UNIT - IV

① What is meant by Inheritance and list its advantages? Explain the various types of inheritance use in java with suitable example.

⇒ Inheritance :

- It is a mechanism in which derived class property inherits the property of base class and at the same time the derived class may have some additional properties.
- This facilitates code reuse and establishes a natural heierarchy between classes.

Advantages :

(i) Reusability : The base class code can be used by derived class without any need to rewrite the code.

(ii) Extensibility : The base class logic can be extended in the derived classes.

(iii) overiding :

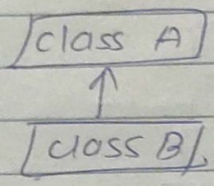
methods of baseclass can be override so that meaningful implementation of the base class method can be designed in the derived class.

(iv) Data hiding :-

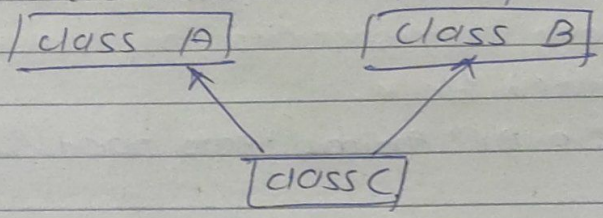
Base class can keep some data private so that it cannot be altered by derived class.

Types of Inheritance:

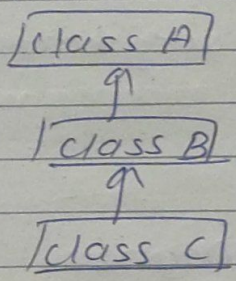
① Single Inheritance:
Only a single class is inherited by derived class. Single parent-child relationship is present



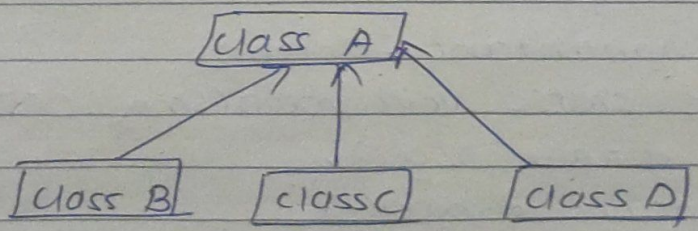
② Multiple Inheritance:
One class extends more than one class, which means class has two parent classes.



③ Multilevel Inheritance:
When a derived class is derived from base class which itself is a derived class.

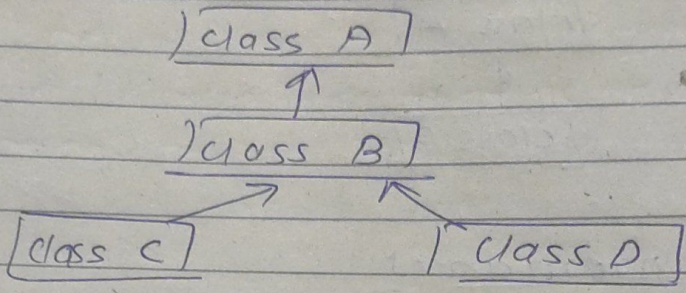


④ hierarchical Inheritance:
More than one class extends the same class.



⑤ Hybrid Inheritance:

combination of more than one types of inheritance in a single program.



eg.

```
class Animal {
        void eat() {
                S.O.P(" Animal eats food.");
        }
}

class Dog extends Animal {
        void bark() {
                S.O.P(" dog barks");
        }
}

class BabyDog extends Dog {
        void weep() {
                S.O.P(" Baby dog weeps");
        }
}

public class Inheritance_Demo {
        public static void main(String S[]) {
        Dog d = new Dog();
        d.eat(); d.bark();
        BabyDog db = new BabyDog();
        db.eat();
        db.bark();
        db.weep();
        }
}
```

② Describe exception. Write any two examples of exceptions. Explain keywords try, catch, throw, throws and finally related to exception handling.

⇒ Exception:-

- It is an indication of some unusual event. Usually it indicates the error.
- In Java exception is handled using five keywords try, catch, throw, throws and finally.
- The Java code that you may think may product exception is ~~done in ord~~ placed within try block.

eg. Divide by zero:

```
class ExceptionDemo {
    public static void main (String s[]) {
        try {
            int a, b;
            a=5;
            b=a/0;
        }

        catch (Exception e) {
            S.O.P(" Divide by zero");
        }
    }
}
```

Various keywords used in handling the exceptions are :-

(i) try :-
A block of source code that is to be monitored for exception.

