(19)

29/07/2024

ASSIGNMENT

NAME :- N.VISHNU SAI SRI

REG.NO.:- 192372039

COURSE CODE:- CSA0985

COURSE :- JAVA PROGRAMMING FOR WEB APPLICATIONS

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 derived from a class, which is also derived from another class.

Class A	Superclass	ex:- parent
Class B	subclass 1	ex:- child.
class C	sub class 2.	ex:- grandchild.

(3) Hierarchical inheritance: Multiple classes inherit from single superclass.

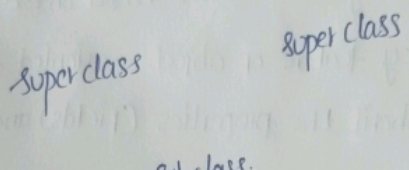
Class A Class B

A

B C

Class C.

(4) Multiple inheritance: A scenario where a class can inherit properties and methods from more than one super class.



(5) Hybrid inheritance: it is a combination of two or more inheritance.

Grand father. ⟶ single inheritance

Father. ⟶ hirearchial inheritance

Son, daughter. ⟶ hybrid inheritance

Single inheritance:

```
class A {
    int a;
    void display () {
        System.out.println ("a = "+a);
    }
}

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

```
public class main{
 public static void main (String [] 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 animal 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 B 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 multiple inheritance example {
    public static void main (String[] args) {
        D obj = new D();
        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 father");
}
}
```

```
public class Singleinheritance example {
        public static void main (String[] args) {
            B obj = new B();
              obj.a = 20;
              obj.b = 30;
             obj.displayA();
             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 . display B();
    obj . display C ();
  }
}
```

output :- inside display A
inside display B
inside display C.

Heirarchial Inheritance:

```
class Animal {
  Void eat () {
  System . out . println ("this animal eats food");
 }
}
class dog extends Animal {
  void bark () {
  System . out . println ("the dog barks");
 }
}
class cat extends Animal {
   Void meow () {
  System . out . println ("the cat meaows");
 }
}
```

(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 way thrown or not.

(4) Throw Statement → this is used to explicity 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 of code in try block");
}
catch (Arithmetic exception e) {
System.out.println ("Arithmetic exception → " + e.getmessage());
```

```
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 daughter ();
        obj2.showD();
        obj2.showF();
        obj2.showG();
    }
}
```

output:-

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

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

output:-
Arithmetic exception => by zero.

(2) try ... catch Block.

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

output :- Array index out of Bounds exception => 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.getMessage());
    }
    finally {
        System.out.println("This is the finally block");
    }
  }
}
```

output:-

Arithmetic exception => / by zero

this is the finally block.

throw (vote).

```
public class main {
static void checkAge (int age) throws Arithmetic
 Exception
 {
  if(age <18) {
throw new Arithemtic Exception ("you are not eligible");
  }
  else {
  System.out.println ("you are eligle to vote).
 }
}
```

output:- you are not eligible.

Nested by blocks:

```
public class exception {
  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 (Array indexout of Bounds Exception e) {
   System.out.println ("exception thrown : "+e get message ());
  }
  System.out println ("out of the block");
 }
}
```

output:-

exception thrown : /by zero

Exception thrown : index 4 out of bounds 4 ands For length 3 out of the block.