**Department of ECE**

**Subject** : oop's using java **Subject code**:200E0501

**Class**: III YEAR I SEM

## INHERITANCE:

A Inheritance is a Cross stone of object-oriented. Programming and it is implemented hierarchical representation. if we are adding a class and it is maintains desired set of elements. if we are adding two or more classes then all classes must be unique manner.if a class is inherited than that is super class and a class will inheritance then it refers as a Sub class.a super class is always veisonery of sub class. A Super-class represents various variables and instantenous of elements in a Specific program. for example a class will in co-operate to another class which will use a key word name called Extended.

**General form of inheritance:**

```
class A{
    int i,j;
    void show i,j(){
        system.out.println("i and "+ i+" "+j);
    }
}
class B extends A{
    int k;
    void show k(){
        system.out.println("k"+k);
    }
    void sum(){
        system.out.println("i+j+k"+(i+j+k));
    }
}
```

**Sample inheritance of java program**

**Program:**

```
class simple inheritance{
public static void main(string args[]){

    A supob=new A()
    B supob=new B()
    sup obj=10;
    sup obj=20;
  system.out.println("contents of supobj");
  sup.obj.show i,j(){
  system.out.println();
  sub obj=7;
  sub obj=8;
  sub obj=9;
system.out.println("contains of sub obj");
    sub.obj.show i,j();
    sub.obj.show K();
     sub.obj.sum();
     system.out.println("i,j and k are sum of the sub ob",(i+j+k));
  }
}
```

```
output:
        contains of subobj:
        i=10
        j=20
        contains of sub obj:
        i=7
        j=8
        k=9
   i,j and k are sum of the sub ob=24
```

**NOTE:**

In above program supob classes can be used itself and subob classes will access the members of supob as a variables i&j which is called methods.java does not supports many supob classes and sing subob class,it will supports many sub classes and single supob class.

## Packages:

1. A package defines a bundle Time a inovative features such Java Examines
2. as Packages and interfaces.
3. A Package is Container for classes and it keeps class name spaces, which is represented as compartmentalized.
4. For Example, class. a Package must creates a name called as list which will stored Packages without Concern and collide the other name of class will store Packages. else where
5. A Package is also define it is used to to store hyrarichal manner with explicity import new class definitions.
6. The General form of Package represents

   Package Pkg;

   Where,
7. Pkg defines Name of Packages
8. In above following statement we are also represented a Package as

   Package My pkg;

9. Java uses all file directives to store Packages then class files to dedare a class type. which will follow a directory and it store Packages.

10. To finding a Package and CLASS PATH represents a Package is mirror of directory & a class path is defines.is environment Variables. A class Path is also defines an option based between Java and Java c. A Path must find one or more classes.

11. To find a path is represented in windows Enviroment. Then the General

syntax is **: C D My programs/ Java/MyPath**

**program**

```
package mypack{
class Balance{
    String name;
    double bal;
  Balance(String n,double b){
  name =n;
  bal=b;
  void show(){
        if(bal<0){
        system.out.println("------>");
        system.out.println("Name"+$+"bal");
        }
       }
      }
     }
class AccountBalance{
        public static void main(String args[]){
        Balance current[]=new Balance[3];
        current[0]=new Balance["Name1",2500000];
        current[1]=new Balance["Name2",1000000];
        current[2]=new Balance["Name3",50000000];
        for(int i=0;i<3;i++){
        current[i].show();
        }
        }

  output:
           java Mypack.AccountBalance
```

Note:

*To Call account balance and to put My.. Pack directory...

*To combine file a file make shure that resultank of My Pack directory

*To Execute the file and it following" Command is

* Java My Pack Account Balance

*It is necessary My Pack directory thence the account balance is a directory. Part of My Pack

*suppose if we execute itself. in account Balance is wrong.,

* suppose the following command is provide means Java Account Balance is not correct Hence account Balance is a qualifyed of Package Name is My Package.

