Java Test 4

Q1: What is the output of following -

1. int ch = 'Z';
   System.out.print(ch);

2. System.out.println("Happy ");
   System.out.print("Birthday");
3. for (int i = 10 ; i > 10; i ++)
   System.out.println(i);
   System.out.println(i * 10);
4. int x=5;
   int x1= ++x – x++ + --x ;
   System.out.println(“x1= ”+r2);

5. char ch = 'C';
   int b = ch;
   b=b+5;
   System.out.println(b + " " + ch);

A1:

1. **90**
2. **Happy**
   Birthday
3. **100**
4. **Compile time Error: r2 variable not declare.**
5. **72 C**

Q2: What is local variable?

A2: **Local Variable**

A local variable is a variable that is declared with in a method. They can be used only within that method and they cease to exist when the method finishes its execution.

Consider the following example.

<u>Program :</u>

```
class localvariable
{
        public static void main(String args[])
        {
                int c=3; // c is local for main method
                int x=5;
                sum(c,x);
        }
        void sum(int a, int b)
        {
                int c=a+b; // c is local variable for sum method
                System.out.println( “sum =”+c);
        }
}
```

Output:

```
Sum=8
```

Q3: What is java’s default way of calling methods: by value or by reference?
A3: **Call by Reference**

Suppose we would like to change the value of the original variable in the calling program; which is not possible if the call by value method is used. Here arguments that are objects are passed by a reference to the called method. This means that any change made to the object by the called method is reflected in the calling method. In Java objects are passed by use of call by reference.

Program:

```
class sampleTest
{
        int i;
        sampleTest(int j)
        {
                i=j;
        }
        //pass by reference
        void change(sampleTest s)
        {
                s.i=s.i*2;
                System.out.println(“side method i="+s.i);
        }
}

class callbyref
 {
        public static void main(String args[ ])
        {
        sampleTest obj1=new sampleTast(20);
        System.out.prlntln(“Before call i=”+ obj1.i);
        s1.change(obj1); //Change calling here
        System.out.prtntln("After call i="+ obj1.i);
        }
}
```

Output:

```
        Before call i=20
        After call i=40
```

In call by reference, the memory address (reference) of the argument is passed to the called method. So the value is changed and stored on the same address.

Note: Methods can take arguments of different data types, variable of the primitive types are passed by value, while variable of the abstract data type (object) are passed by reference.

Q4: Write a short note on overloading and give example?
A4:
Overloading
Overloading is a way to implementing Polymorphism. Overloading occurs when two or more methods in one class have the same method name but different parameters. Method and Constructor can overload within class.

1. **Overloading of methods**

Methods defined multiple times, each definition, having the same name but accepting different parameters, either in number of parameters (or) types. The compiler knows which method to call based on the parameter it is passed.

Program :

```
class overload
{
void printchar()
        {
                for(int i=0;i<10;i++)
                System.out.print(“a”);
        }
        void printchar(char c)
        {
                for(int i=0;i<10;i++)
                System.out.print(c);
        }
        void printchar(int k,char c)
        {
                for(int j=0;j<k;j++)
                System.out.print(c);
        }
}
class poly
{
        public static void main(String args[ ])
        {
                overload o=new overload();
                System.out.println(“\nNo Argument”);
                o.printchar();
                System .out.println(“\nOne Argument”);
                o.printehar(“b”);
                System.out. println(“\nTwo Argument”);
                o.printchar(5,'c');
        }
}
```

Output:

```
No Argument aaaaaaaaaa
One Argument bbbbbbbbbb
Two Argument ccccc
```

2. **Constructor Overloading**

We already know about the argument constructor. Triangle class declared in the argument constructor as follows.

```
class Triangle
{
	double base;
	double height;
	Triangle()// default constructor
	{
		base=1.0;
		height=1.0;
	}
	Triangle(double b; double h)//constructor
	{
		base=b;
		height=h;
	}
	//double area double area()
	{
	retum(0.5*base*height);
	}
}
```

Q5: How many time the println() statement execute ?

```
		for (int i=3;i<10;i++){
		    System.out.println("my dear friend.");
		    i++;
		}
```

A5:

Output:

```
my dear friend.
my dear friend.
my dear friend.
my dear friend.
```

The println() statement execute 4 times.