(ii) catch :
The catch block handles the specific type of exception along with the try block. Note that for each corresponding try block there exists the catch block.

(iii) finally :-
It specifies the code that must be executed even though exception may or may not occur

(iv) throw :-
This keyword is used to throw specific exception from the program code.

(v) throws :-
It specifies the exceptions that can be thrown by a particular method.

③ Define package used in Java. Explain use, syntax, CLASSPATH, hierarchy of package with example.

⇒ Use Package:

To use a class or interface from a package, you can either import the whole package or the specific class/interface.

```
import mypkg.myclass;
or
import mypkg.*;
```

Syntax:

To create a package, you use the 'package' keyword at the beginning of our java source file followed by package name

```
package mypkg;
```

CLASSPATH:

```
> set CLASSPATH = ;D:\;
> cd my_Package
> My_package > javac A.java
> my-Package > javac PackageDemo.java
```

It is an environment variable that tells the JVM and Java compiler where to look for user-defined classes and packages.

Hierarchy of packages:

Java allows you to create a hierarchy of packages, ie. packages with packages. This helps in organizing classes better.

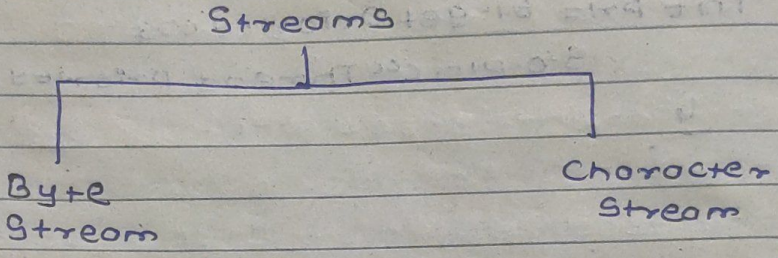
eg

```
src/
    com/
        example/
            myApp/
                main.java
            utilities/
                Helper.java
```

5] What is concept of stream explain byte stream and character stream in detail in java?

→

- Stream are fundamental for reading or writting to data source such as file, network connection.
- Stream can be unidirectional or bidirectional it can also be synchronous or asynchronous.



① Byte Stream :-

- Byte stream represented by class java.io package, are used for handling raw binary data.
- They read and write data in form of bytes.
- It is suitable for reading and writting binary files, such as image, audio or video files.
- Byte stream class typically end with name InputStream or OutputStream.

- Some common byte stream class include :-

① File InputStream
② File OutPut Stream
③ Data InPut Stream
④ Data OutPut Stream

* Example of reading from file using byte stream :-

```
try {FileInputStream fis = new FileInput
        Stream ("example.txt");}{
    int byteread;

    while (byteread = fis.read() != -1){
        s.o.p ((char) byteread); // convert
                                    byte to char
    }                               and print
}
catch (IOException e){
    e.printStackTrace();
}
```

② Character Stream :-

- Character Stream is also represented by class in java.io package and used for handling textual data.
- They read and write data in form of character.
- Character Stream automatically handle encoding and decoding of character making them suitable for reading and writting text file

~~*Example~~ :-

- Character Stream classes typically have names ending with "Reader" or "Writter".
- Some of classes of character stream class include:-
  1. FileReader
  2. BufferredReader

6] What is abstract class and poly morphism in java with example?

→

* Abstract class:-

- In java an abstract class is a class that cannot be instantiated on its own and is typically used as blue print for other classes to inherit from.
- Abstract class may contain abstract method which are method without body and concrete method which have body.

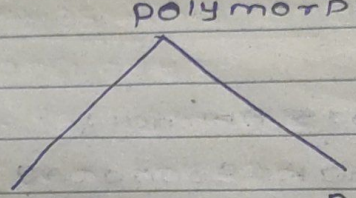
* example:-

```
abstract class Animal{
    abstract void makeSound();

    //concrete method
    void sleep(){
        S.o.println("sleep");
    }
}

class dog extends Animal{
    void makeSound(){
        S.o.println("woof");
    }
}

Public class main{
    Psvm (String args[]){
        dog d = new dog();
        d.makeSound(); // abstract
        d.sleep(); // concreate
    }
}
```

- In this example Animal is abstract class which has abstract method make sound.
- Dog is subclass of Animal and Provide their own implementation of makeSound()

*Polymorphism:-

- Polymorphism consist of two greek words 1st word :- poly and 2nd word :- morphism poly means many and morphism means forms
- Polymorphism refers to ability of single method or piece of code that can operate on different type of data.



