OOPS(Module-3)
---
-class and Object
-Inheritance(IS-A)
-Has-A
-call to super
-method overriding
-super keyword
-Abstraction
-Interface
-Encapsulation
-Polymorphism
-TYpe and Object Casting

Object Oriented Programming system:It is an programming methodology which fulfills the project requirement.

-class and Object
---
-A class is a logic Entity (Because it represents logic of APP).
-An Object is physical Entity(Because it contains memory).
-A class is a blue print or template using which we can develop code of APP
-An Object is an Entity which gets created using class and it represents the state and behaviours
-Where state indicates variables(datamembers) and behaviour indicates methods(member function)

Ex: class Student
```
{
 //develop code for Student APP

}
```
Student-Object

| states | Behaviours |
|---|---|
| name | studying() |
| age | writingexams() |
| height | playing() |
| weight | etc |
| marks | |
| contact | |
| etc | |

Ex2:Dog
Ex3:Car
Ex4:Employee

-Differences between class and Object

| class | Object |
|---|---|
| -A class is logical entity | -An Objects is physical entity |
| -class represents logic | -Object represents state and behaviour |

-class can be created using class keyword
-Object can be created using new keyword

-When class is created,Memory will not be allocated
-When Object gets created memory will be allocated (Heap)

-For each module we can create one class
-For one class we can create muleiple Objects.

---

Inheritance Or IS-A relationship

---

-It is the First Pillar of OOPS.
-Inherit means Acquire,posses,access,take etc
-DEf:One class Acquiring the properties of another class is called as Inheritance.
OR
One class is Accessing the properties of another class is called as Inheritance.
-Here properties is defined as methods and variables.

extends keyword

---

-extends is a keyword whicch indicates that we are creating a new class from an existing class.

```
class A
{}
class B extends A//B is created using A
{}
```

-SUper class and Sub class

---

-THe class whose properties are acquired is called as super class or parent class or Base class.
-THe class Who is acquiring the properties is called as child class or Sub class or derive class

```
class Super class
{

}
class  Subclass extends Super class
{

}
```

-Super class contains its own properties
-Sub class contains its own properties as well as properties of super class.

| | WithoutInheritance | With Inheritance |
|---|---|---|
| Ex: | class Father<br>{<br>gold()<br>land()<br>money()<br>car() | class Father<br>{<br>gold()<br>land()<br>money()<br>car() |

```
        }                       }
        class Son                   class Son extends Father
        {                       {
         gold()                   girlfriends()
         land()
         money()                   }
         car()
         girlfriend()
        }
```

-Points to remember
---
-In a program we can have multiple classes when we Compile a program depends on no of classes
we define those many .class files will be generated.
ex:

```
class A
class B extends A
class user
{  main()}
```

-Afer compilation:  A.class
B.class
user.class

-As per convention Above program we should save as user.java because it contains main(

-When we have multiple class in single program we can keep only one class as public
and we should keep our User logic class as public.

-Bussiness logic class->which does not have main()
-User logic class-> which contains main().

Note:
---
-Only non static methods can be inherited static methods cannot be inherited because they will
get loaded only once in static pool area.

-if we create Object of super class i can access only super class methods
-if we create object of sub class we can access both super and subclass properties.

TYPES OF INHERITANCE
---
1.SINGLE LEVEL
2.MULTI LEVEL
3.HIERARCHICAL
4.MULTIPLE
5.HYBRID

Single level
---
```
class Bank
{
  String Accname="Rohan";
```

```
  int acno=89898989;
  public void details()
  {
    System.out.println("Acc name:"+Accname+"\n Accno :"+acno);
  }
}
class Deposit extends Bank
{
  double bal=100;
  public void depamt()
  {
    System.out.println("Dposit amt"+bal);
  }
}
public class Cust
{
 public static void main(String args[])
 {
   Deposit d1=new Deposit();
   d1.details();
   d1.depamt();
}
}
```

Multi-level Inheritance

---

```
class Bank
{
  String Accname="Rohan";
  int acno=89898989;
  public void details()
  {
    System.out.println("Acc name:"+Accname+"\n Accno :"+acno);
  }
}
class Deposit extends Bank
{
  double bal=100;
  public void depamt()
  {
    System.out.println("Dposit amt"+bal);
  }
}
class Withdrawl extends Deposit
{
  double amt=200;
  public void withamt()
  {
   if(amt>bal)
   {
     System.out.println("You cannot withdrawl-balance exceeds");
  }
  else{
```

```
   System.out.println("Collect Amount");
}
}}
public class Cust
{
  public static void main(String args[])
  {
    Withdrawl d1=new Withdrawl();
    d1.details();
    d1.depamt();
    d1.withamt();
}
}
```

