CSA0985 - Java Programming

ASSIGNMENT-2

M. Soumya
192372064

1. 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.

Type of Inheritance

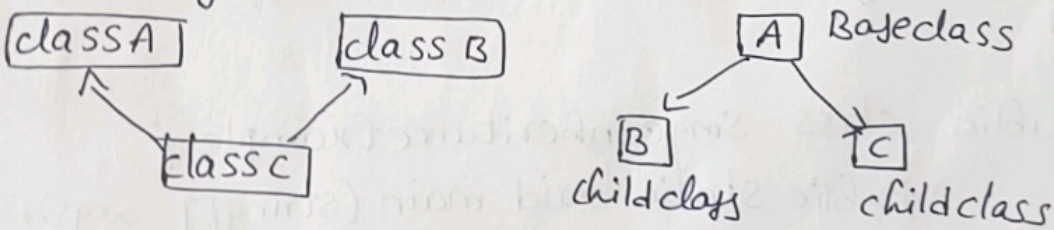
1. single Inheritance : A class inherits from one superclass

   class A → class B
   (parent) (child)

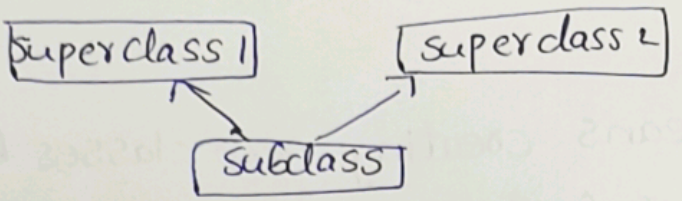
2. Multilevel inheritance: A class is derived from a class, which is also derived from another class.

   classA superclass Ex:- parent
   ↑
   classB subclass 1 Ex:- child.
   ↑
   classC subclass 2 Ex:- grandchild

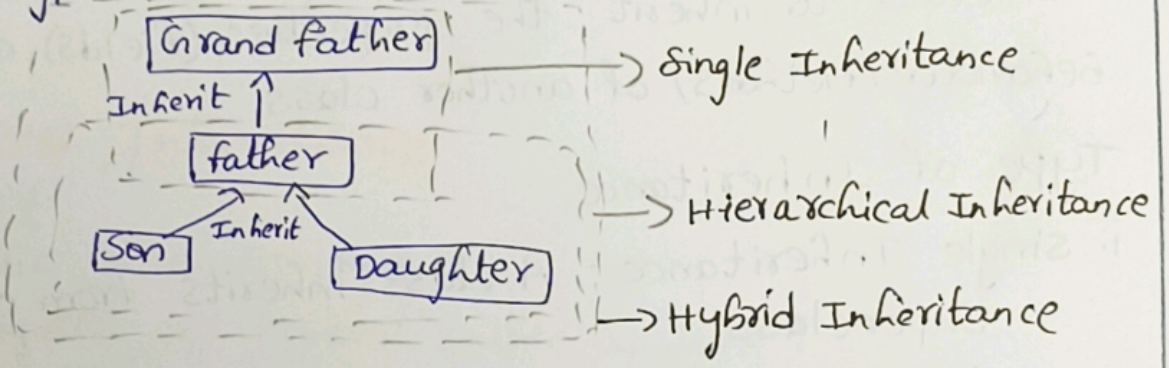
3. Hierarchial Inheritance:- Multiple classes inherit from a single superclass.



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);
    }
}
class B extends A {
    int b;
    void displayB() {
        system.out.println("b = "+b);
    }
}
Public class SingleInheritanceExample {
    Public static void main(String[] args){
```

```
        B obj = new B();
        obj.a = 20;
        obj.b = 30;
        obj.displayA();
        obj.displayB();
    }
}
```

output:- a = 20, b = 30

Multilevel Inheritance

```
class A {
    Public void displayA() {
        system.out.println("Inside displayA");
    }
}
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 :- 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 meows");
  }
}
  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 methodA() {
        System.out.println("Method from class A");
    }
}
Interface B {
    void methodB();
}
Interface c {
    void method c();
}
class D extends A implements B, C {
    Public void methodBC() {
        System.out.println("Method from Interface B");
    }
    Public void methodC() {
        System.out.println("Method from Interface c");
    }
}
Public class MultipleInheritanceExample {
    Public static Void main(string[] args) {
        D obj = new D();
        obj.methodAC();
        obj.methodBC();
        obj.method C();
    }
}
```

output:-

```
Method from class A
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 show s() {
       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();
```

```
        obj 2. showD();
        obj 2. showF();
        obj 2. 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 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());
  }
  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 (ArrayIndexOutOfBoundsException e)
{
  system. out. println ("ArrayIndexOutOfBoundException => "
                                  + e. getMessage());
  }
}
}
```

output:- ArrayIndexOutOfBoundsException => Index 10 out of bounds for length 3.

Finally block

```
class Main {
  Public static void main (string[] args) {
try {
   int divideByZero = 5/0;
}
catch (ArithmeticException e) {
  System.out.println ("ArithmeticException =>"
                          +e.getMessage());
}
  finally {
   System.out.println("This is the finally block");
  }
}
}
```

output:-

```
ArithmeticException => /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 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:"+e.getMessage());
}
system.out.println(a[4]);
}
catch (ArrayIndexOut of Bounds Exception e) {
     System.out.println("Exception thrown:"+e.getMessage());
}
system.out.println("out of the block");
}
}
```

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