Compile time

↓

When there are multiple function with same name ~~called~~ but different parameter is called method overloading

Run time

↓

When derived class has a defination for one of member function of base class then it is overriding.

eg:-

```
class Animal{
    void makeSound(){
        S.O.println("some sound");
    }
}
class dog extends Animal{
    void makeSound(){
        S.O.println("WOOF");
    }
}
```

```
class cat extends Animal{
        System.out.println("meow");
    }
}

Public class main{
        psvm(String args[]){
        ~~Animal a1 = new Animal();~~
        Animal a1 = new dog();
        Animal a2 = new cat();

            a1.makeSound(); // woof
            a2.makeSound(); //meow
    }
}
```

PAGE NO.
DATE:

Q. State the importance of Finally blocks. Illustrate the ways Finally block differ from finalize() method.

→

Finally block :

- This block will be executed after try/catch block.

- Finally block of code always executes.

- We can use finally where we need to close file handling, close SQL statements, close network connections.

Importance of finally block :

1. Guaranteed Execution:

The code within a Finally block is always executed after try block, regardless whether exception thrown or handled.

- This is crucial for cleaning resources like file handles, database connections.

2. Resource management:

Ensuring resources are properly closed or released is critical in preventing resource leaks.

3. Exception Handling:

complement exception handling by allowing cleanup actions that need to occur.

4. Consistent state:

Using a finally block helps to maintain the program's consistent state by ensuring necessary cleanup operation are performed.

Difference betn Finally block & finalize method.

1. Purpose & Usuage:

- Finally block used for cleanup operation that should be executed after ~~try~~/catch block.

- finalize()method is called by garbage collector before an object is destroyed. It is used for cleanup purpose.

2. Execution Timing:

- Finally blocks gets executed immediately after try/catch block.

- finalize() method is called by garbage collector at an undetermined time.

PAGE NO.
DATE:

3. Control:

. You can have explicit control over finally block.

. You have no control over finalize() method.

4. Resource management:

. Finally block should be used for closing or releasing resources like files, sockets, & database connections.

. Finalize method is not recommended for resource management due to its unpredictable execution.

Q. Write a program to create interface A in a package; in this interface we have 2 methods meth1 & meth2. Implements this interface in another class named myclass by importing your package.

→

Filename: mypackage/A.java

```
package mypackage;

Public interface Ad
        void meth1();
        void meth2();
}
```

Filename: myClass.java

```
import mypackage.A;

public class myClass implements A {
    public void meth1() {
        sop(" meth 1 ");
    }
    public void meth2() {
        sop(" meth2 ");
    }
    public static void main(String [] args) {

        myclass obj = new myClass();
        obj.meth1();
        obj.meth2();
    }
}
```

```java
//Create a custom Exception class. You need to consider two integer inputs that the user must
//supply, You will display the sum of the integers if and only if the sum is less than 100. If it is not less
//than 100, throw your custom exception.

import java.util.*;
class Demo{
	public static void main(String [] args){
		Scanner sc = new Scanner(System.in);
		try {
			System.out.print("Enter 1st number : ");
			int num1=sc.nextInt();
			System.out.print("Enter 2nd number : ");
			int num2=sc.nextInt();

			int sum=num1+num2;
			if(sum>=100){
				throw new error("Sum is to large");
			}else{
				System.out.println("Sum of two numbers is : "+sum);
			}
		}

		catch(error e){
			System.out.println(e.getMessage());
		}
	}

}
class error extends Exception {
	public error(String msg) {
		super(msg);
	}
}
```

* Elaborate the significance of keyword "super" in java demonstrate with simple and short example for super keyword in java constructor.

→

* super keyword :
  - In java the keyboard 'super' is used to refer to the superclass (parent class) of current object It is often used in the following senarios.