Hierarchical Inheritance

---

```
class Bank
{
   String Accname="Rohan";
   int acno=89898989;
   double avabal=100;
   public void details()
   {
      System.out.println("Acc name:"+Accname+"\n Accno :"+acno);
   }
}
class Deposit extends Bank
{
   double depamt=100;
   public void depamt()
   {
      System.out.println("Dposit amt"+depamt);
      avabal=avabal+depamt;
      System.out.println("Total Ava bal is:"+avabal);
   }
}
class Withdrawl extends Bank
{
   double amt=100;
   public void withamt()
   {
    if(amt>avabal)
    {
       System.out.println("You cannot withdrawl-balance exceeds");
  }
  else{
    System.out.println("Collect Amount");
}
}}
public class Cust
{
  public static void main(String args[])
  {
```

```
    Deposit s1=new Deposit();
    Withdrawl d1=new Withdrawl();
    s1.details();
    s1.depamt();
    d1.details();
    d1.withamt();
}
}
```

Output

```
------
C:\Docs\programs>java Cust
Acc name:Rohan
Accno :89898989
Dposit amt100.0
Total Ava bal is:200.0
Acc name:Rohan
Accno :89898989
Collect Amount
```

Multiple Inheritance

---

-Multiple inheritance is One class is inheriting two immediate super classes ata the same time

-But in java a class can extends only one class at a time

-So Multiple inheritance is not possible through classes because of-

1.Ambiguity probleum
2.Diamond probleum
3.Constructor chaining probleum.

-If one class extends two classes and in case if both classes contains same method
than while calling a method JVM will get confuse which class method to call this probleum
is known as Ambiguity probleum.

-Since the structure/shape of class diagram is in diamond form it is also referred as Diamond probleum

-Constructor chaining probleum

---

-Call to super

---

-The process of calling one contructor from another constructor of different class is called as
call to super.
-Call to super must be the first statement in constructor
-Call to super is implict(defaulty added) as well as explictly(added by programmer)
in nature.

Q>Can we inherit Constructors or not?
A.No we cannot inherit constructors because they are not member of a class(MEm of classs are
methods and variables) and constructors are mainly used for intialistion of NSvariable.

Call to super-Implict

---
super() is implict(defaultly added by jvm ) in nature if super class constructor doesnot have any arguments.

program

```
class A
{
  public A()
  {
   System.out.println("A class default constructor");
  }
}

class B extends A
{
  public B()
  {
   //super();defaultly added by JVM
   System.out.println("B class DEfault Constructor");
   }
}
public class Main
{
 public static void main(String args[])
 {
  B b1=new B();
 }
}
```

Output

```
C:\Docs\programs>java Main
A class default constructor
B class DEfault Constructor
```

Call to super-Explict

---
call to super is Explict(we have to add it) in nature if super class constructor contains any arguments

Example

```
class A
{
  public A(int i)
  {
   System.out.println("A class default constructor");
  }
}

class B extends A
{
  public B()
```

```
   {
    super(100);//Explictly we added
    System.out.println("B class DEfault Constructor");
    }
}
public class Main
{
  public static void main(String args[])
  {
    B b1=new B();
  }
}
```

Output

```
--------
C:\Docs\programs>java Main
A class default constructor
B class DEfault Constructor
```

Call to super with 3 classes

```
----------------------------
class A
{
   public A(int i)
   {
    System.out.println("A class default constructor");
   }
}

class B extends A
{
   public B(int i)
   {
    super(100);
    System.out.println("B class DEfault Constructor");
    }
}
class C extends B
{
  public C()
  {
    super(55);
    System.out.println("C class constructor");
  }
}
public class Main
{
  public static void main(String args[])
  {
    C b1=new C();
  }
}
```

