SA0985 - Java Programming. (b) K.Meghan
19232112

29/03/24

Assignment - 02

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.

Types of Inheritance.

1. Single Inheritance: A class inherits from one super class

$$\begin{tikzcd}\text{Class A (Parent)} \arrow[r] & \text{class B (child)}\end{tikzcd}$$

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

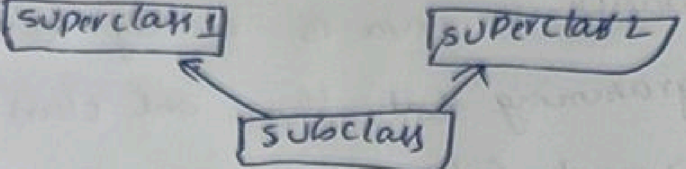
$$\begin{tikzcd}\boxed{\text{Class A}} & \text{super class} & \text{Ex:- Parent} \\ \boxed{\text{Class B}} \arrow[u] & \text{subclass1} & \text{Ex: child} \\ \boxed{\text{Class C}} \arrow[u] & \text{subclass2} & \text{Ex: grandchild}\end{tikzcd}$$

3. Hierarchical Inheritance:- Multiple classes inherit from single superclass

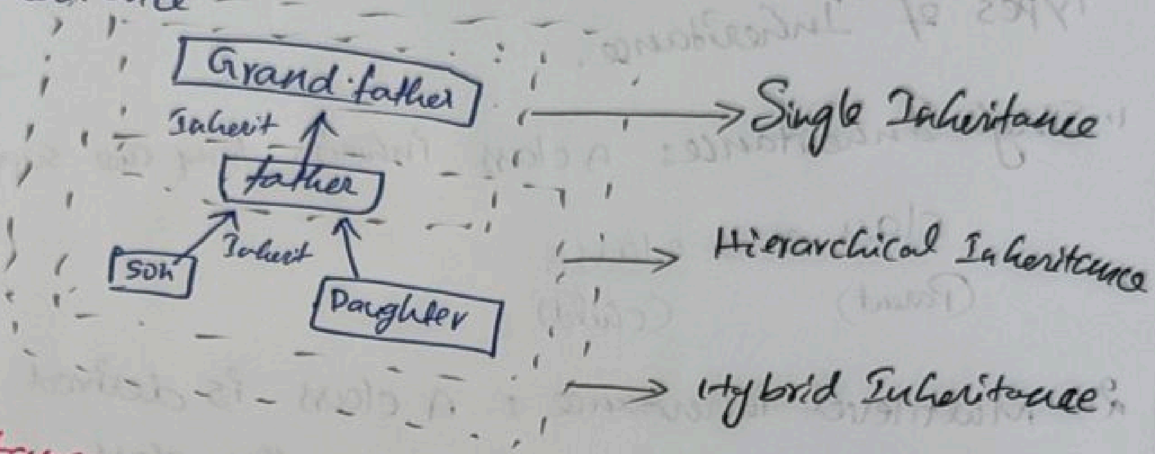
$$\begin{tikzcd}\boxed{\text{Class A}} & & \boxed{\text{Class B}} \\ & \boxed{\text{class C}} \arrow[ul] \arrow[ur] &\end{tikzcd}$$

$$\begin{tikzcd} & \boxed{A} \text{ Base class} \arrow[dl] \arrow[dr] & \\ \boxed{B} \text{ child class} & & \boxed{C} \text{ child class.}\end{tikzcd}$$

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 types of inheritance



Single Inheritance

```
Class A{
  int a;
void displayA(){
    system.out.Println("a="+a);
}
}
Class B extends A{
    int b;
void display B(){
   system.out.println("b="+b);
}
}
Public class single inheritance Example {
   Public static Void main (string[] args){
```

tput
```
Inside displayA
Inside displayB
Inside displayC
```

Heirarchical 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();
 }
}

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 display A.(){
   system.out. Println("Inside display A");
 }
}
 class B  extends A{
   Public void displayB(){
      system.out. Println ("Inside display B");
 }
}
 class c extends B{
    Public void display c (){
    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

```
  Method from class A
  Method from interface B
  Method from interface c.
```

Hybrid Inheritance.

```
class Grandfather{
  Public Void shown(){
 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 show D(){
    System.out. Println("she is daughter");
 }

   Public static void main(string args[]){
           son obj = new son();
           Obj shows();
           Obj .showF();
           Obj. showG();
           Daughter obj 2 = new Daughter();
```

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 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 C (){
    System.out. Println ("Method from Interface C");
}
}
Public class Multiple InHeritance Example{
    Public static void main (String[] args){
        D obj =new D();
        obj. methodA();
        obj. method B();
}       obj. method C();
}
obj2. show D();
Obj2. show F();
Obj 2-show G();
}
```

Output

```
He is son
He is father
He is grandfather
she is daugther
He is father
He is grand father
```

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 is block contains code that is execu regardless of whether an exception was 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.getmessa
}
catch (Exeeption e) {
}   system.out.println(" exeeption => "+e.getMessage());
```

utput := Arithmetic Exeeption => 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 (Array Index Out of Bounds Exception e)
  {
   system. out. println ("Array Index Out of Bound Exception =>" + e.
                                              getMessage());
  }
 }
}
```

Output
```
Array Index Out of Bounds Exception => Index 10 out of bound
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 classmain{
 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){
    check Age (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 (Array Index out 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 for length 3 out of the block.