(24) static and inheritance in Java

① When to use Static and instance Variables :- ↳ Hyderabbas Sir ↳ 11th Nov live class

① Memory for static variable and instance Variable are allocated on heap area.

↳ static variable generally in heap area only once at the time of class loading.

↳ for Instance Variable Memory allocated inside the objects

↳ whenever new object created one more time Memory will be created.

Loan Application for giving loans to farmers

↳ When to use static variables and instance Variables ?

Principle amount, time duration, rate of interest, simple interest } different for all farmers. So we have to use Specific Variable but not static

#3 Things Activities are performed.

① Asking farmers to Enter principle amount and time duration required.
② Calculate simple interest.
③ display simple interest.

Code :-

```
import java.util.Scanner
Class farmer {
Private float Pa;
Private float td;
Private float Si;
Private static float ri;

Void input() {
  Scanner scan = new Scanner(System.in);
  S.o.p("kindly Enter the principle amount");
  Pa = scan.nextfloat();
  S.o.p("kindly Enter time duration required");
  td = scan.nextfloat();
}
static {
  ri = 2.5;
}
Void Compute() {
  Si = (Pa * td * ri) / 100; }
Void disp() {
  S.o.p("SI is " + Si);
}
Public class static {
    Public static Void main(String() args){
    Farmer f1 = new Farmer();
    f1.input();
    f2.Compute();
    f2.disp();
}
```

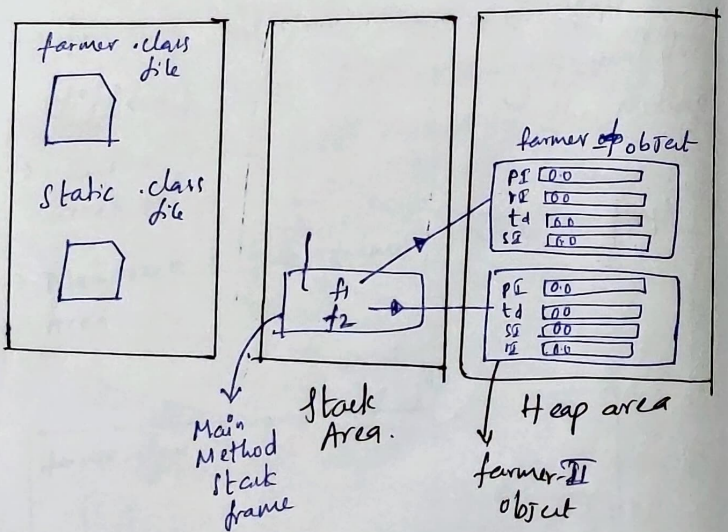
Principle amount, time duration, Simple interest are Specific hence they are store as instance Variable

rate of interest ri is generic hence it is Store in static

→ for Every interface, Enum and class different .class files will be generated

→ Entire class data present in the Method area

→ Main Method Stack frame Created in Stack area.



→ Rate of interest is generic hence it stored in static Variable.

→ Whenever a Common copy of data ~~member~~ has to be accessed by all ~~the~~ the objects of class that data Member can be static

→ static Method are also Called Generic Method.

→ Non-static Method are also Called Specific Method.

② Inheritance :- is a Mechanism in which one object acquires all the properties and behaviour of a parent object.

→ idea behind Inheritance in Java is that you can Create new classes that are built upon Existing classes. When you inherit from an Existing class, you can reuse Methods and fields of the parent class.

→ Moreover you can add new Method and fields in your Current class also.

Why use inheritance in Java :-

① for Method overriding (so runtime polymorphism can be achieved.

② for. Code Reusability.

→ Inheritance ⇒ Code Reusability

↳ Related Concepts
- ↳ polymorphism
- ↳ Abstraction
- ↳ Interfaces

Code - ① No Relationship between classes.

```
Class Demo1  → class ①
{
   String name = "Hyder";
   int age = 28;
   Void disp()
   {
    System.out.Println("Demo1" +agetname);
   }
}
Class Demo2  → class ②
{

}
Public class Inherit1 {
      Public static Void Main(String[] args){

        Demo2 demo = new Demo2();
        demo.disp();
      }
}
```

we get Error.
we created object of Class Demo2 and called Method which is inside Demo1 called class

→ In Java between ~~two~~ classes and interfaces we can Establish relationship.