# ACCESS PROTECTION:

*Access defines a way and approach and protection defines a selter and defenter.

*A Package is used to add then it refers as dimensions through access control.

*Java Provides different state of classes it is fully gained control then it Visibility variables, methods, classes, subclasses, and packages.

*Classes and packages are encapsulating and maintaining name spaces with in a scope of Variables and methods.

*A package acts as a containter for classes then it Subordinate to other packages.

*A class acts as a containter then it refers code & data a class in Java is the smallest stage of obstruction.

*Java characteristics 4 ways of class members which is interplay between classes and Packages as follows:

a) Subclasses with same package

b) Non Subclasses with Some package

c) different neither be subclasses (or)packages

d)Access protection which in consider as 3 access specifiers they are:

i) private

ii)Public

iii)protection.

**PRIVATE:**

A PRIVATE access specifier represents it will anything decleration but cannot seen out said of classes.

**PUBLIC:**

public access specifier represents it will anything decleration ajnd it access any where of data.

**PROTECTION:**

A Protected access Specifier represents to allow the elements then it will declare directly of classes and Subclasses then by declaration of allowing Element is called protected.

**ACCESS CLASS MEMBERS:**

| **Property** | **Private** | **No modified** | **Protected** | **Public** |
|---|---|---|---|---|
| **Same class** | YES | YES | YES | YES |
| **Sub class with same package** | NO | NO | YES | YES |
| **Non-subclass with same package** | NO | NO | YES | YES |
| **Different package with subclass** | NO | NO | NO | YES |
| **Different package with Non sub class** | NO | NO | NO | YES |

Fig: Access class members

EX:

```
package p1{
public class protection{
    int n=1;
    private int n_pri=2;
    public int n_pub=3;
    protected int n_prot=4;

  protection(){
        system.out.println("Base constractor");
        system.out.println("n= "+n);
        system.out.println("n_pri"+n_pri);
        system.out.println("n_pub"+n_pub);
        system.out.println("n_prot"+n_prot);

    }
    }
```

NOTE:

In the above program and it is also consider as first class Package which will provide four integer variable and it acts as actual legal protection made with a variable defined as In that need to consider as default protection mode and it follows n-pri as private, n-pub as Public, n-Pot as protected.

## IMPORT PACKAGE:

*A import defines a mean which deliver the things if a import statement is occur to bringing all classes and packages for visibility purpose.

*If a import statement is occurs a classes must vefers and it follows a name.

*A import statement is convinient of all programmers not to do technically to write complete Java programme.

*If we offer dozens classes then a import statement must lot of saving a classes.

*IF It is a java source code the import Statement is occur immediately package must following class definitions.

*The general form of import staternent is

import pkg1 [.pkg2]. class name/*;

* In the above Syntax pkg1 refers top level packages, pkg2 refers Subordinate packages, it is inside and outside separate packages by using a symbol (.). Finally to Specify explicit class names by using an /is/.*

* Hence our java compilar is easily imports different packages it is also referred as code fragment

it represents as follows.

import Java.io.*/

import Java utility date',

*All Java classes must store in a package then it is called Java All language functions must also to stove package then it is called as java.lang;

EX:

```
package Mypack
public class Balance{
    String name;
    double bal;
  public Balance(String n,double b){
  name =n;
  bal=b;
 public void show(){
    if(bal<0){
    system.out.println("------>");
    system.out.println(name+"$"+bal);
    }
    } }
    }
    import My pack;
    class Test Balance{
    public static void main(String args[]){
    Test Balance=new Test("Name",15000000);
    Test.show();
    }
}
```

## INTERFACE:

*A interface defines to join two things.

*A intentare is also defines Syntatic similarity of classes.But they are lack of methods and variables it must declare without body of Pongsam.

*It means that to define an interface don't make assumsions how to implement.

*Once it defines a number of classes must implement for interface.

*If one class defind then it will implement number of interfaces.

