(18) 29/09/24

In Heritance

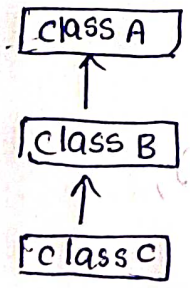
Inheritance means creating new classes based on existing ones. Inheritance in java is a key features of object-oriented programming that allows one class to inherit the properties (fields) and behaviours (methods) of another class.

Types of inheritance:

1. Single inheritance: A class inherits from one supper class

Class A → class B
(parent) (child)

2. Multilevel inheritance: A class is derived from a class which is also derived from another class.

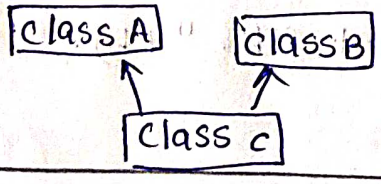


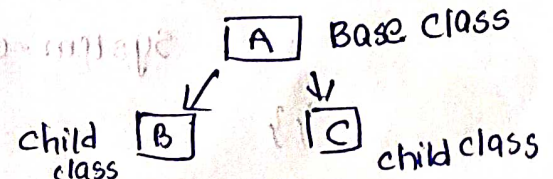
Superclass ex:- parent

subclass 1 ex: child

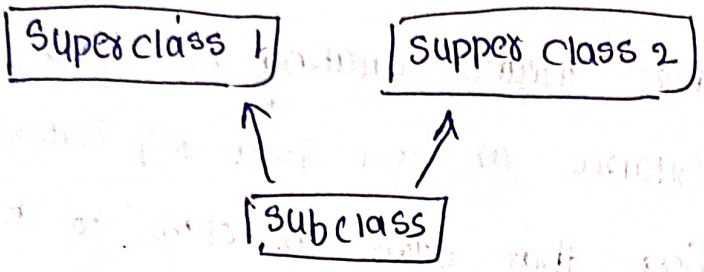
Subclass 2 ex: grandchild

3. Hierarchial inheritance :- Multiple classes inherit from a single Super class

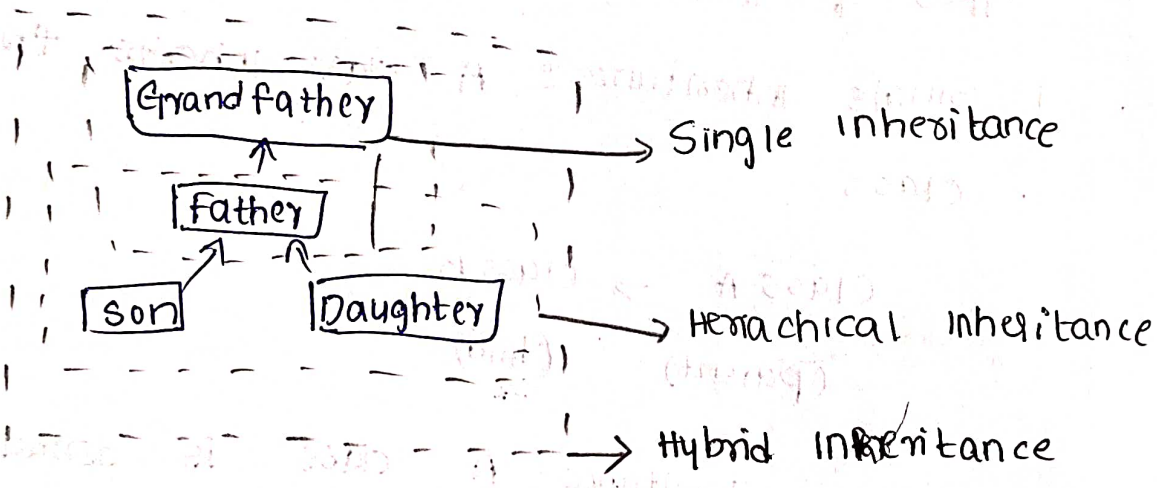




4. Multiple Inheritance: A scenario where a class inherit properties and methods from more than one sup



5. Hybrid Inheritance; It is a combination of two or more type of inheritance.



Single Inheritance:

```
class A {
    int a;
    System.out.println("a = "+a);
}}
class B extends A {
    int b;
    void display B O {
        System.out.print("b= "+b);
    }}
```

```
public class single inheritance example {
    public static void main (string [] args) {
        B.obj = new ();
        obj.a = 20;
        obj.b = 30;
        obj: display A ();
        obj: display B ();
    }}
```

output: a=20 , b=30

Multilevel inheritance.

```
class A {
    public void display A () {
        System.out.println ("Inside display A");
    }}
    Class B extends A {
        public void display B () {
            system. out. println (" inside display B");
    }}
        class C extends B {
            Public void display B() {
            system. out. println ("Inside display c");
    }}
        public class main {
            public static void main (string [] args) {
                c .obj = new C ();
                obj. display A();
```

```
Obj.displayB();
Obj.displayC();
}}
```