Establishing Relationship between Class ① & ②

Code ①

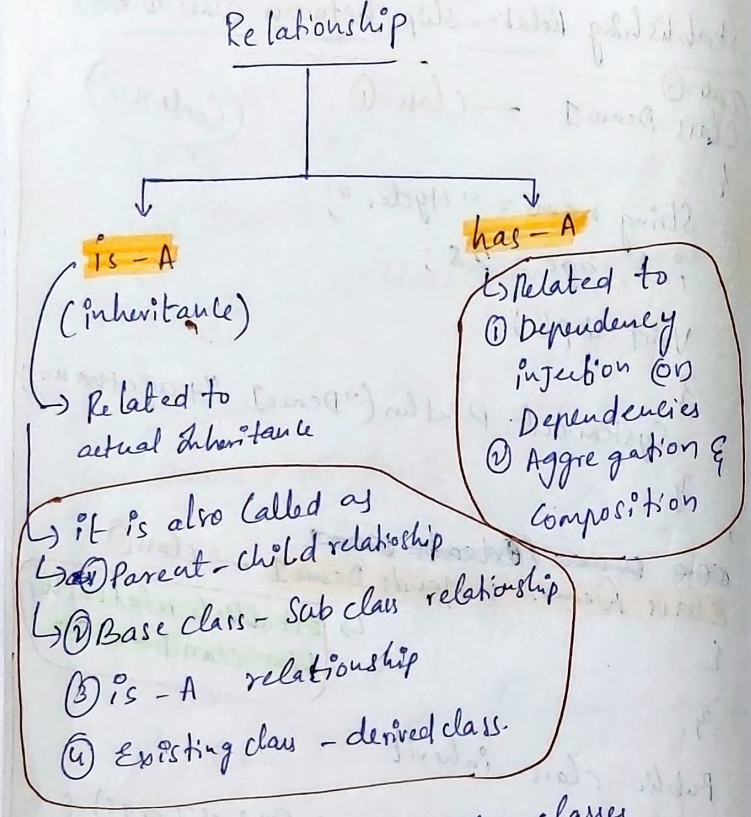
```
Class Demo1  — class ①
{
   String name = "Hyder";
   int age = 28;
   Void disp()
   {
    System.out.Println("Demo1 "+agetname);
   }
}
~~class Demo2 Extends Demo1~~
Class Demo2 Extends Demo1  → class ②
{

}
Public class inherit
{
  Public Static Void main(String[] args) {
      Demo2 demo = new Demo2();
      demo.disp();
  }
}
```

↳ Establish relationship b/w class Demo1 & 2

Code ②

→ Java Supports Relationships.



→ usually we have unrelated classes
→ we develop relationship between classes
→ due to this Code-reusability takes place

① usually we have unrelated classes.
↳ we develop relationship between classes
↳ Then Code reusability happen.
→ this relationship is established using
① Extend Keyword.
Inheritance :- refer code ①
→ All properties and behaviour will get inherited in Demo2.
→ The Moment Relationship between Demo-1 and Demo-2 class is Established
→ all properties and behaviour of in Class Demo-1 will get inherited in Demo-2
→ behind the scenes it happen.
→ This only called Code reusability (or) Concept of inheritance
→ inheritance refers to process of one Class acquiring properties and behaviour of another class.
→ writing a project in heerarchy of classes.

① to achieve inheritance (or to build relationships between class we use Extend keyword

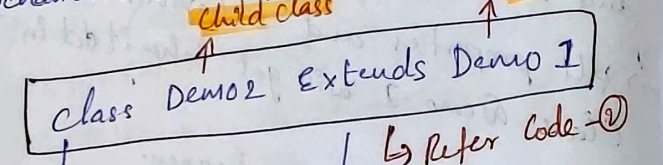
→ Demo 2 ⇒

→ Class before the Extend keyword the class who want to reuse the code.

