Copy constructor

class Complex {

private double re, im;

// A normal parametrized constructor

public Complex(double re, double im) {

this.re = re; this.im = im;

}

// copy constructor

Complex(Complex c) {

System.out.println("Copy constructor called");

re = c.re; im = c.im;

}

}

public class SomeClass

{

public String getName()

{

return lastName;

}

//Copy constructor

public Year getYear()

{

return new Year(leapYear);

}

**}**

\* Copy constructor is used to get a new instance of the class.

\* There may be a chance of altering the original instance if we pass the

original one. To avoid that, using copy constructor we are creating a new instance and will give that back to the calling function.

\* Copy constructor is not needed for immutable objects. Since we cannot alter the immutable object once it is created.

\* Instead of using clone we can use copy constructor.

Abstract and Interface

Class that extends abstract class and implement interface with same variables and methods.

* Within this class the variable of abstract class is invoked.
* For the object of this class, the variable of the interface is invoked.
* Among the concrete method in abstract class and default class in interface the method from abstract class takes preceedance.

Class abstract AbsCls{

Public String name = “abstract”;

Protected printName();

Public printAddress(){syso(“abstract”);}

}

Interface IntCls {

Public String name = “Interface”;

Public printName();

Default printAddress(){syso(“Interface”);}

}

Class ChummaCls extends AbsCls implements IntCls {

Public printName(){syso(super.name);}

}

Class MainCls {

Public static void main(String args[]) {

ChummaCls cls = new ChummaCls();

Syso(cls.name); 🡪 Interface

Cls.printName(); 🡪 abstract

Cls.printAddress(); 🡪 abstract

}