Output: inside display A
inside display B
inside display C

Hierarchial Inheritance

```
class Animal {
    void eat() {
        System.out.println("This animal eats food");
    }}
class Dog extends Animal {
    void barker {
        System.out.println("The dog barks");
    }}
class cat extends Animal {
    void meow() {
        system.out.println("The cat meows");
    }}
Public class Main {
    Public static void main(sting[] args) {
        Dog dog = new Dog();
        dog.eat();
```

```
dog.bark();
Cat cat = new Cat();
cat.eat();
cat.meow();
}}
```

Output: This animal eats food
The dog barks
This animals eats food
The cat meows

Multiple Inheritance

```
Class A {
    void method A () {
    System.out.println ("method from class A");
}}
    interface B {
        void method B ();
    }
    interface C {
        void method c ();
    }
    class D extends A. Implements B, C {
            public void method B() {
```

```
    System.out.println("method from interface B");
}
public void method(){
    System.out.println("method from interface c");
}}
public class multipleInheritance example{
    public static void main(String[] args){
        ID obj = new ID();
        Obj.method A();
        Obj.method B();
        Obj.method C();
    }}
```

output:

Method from class A
Method from Interface B
method From Interface C

Hybrid Inheritance

```
class Grandfather {
    public void show G() {
        System.out.println("He is grandfather.");
    }}
class father extends Grandfather{
    public void ShowF() {
        system.out.println("He is father.");
    }}
```

```
class son extends father {
    public void showS() {
    System.out.println("He is son");
}}
public class Daughter extends father {
   public void showD(){
   System.out.println("she is daughter");
}
   public static void main(String args[]) {
       son obj = new son();
         obj.showS();
         obj.showF();
         obj.showG();
       Daughter obj2 = new Daugther();
          obj2.showD();
          obj2.showF();
          obj2.showG();
}}
```

Output:

```
He is son
He is father
He is grandfather
She is daughter
He is father
He is grand father.
```

2. Exception Handling

Exception is an error that occurs during the execution of program.

Key component of exception handling.

1. Try Block - This is where you write the code that may throw an exception. If an exception occurs, the execution of the try block stops and control is transfered to catch block.

2. Catch Block - this is where you handle the exception Block of code to be executed if an error occurs to try block.

3. Finally Block - This block contains code that is executed regardless of whether an exception was thrown or not.

4. throw statements - This is used to explicitly thrown an exception from a method, or block of code

1. Try - - catch block

```
class main {
        public static void main (string [] args) {
try {
    int a = 5/a;
    system.out.println ("Res of code in try block");
}
catch (Arithmetic exception e) {
    System.out.printen ("Arithmetic exception =>" +
                        e.ger message());
```

```
} catch (exception e) {
        System.out.println ("exception => " + e.getmessage());
}}}
```

Output: Arithmetic exception => / by zero

2. Try --- catch block

```
public class main {
    public static void main (String[] args) {
try {
    int a[] = {10, 20, 30};
    system.out.println (a[10]);
} catch (ArrayIndex OutofBounds exception e)
{ system.out.println ("Array index out of Bound
                    exception => " + e.getmessage());
}}}
```

Output: Array indexout of Bounds exception => index is out of bounds for length 3.

Finally block

```
class main {
    public static void main (String[] args) {
```

```
int divide By zero = 5/0;
}
catch [Arithmetic Exception e) {
        System.out.println ("Arithmetic exception =>" + e.g
    message ());
}
Finally {
    system.out.println ("This is the finally block").
}}}
```

Output:

Arithmetic exception => / by zero
This is Finally block:

Throw (vote)

```
Public class main {
        static void checkage(int age) throws Arithemetic
exception
    {
    if (age < 18) {
throw new Arithmetic exception ("you are not eligible").
}
else {
system.out.println(" you are eligible to vote");
}}
public static void main (String [] args {
                checkage (15);
    }}
```

Output: You are not eligible.

Nested try block

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

try {
  int a[]= {1,2,3};

try {
   int b=1/0;

 } catch (exception e){
   System.out.println (" Exception thrown :" + get message
     age(); }

     system.out.println(a[4]);
   }
   catch [Array - index out of bounds exception e) {
     System.out.println(" exception thrown :" +e. get message()).

     }
     system.out.println ("out of the block");

  }
}
```

Outputs: Exception thrown : / by zero

exception thrown: index 4 out of bounds for length 3

out of the block