→ Demo1 ⇒

→ Class After the Extend keyword from this class properties and behaviours will get inherited.

Eg:-



↳ Refer Code - ⑦

it creates relationship between Class Demo-1 & 2.

Child class (or) Sub (or) derived class

⇒ Which Ever class is going to take Properties is called Sub-Class (or) child class (or) derived class

→ Demo2 is Sub-class

② Parent (or) Base (or) Existing class

↳ from this class properties and behaviours will get inherited

→ Demo 1 is parent class.

Code :-

Parent class (or) Base (or) Existing class

```
Class Demo1 {
    String name = "Vaishi";
    int age = "23";
    Void disp()
    {
        S.o.p("Demo 1 " +age +name);
    }
}
```

Child class

```
Class Demo 2 { Extends Demo 1 {
}
```

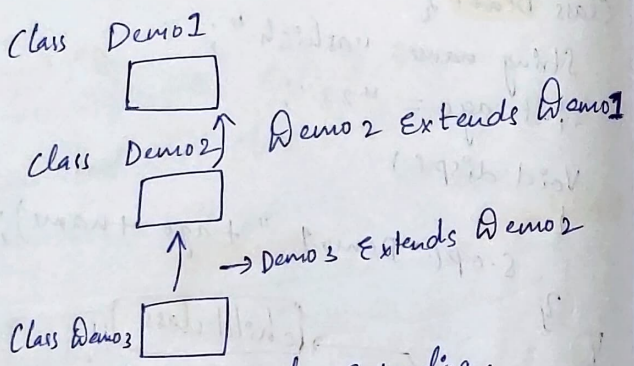
```
Public class Inherit 1
{
    Public static Void main(String[] args)
    {
        Demo2 demo = new Demo2();
        demo.disp();
    }
}
```

Keypoints :- Inheritance (is-A relationship)

① Single inheritance is allowed
(one class Extends another class)

② object class is parent of all classes

③ Multi level Inheritance is allowed;



→ There is restriction for Extending.

Code:-

```
Class Demo11 {
    String name = "Ashish";
    int age = 23;
    Void disp() {
S.o.p("Demo1"+name+age);
}
Class Demo12 Extends Demo11 {

}
Class Demo13 Extends Demo12 {

}
```

Class Code:- Multilevel inheritance

```
Class Demo11 {                          → class-①
    String name = "Ashish";
    int age = 23;
    Void disp() {
    S.o.p("Demo1 "+name+age);
    }
}
Class Demo12 Extends Demo11 {           → class②
                ↓ Child class   ↓ Parent class
}
Class Demo13 Extends Demo12 {           → class③
                ↓ Child class   ↓ Parent class
}

Public class Inherit3 {
    Public static Void main(String [] args)
    {
    Demo13 demo = new Demo13();
    demo.disp();
    }
}
```

output:
Demo1 Ashish 23

demo. → Methods in object Class will be visible
↳ object class contains all these Methods

→ object Class is parent of all classes.

→ In Java object class is parent of all classes.

→ object class contains Methods (inbuilt)

→ In Java if Extend keyword is not used after class by default behind the seen it is Extending object class

→ What Ever properties and behaviour of object class are there in Demo11, Dem12 and Demo 13

Eg: object class

Parent class

```
Class Demo11 Extends object
{                  ↓child class
    String name = "ashish";
    int age = 28;
    Void disp()
    {
        System.out.println("Demo1"+name+age);
    }
}
Class Demo12 Extends Demo11
{                  ↓Child class      ↓Parent class
}
Class Demo13 Extends Demo12
{                  ↓class class      ↓Parent class.
}
```