Combination of call to this and call to super
---

```
class A
{
   public A()
   {
     this(100);
     System.out.println("A class default constructor");
    }
   public A(int i)
   {
    System.out.println("A class integer constructor");
   }}
class B extends A
{
   public B()
   {
    this(99);
    System.out.println("B class default Constructor");
    }
   public B(int i)
   {
     //super();//def call to super
     System.out.println("B class integer constructor");
   }}

public class Main
{
  public static void main(String args[])
  {
    B b1=new B();
  }
}
```

```
C:\Docs\programs>java Main
A class integer constructor
A class default constructor
B class integer constructor
B class default Constructor
```

Constrcructor chaining probleum
---

```
class A
{
  public A()
  {

 }
}
```

```
class B
{
   public B()
    {

    }
}
class C extends A,B
{
  public C()
  {
    super();//Constructor chaining probleum
  }
}
```

-IN THE ABOVE PROGRAM WHEN CALL TO SUPER EXECUTED JVM WILL GET CONFUSE WHICH SUP ER CLASS CONSTRUCTOR TO CALL
BECAUSE THERE ARE TWO SUPER CLASS HENCE IT GETS CONFUSE THIS PROBLEUM IS KNOWN A S CONSTRUCTOR
CHAINING PROBLEUM.
-THEREFORE DUE TO AMBIGUITY PROBLEUM,DIAMOND PROBLEUM AND CONSTRUCTOR CHHAIN ING PROBLEUM
MULTIPLE INHERITANCE IS NOT POSSIBLE THROUGH CLASS BUT POSSIBLE THROUGH INTERFAC E.

HYBRID INHERITANCE
---
-iT IS A COMBINATION OF MULTIPLE AND HIERARCHICAL INHERITANCE SINCE MULTIPLE IS NOT POSSIBLE
THROUGH CLASSES HYBRID IS ALSO NOT POSSIBLE THROUGH CLASSES.
ADVANTAGES
---
-REUSABILITY OF CODE
-AVOID DUPLICACY OF CODE

NOTE:-
---
-TO EVERY CLASS THERE IS A DEFAULT SUPER CLASS PRESENT,WHETHER WE WRITE IT OR DO N OT WRITE IT.
I.E TO ALL PREDEFINE AND USERDEFINE CLASSES OF JAVA THERE IS A DEFAULT SUPER CLASS CALLED AS OBJECT CLASS.

| WHAT WE CAN SEE | WHAT IT IS ACTUALLY |
|---|---|
| class A | class A extends Object |
| { | { |
| } | } |

-Object class contains 9 methods and these methods are used in multiple classes of java
so instead of defining it separately in all classes they have define it in object class

and make it as super class so that without defining it again and again other class can simply used it.

Note:-
---
-WE can inherited public,protected and default methods
-We cannot inherit private method because they are accessible only within class
-we can inherit non static,final and absract methods
-We cannot inherit static methods because they will be loaded only once in SPA
-We cannot inherit constructors becausse they are mainly used for initialisation and they are not member of class.

HAS-A relation ship
---
-One class containing the reference of another class is called as HAS-A relationship.

```
Ex: class Engine
    {
      //properties of engine
    }

    class Car
     {
        Engine e=new Engine();//Car has-a reference of engine
     }

Ex: class RAM
    {

    }
    class Mobile
    {
      RAM r=new RAM();//Mobile has-a reference of RAM
    }
```

Real Time example
---
For BANK APP
---
```
class Loans
{
   //50methods
}
class HomeLoan
{
  //50 methods
}
class CarLoan
{
  //50 methods
}
class GoldLoan
{
  //50 methods
```

```
}
class MortLoan
{
  //50 methods
}
```

Inheritance

---

```
class Loan
{
  //ALL common methods
}
class TYPEsof loans extends Loan
{

}
```

-Tell me in ur project where u used inheritance?

TODAY'S TASK

---

-FIRST COMPLETING NOTES
-START LEARNING FROM METHODS
-STATIC

---

HOW TO CREATE,HOW TO CALL
nONSTATIC
---
HOW TO CREATE,HOW TO CALL
OBJECT CREATION
CONSTRUCTORS
---
HOW TO DEFINE CONSTRUCTORS
TYPES
WHAT IS USE OF IT
HOW IT WILL WORK(REFER DIAGRAM)

ASSIGNMENT
---
CREATE AN EXAMPLE PROGRAM
-FOR STATIC METHODS
-FOR NON STATIC METHODS
-FOR DEFAULT CONSTRUCTOR
-FOR PARAMETERIZED CONSTRUCTOR