    1. TO call superclass constructor.

    2. To access a field or method of superclass that has been overriden in the subclass

* significan of "super" in constructor.

  - when a subclass is inherits from a superclass the constructor of the superclass need's to be called to initialize the superclass ~~has no argument constructor~~. part of the object.
  - If superclass has no-argument constructor java automatically insert a call to "super()" if you don't specify it.
  - Hoever if the the superclass has a parameterized constructor you need to explicitly call "super()" with the appropriate argument.

* Example :-

```
class Animal {
    string name;

    Animal (string name) {
        this.name = name;
        System.out.println ("Animal constructor
                            is created");
    }
}
class Dog extends Animal {
    int age;

    Dog ( string name, int age) {

        super (name);
        this.age = age;
        System.out.println ("Dog constructor
                            is called");
    }

void display () {

    system.out.println ("Name :" + name +
                        "Age :" + age);
    }
}

public static void main (string[] args) {
    Dog mydog = new Dog ("Buddy", 3);
    mydog.display();
    }
}
```

Page No.
Date

* Explain Package and interface in Java & with suitable example ?

* ~~Pakage in~~
* Package in Java :-
  - A package in Java is a way to group related classes, interfaces and sub packages.
  - It helps in organizing your code and avoiding name conflicts.

* Example :
  1. Create a Package
     - Define package at the top of your Java source file using the 'package' keyword.

~~#~~ code :

```
package com.example.myapp ;

public class Myclass {
    public void display () {
        System.out.println ("hello,
        from Myclass from com.
        example.myapp Package");
    }
}.
```

2. using created package.
   - we can used package by simple importing it.

```
import package com.example.myapp;

    public class Main {
        public static void main (String[] args){
            Myclass obj = new Myclass();
            obj.display()
        }
    }
```

* Interface In Java :-
- An interface in Java is a ~~reference type~~ like a blueprint of a class It defines a set of method that a class must implement.
- It's a way to achive abstraction and define comonon behaviour that multiple classes can share.

* Example :

```
interface a {
        public void myfun();
}
interface b {
        public void myfun();
}

class Myclass implements x,y
{
    public void myfun() {
        System.out.println ("More than
                1 interface implement");
}
```

```
public static void main (String args[])
{
    Myclass obj = new Myclass();
    obj.myfun();
}
}
```

* State difference between character and byte stream in Java Give 2 input & 2 output classes for character stream.

character stream	Byte stream.
1. operate with unicode character	1. operate with raw Binary data.
2. It handle charater oriented data such as text files	2. Handing any type of file including non-text data.
3. They are typically use classes that end with 'Reader' and 'writer'.	3. They are typically use classe that end with 'InputStream' and 'outputStream'.
4. It can handle various symbols in language	4. It can handle file like including images, audio, non-text data.
5. classes - FileReader - BufferedReder	5. classes - FileInputstream - Buffered Input Strem

* classes of character stream :-
  * Two input classes for character Strem.
    1. File Reader :
       - Read character file.

2. Buffered Reader :
- Read text from a character-input stream.
- It efficient Read character from array.

* 2 Two output classes for character streams:
1. Filewriter
- writes character files

2. Bufferwritter
- writes text to a character-output stream.
- efficienty write a character.

Q.13 Explain the concept of dynamic dispatch while overriding method in inheritance. Give example and advantages of doing so.

☞ [SPPU : Dec.-17, Marks 5]

Ans. : • The dynamic method dispatch is also called as **runtime polymorphism**.

- During the run time polymorphism, a call to overridden method is resolved at run time.
- The overridden method is called using the reference variable of a super class(or base class). The determination of which method to invoke is done using the object being referred by the reference variable.

Following is a simple Java program that illustrates the Run time polymorphism.

```
class Base
{
 void display()
 {
    System.out.println("\n Base Method Called");
 }
}
class Derived extends Base
{
  void display() //overridden method
```

```
{
    System.out.println("\n Derived Method Called");
}
}
public class RunPolyDemo
{
    public static void main(String args[])
 {
  Base obj=new Derived();//obj is reference to base class
                         // which is referred by the derived class
  obj.display(); //method call which is determined at run time
 }
 }
```

Output

```
D:\test>javac RunPolyDemo.java

D:\test>java RunPolyDemo

 Derived Method Called
```