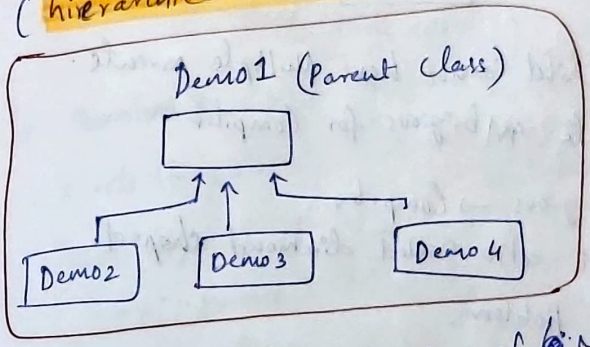
behind the Seceens it is Extending object class

→ Properties and behaviour of Demo11 and object class are inherited in Demo12.

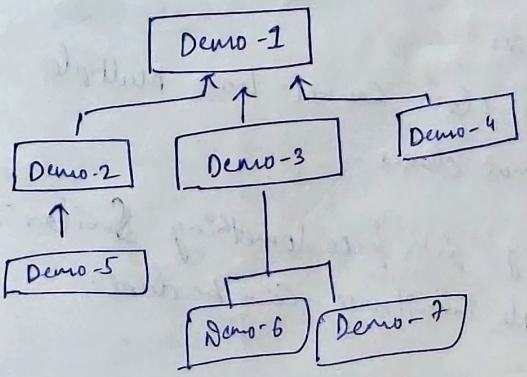
→ Properties and behaviour of Demo12 and object class are inherited in Demo13.

④ one parent or Base Class Can have any no. of child / sub classes.

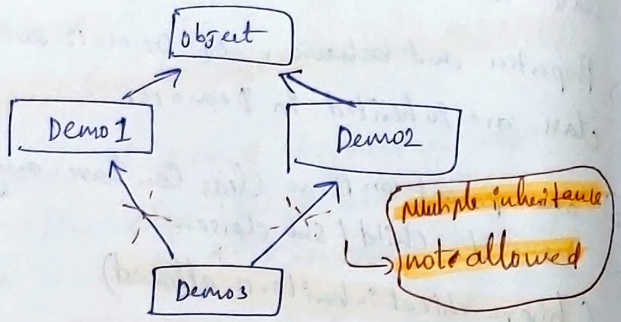
(hierarchical inheritance allowed)



⑤ Hybrid Inheritance :- Mixture of Multilevel and hierarchical inheritence is called hybrid inheritance



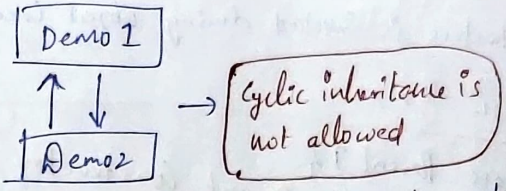
⑥ Multiple Inheritance is not allowed.



→ one child cannot have Multiple parents.
→ it Create ambigious for Compiler
→ ambigious → Compiler
↳ it also called diamond shaped problem.

→ At a time one class can Extends one class.
→ one class can have Multiple child classes.
→ one class cannot have Multiple Parent classes.
→ using interface something similar to Multiple inheritance can be done.

⑦ Cyclic inheritance is not allowed



⑧ Private Memebers of class doesn't participate inheritance. to preserve Encapsulation.
↳ direct access is

```
Class Parent1 {
    Private String name;
    Void disp1() {
        S.o.p("Parent 1");
    }
}
Class Child Extends Parent1
{
    Void disp2() {
        name = "Ashish";
    }
}
Public class inherit {
    Public static Void main (String[] args){
        Child c = new Child();
        c.disp2();
    }
}
```

→ it will not Participate in inheritance

→ we get Error here

① Constructors will not Participate Inheritance But Parent Class Constructor will get Called.

→ Constructors is invoked during object Creation.

Code:-

```
Class Parent1 {
    Parent1() {
        S.o.p("Parent Constructor");
    }
    Voiddisp();
    {
        S.o.P("Parent");
    }
}
Class child Extends parent1
{
    //child() {
    // Super();
    // }
    Void disp2() {
        S.o.P("child class");
    }
}
Public class Inherit {
    Public S. M V (Sty[] args) {
        child c = new child();
        C.disp();
    }
}
```

① We are Calling child Class Constructor.

