making changes in one string also make changes to the other string. So both string still pointing to the same object after making the changes to the object (here sb2).

Q:2

```
package helloAll;
 //Method overloading
public class Main
{

  public static void FlipRobo(String s)
 {
      System.out.println("String");
 }
      public static void FlipRobo(Object o)
      {

          System.out.println("");
      }

      public static void main(String args[])
//main program
      {
          FlipRobo(null);
      }
}
//class end
```

/*Output:String

Explanation: In this case of method overloading the most specific method is chosen at compile time.
As ‘java.lang.String’ and ‘java.lang.Integer’ is a more specific type than ‘java.lang.Object’,that’s why String is preffered.

Q:3

```
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();
    }
}
```

//OUTPUT:

```
Error: Main method not found in class First, please
define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend
javafx.application.Application
```

Q:4

```
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)
//main program
    {
        Calculator obj=new Calculator();
        obj.calc(2);
        obj.printNum();
    }
```

```
}                                                    //class
end
```

//OUTPUT:20

Explanation : Here the class instance 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 in statement this.num = num * 10, num represents local variable of the method whose value is 2 and this.num represents class instance variable whose initial value is 100. Now in printNum() method, as it has no local variable whose name is same as class instance variable, so we can directly use num to reference instance variable, although this.num can be used.

Q:5

```
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);
    }
}
```

//Output:4

Explanation : append(String str) method,concatenate the str to s1. The substring(int index) method return the String from the given index to the end. But as there is no any String variable to store the returned string,so it will be destroyed.Now indexOf(String s2) method return the index of first occurrence of s2. So 4 is printed as s1=" JavaLove" .

Q:6

```
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();
    }
}
```

//output:

```
Error: Main method not found in class writer, please
define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend
javafx.application.Application
```

Q;7

```
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");
    }

}
```

//output: Not Equal

Explaination: String s1 = "FlipRobo". Here, FlipRobo string will be stored at a location and and s1 will keep a reference to it. String s2=new String("FlipRobo ") will create a new object, will refer to that one and will be stored at a different location.

Explanation: Since, s1 and s2 are two different objects the references are not the same, and the == operator compares object reference. So it prints “Not equal”, to compare the actual characters in the string .equals() method must be used.

Q:8

```
public class FlipRobo {

    public static void main(String[] args) {
        try
        {
            System.out.println("First stament 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 stament of try block
15
finally block
Main method
```

Explanation:

Since there is no exception, the catch block is not called, but the *finally* block is always executed after a try block whether the exception is handled or not.

Q:9

```
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();
     }
     }
```

```
//Output: constructor called
          constructor called
```

Q:10

```
class FlipRobo {
static int num;
static String mystr;

FlipRobo()
{
    num=100;
    mystr="construcutr";
}
 //First Static block
static
{
    System.out.println("Static Block1");
    num=68;
    mystr="Block1";
}
//second static block2
static
{
    System.out.println("Static Block2");
    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);
}
 }
```

```
//Output: Static Block1
        Static Block2
        Value of num=100
        Value of mystr=construcutr
```

Explaination:calling constructor a.num which has num=100 variable , a.mystr call function mystr=constructor printed