CSA0985 - Java programming P. Reddy renuka

ASSIGNMENT - 2 (3) 192311062

Inheritance:

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

Types of inheritance:

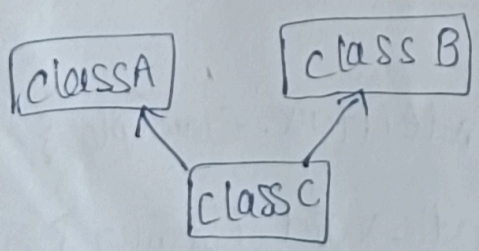
1. **Single inheritance:** A class inherits from one super class

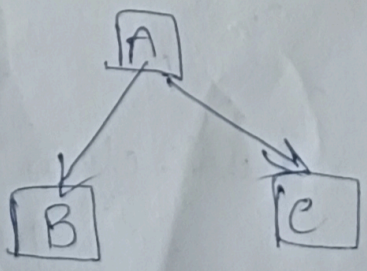
class A → class B
(parent) (child)

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

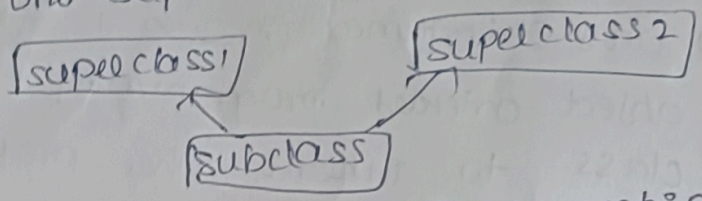
class A	superclass	Ex: parent
class B	Subclass1	Ex: child
class C	subclass2	Ex: grandchild

3. **Hierarchial Inheritance:** multiple classes inherit from a single superclass.

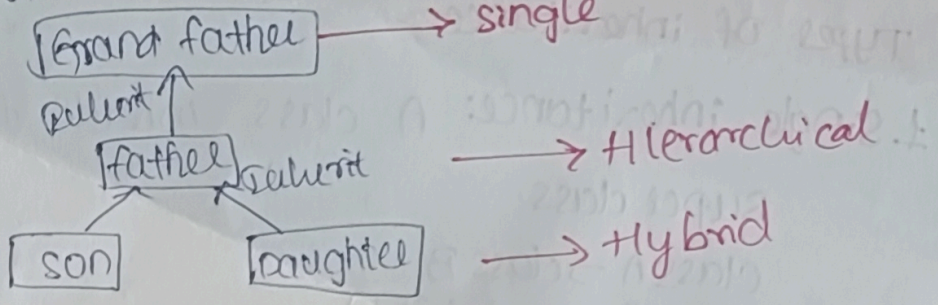




4. Multiple Inheritance: A scenario where a class can inherit properties and methods from more than one superclass



5. Hybrid Inheritance: It is a combination of two or more types of inheritance.



single inheritance:

```
class A {
    int a;
    void displayA(){
        system.out.println("a="+a);
    }
}
public class B extends A{
        int b;
        void display B(){
            system.out.println("b="+b);
        }
}
public class single inheritance example {
        public static void (String [] args){
```

output:

```
This animal eats food
The dog barks
This animal eats food
The cat meows.
```

Multiple Inheritance:

```
class A {
    void methodA() {
        System.out.println("Method from class A");
    }
}
class B extends A {
    public void displayB() {
        System.out.println("Inside display B");
    }
}
class C extends B {
    public void displayB() {
        System.out.println("Inside display C");
    }
}
public class main {
    public static void main(String[] args) {
        C obj = new C();
        obj.displayA();
        obj.displayB();
        obj.displayC();
    }
}
```

output:

```
Method from classA
method from interface B
method from interface C
```

Hybrid inheritance:

```
class Grandfather {
    public void showG(){
    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 static void main (String args[]) {
    Son obj = new Son();
    obj.showS();
    obj.showF();
    obj.showG();
    Daughter obj2 = new Daughter();
    obj2.showD();
    obj2.showF();
    obj2.showG();
}}
```

Multilevel Inheritance:

output:

He is son
He is father
He is grandfather
She is daughter
He is father
He is grandfather

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();
}
}
```

2. Exception handling:

Exception is an error that occurs during the execution of program

Key components of exception handling.

1. Try Block - This is where you write the code that might throw an exception. If an exception occurs, the execution of the try block stops and control is transferred 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 why thrown or not
4. Throw statement: 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= 5/0;
    System.out.println("Res or code in try block");
  }
  catch (Arithmetic exception e){
    System.outprintln (" Arithmetic exception =>"+e.
                                    get 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.print(n[a[10]);
}
catch(Array Index out of Bounds exception e)
{
system.out.println("Array Index out of Bound exception)"
                                +e.getMessage());
}
}
}
```

output: Array Index out of BoundsException => Index 10 out of Bounds for length 3.

Finally block:

```
class main {
  public static void main(String[] args) {
  try {
    int divide by zero = 5/0;

  }
   catch (Arithmetic Exception e) {
   System.out.println ("Arithmetic exception => " + e.get
                                                message());
  }
  finally {
   System.out.println ("This is the finally block");
}
}
}
```

output:

Arithmetic exception => / by zero
this is the finally block

Throw (note):

```
public class main {
  static void checkAge (int age) throws Arithmetic
   exception
  {
  if (age < 18) {
```

```
throw new ArithmeticException("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:"+
                    e.getMessage());
}
system.out.println(a[4]);
}
catch(arrayIndexOutOfBoundsException e){
system.out.println("exception thrown:"+e.get
                                    Message());
}
```

```
        System.out.println("Out of the block");
    }
}
```

output: exception thrown : 1 by zero
Exception thrown : Index 4 out of bounds for length 3 out of the block.