② behind Seceen. JVM includes child Constructor and Super Method in that Class

③ Super() Method Class parents Constructor.
↳ body of parent Constructor get Executed

④ then again Control goes to Super() Method.

⑤ After that block of code after child Constructor get Executed.

Conclusion :-

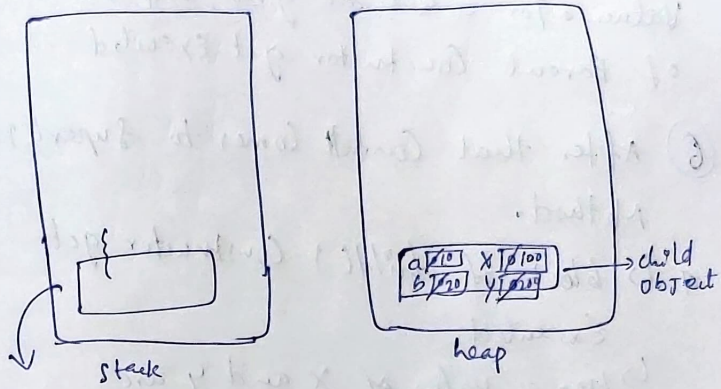
Constructor will not participate in inheritance But parent class Constructor will get Called by Super() Method which is inside the child class.

Constructor Execution in case of inheritance

```
Class Parent {
  int a, b;
  Parent() {
    a = 10;
    b = 20;
    S.o.p("Parent Constructor");
  }
  Parent(int a, int b) {
    this.a = a;
    this.b = b;
    S.o.p("Parent Parameterized Constructor");
  }
Class Child Extends Parent
{
  int x, y;
  Child() {
    Super();
    x = 100;
    y = 200;
  }
  Child(int x, int y) {
    this.x = x;
    this.y = y;
  }
  Void disp() {
    S.o.p(a);
    S.o.p(b);
    S.o.p(x);
    S.o.p(y);
  }
}
Public class inheri1 {
  Public static Void main(String[] args) {
    Child c = new Child();
    c.disp();
  }
}
```

Super() Method is included behind the Seceans

O/p: Parent Constructor
10
20
100
200



① Main Method stack frame get Executed.

② Child object is Created.

③ Control goes to zero parameterized Constructor. Super() Method. Present in child() Constructor.

④ Super() Method calls parent class Constructor which zero parameterized.

→ In child class we have x, y, a & b Variables.

→ a and b are inherted from parent class. Memory for x, y, a & b are Created in heap area. with default Values.

(5) Control come to parent class Constructor Value of a & b are given and block of Parent Constructor get Executed.

(6) After that Control Comes to super() Method.
↳ block of child() Constructor get Executed.
↳ here value of x and y are given
Eg x=100, Y=200

(7) After that Control go to main Method where object is created

(8) c.disp() is Called. it get Executed

→ Creating object of Parent Equal to Creating object of Child.

Calling Parameterized Constructor.

```
Class Parent {
int a,b;
Parent(){
 a=10;
 b=10;
 S.o.p("Parent Constructor");
}
Parent(int a,int b){
  this.a=a;
  this.b=b;
  S.o.p("Parent Parameterized Constructor");
}
}
class child extends Parent {
int x,y;
child(){
  x=100;
  y=200;
}
child(int x,int y){
super(x,y);
this.x=x;
this.y=y;
}
void disp(){
S.o.p(a); S.o.p(b); S.o.p(x); S.o.p(y);
}
Public class inherit {
    Public static v m(String[] args){
    child c = new child();
    c.disp();
    child c1 = new child(1000,2000);
    c1.disp();
}}
```

O/P :-
Parent Constructor
10
20
100
200
Parent Parameterized Constructor
1000
2000
1000
2000

Calls disp() Method

here super(x,y) Method Called Explicitly