*The definition of interface is also similarity of classest than it represented a general form as follows:

access interface classname {

return type method_name 1 [parameter list];

return type method_name 2 [parameter list];

type final variable name 1 = value;

type final variable name 2 = value;

. .

. .

. .

. .

return type method_name n(parameter list);

type final variable name n=value;

}

*In the above general form of an interface all variables must be declarable inside. hence it refers as implicit public and static. it is meaning full not to change by Implementing a class. hence it is also initalize then it must follows all methods and Variables must implicitly public.

*A simple definition of an Interface by using a method as call back and it is implemented in to access specifier.

```
interface call back
{
public void call back ()
```

In the above definition call back is an interface which follows access specifier is public hence call back is a method b/w parameter and definition.

## IMPLEMENTATION OF INTERFACE:

*One or more classes can be implemented in an interface.

*Implementation is refers as tool.

*If implementation of an interface we can defincclauses it means that to create one or more methods we can define an interface.

*The general form of implementing of an interface also

```
class classname [Extend super class] împlement (interface 1..... interface N)

  {
   //class body
  }
```

EX:

```
Class client implement callback{
    Public void callback(int p){
     System.out.println("Call back called is"+p);
        }
        }
```

# ABSTRACT CLASS:

*Abstract defines a Summary and  Concrete of classes.

*A stiuation is occurs to define a Super class then it will Structuring and abstracting without Implementation of methods.

*If a situation occurs to create a Superclass, then it follows generalised forms of Subclasses.

*A classes is occurs the nature of the method most implementing Sub classes.

*The general form of abstract classas follows abstract

```
Abstract type_name(parameter list){
    Abstract class figure{
    double dim1;
    double dim2;

 figure(double a, double b){
   dim1=a;
   dim2=b;

 }

 double area(){
 }
 }
 class rectangle extends figure{
             rectangle(double a,double b){
             super(a,b);
          double area(){
          system.out.println("Inside of area of the rectangle");
          return dim1+dim2;
          }
          }
class Triangle extends figure{
            triangle(double a,double b)
            super(a,b);
            double area(){
            system.out.println("Inside of area of the triangle");
          return dim1+dim2/2;
          }
          }

class AbstractArea{
    public static void main(Sring args[]){
    Rectangle r=new rectangle(20,20);
    Triangle t=new triangle(30,30);
    Figure fig;
    system.out.println("Area of:"+Figure.area());
    system.out.println("Area of:"+Figure.area());

    }
    }
```

*In the above Program a abstract follows classes and methods.

*In above program a inside of comments in the main declaration is illegal and it is  a type of abstract is figure which was over ride the area's of subclasses suppos we need to prove the areas are not over riden and it shows an complation error.

*A abstract type of figure does not create and declare objects hence we are Providing a reference variable name as figref. it is a variable and it refers specific Object must derived sub classes.

*A classes Occur in Super their was follows reference variable which was over ridden and resolve run time process.

## POLYMERPHISM:

*A poly defines many and merphism defines forms.

*A polymerphism is a nature of result which of maintains happens relationship Through mechanism it happens message Passing inheritance and concept of Subtitutable.

*A purest Polymerphism is defind single function then its applicable to argument with variety types of purest Polymorphism having a function rerfers code body and it is intterpretedable which is defines with different meaning.

*A PolymerPhism is also defines it associates many faces which was hold the valves with different types.

*A Good example of polymorphism & it is associated General form as follows

EX:

```
public class solataire{
    static cardpiles allpies[];
    public void paint (Graphiles g){
    for(int i=0;i<13;i++){
    allpiles[i].display(g);
    }
   }
    }
```

*In the above Program a array is Allpiles and it is represents is solataive than it is a game.

*A array of all piles is also available in candpils and it is a type. A array of all piles which holds the values and it is differ from subclass to parent class.

*The array of a value wchich provides a msg name called as Display it is a method and it is associated dynamic kind of variables but not associated in static class.