

Multiple Inheritance & Polymorphism:-

Abstract class :-

Abstract class के class होती हैं जिसके abstract class के बनाया जाता है। Abstract class के method abstract या non-abstract होते हैं। Abstract class के non-abstract method को implement करना चाहिए। Abstract class के non-abstract method को विकल्प नहीं होता। Abstract class के non-abstract method को बनायी हुई object द्वारा बनाया जाता है। Abstract class के non-abstract method को call करने के लिए उसे subclass में inherit करना पड़ता है। subclass के object के द्वारा उसी abstract class के method के द्वारा data members को call किया जाता है।

Important Points:-

- (1) abstract class के Abstract keyword के द्वारा बनाया जाता है।
- (2) Abstract class के object को भी create नहीं किया जाता है।
- (3) Abstract class के अन्दर abstract रूप से non-abstract जैसे class के method होते हैं।
- (4) Non-abstract class के subclass के द्वारा inherit किया जाता है। उसका द्वारा abstract method के body के implementation abstract class में न होते। Subclass में किया जाता है।
- (5) Abstract class के अन्दर method के static रूप से constructor बनाया जा सकता है।

Syntax:

abstract class name

 abstract data type method();

Body of method

?

Abstract Method :-

ये के method होते हैं जो abstract keyword के द्वारा बनाया जाता है तथा abstract method को generally abstract class के अन्दर ही बनाया जाता है। abstract method की body abstract class में implement नहीं की जाती है। इसकी body उस subclass में बनायी जाती है। परन्तु दूसरी abstract class को inherit किया जाएगा।

Important Point :

1 Abstract Method को हमेशा abstract class के भी होते हैं। इसका लाभ यह है Abstract keyword के साथ declare किया जाता है।

abstract keyword के द्वारा method के implementation को hide कर किया जाता है।

Syntax :

abstract class name

{

 abstract datatype method name();
}

Program :

```
abstract class ATM
```

```
{ abstract void credit();  
void test();
```

```
{ System.out.println ("Thankyou for using ATM");
```

```
}
```

```
class SBI extends ATM
```

```
{ void credit()
```

```
{ System.out.println (" Thank you for using SBI ");
```

```
class CBI extends ATM
```

```
{ void credit()
```

```
{ System.out.println ("Thank you for using CBI ");
```

```
public static void main (String args[])
```

```
{ SBI s = new SBI ();
```

```
s.credit();
```

```
CBI c = new CBI ();
```

```
c.credit();
```

```
c.test();
```

```
}
```

```
}
```

Output:

Interface : Interface class की रूप होने वाले लैंडिन फ़ार्मेंट में सबसे पहले वार्ड दिया जाता है। इसके बारे में विस्तृत विवरण निम्नलिखित बाब्ताएँ हैं:

- variable declare किया जाता है। यह By default + final होता है तथा यह method होती है और abstract होती है। Abstract method की method होती है जिनका implementation उस class या interface में नहीं दिया जाता है। यिसमें वो declare की गयी है।
- Interface को define करने के लिए interface keyword का use किया जाता है।

Syntax:-

Interface Interfacename

{

variable declaration;

{

method declaration()

{

}

उपर लिखे Syntax में Interface एक keyword है। Interface name Interface या Name हो सकता है। इसके अन्तर्गत variable static या final होते हैं। यह हम को सहने हैं कि variable constant होती है। तथा method abstract होती है।

Syntax:

static final type variable

Name = value;

Extending Interface :- class की रूप

Interface को नहीं extend किया जा सकता है। sub-class की रूप sub

- Instance Interface के द्वारा नहीं Superclass
- Interface की सभी properly या method को Inherit किया जा सकता है।

जबकि यह extends keyword का use करता है।

Syntax: Interface Interface.B extends Interface.A

Program: -

Interface Student

```
{  
    int RollNo = 100;  
    String Name = "Ram";  
    void show();
```

Interface Student1 extends Student

```
{  
    void display  
}
```

Implementing Interface: Interface की declare की गई method abstract होती है और उनमें Implement करने के लिए हिस्से class की परंपरात्मक ढंग से interface Interface की properties की class ने Inherit कर ली है और इसके implements का use करता है।

Syntax: classclassname implements Interface name
{
 body of classname
}

Program:

Class Extends Class	Interface Extends Interface	Interface Implements Class
---------------------------	-----------------------------------	----------------------------------

Rules of Interface :-

- i) Interface का object create नहीं करते हैं।
- ii) Interface का constructor नहीं होता है।
- iii) Interface का Interface का गति Extends होता है।
- iv) Interface का class ने उसका implements करता है।
- v) Interface का variable final वाले static एवं अति abstract होते हैं।

Program:-

Interface Student

{
 int RollNo = 100;

 String Name = "Ram";

 void Show();

}

Interface Student2 extends Student1

{

 void display();

}

class student implements Student2

{

 void Show()

{

 System.out.println("Roll No of Student: " +
 RollNo);

}

 void display()

{

 System.out.println("Name of Student: " + Name);

}

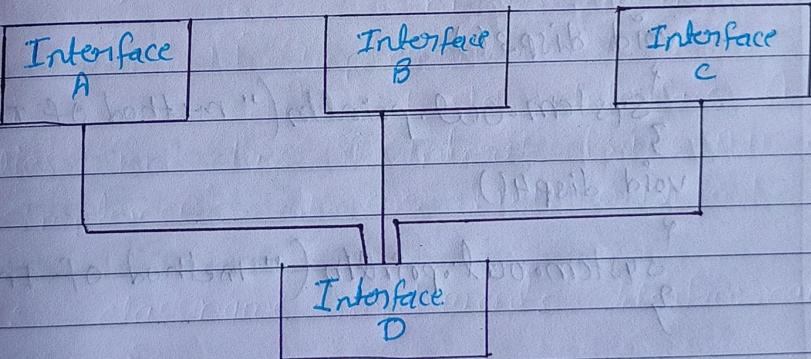
```
public static void main (String args[])
```

```
{  
    student s = new student();  
    s.show();  
    s.Display();
```

```
}
```

Multiple Inheritance :-

multiple inheritance में किसी एक class के द्वारा multiple Interface की property को access करते ही सकता है। यह class के द्वारा possible की होता है। इसलिए Java में multiple Inheritance archive करने के लिए Interface को use किया जाता है।



Program:-

```
Interface A
```

```
{  
    void disp1();  
}
```

```
Interface B
```

```
{  
    void disp2();  
}
```

Interface C

{

void disp3();

}

Interface D Extends A, B, C

{

void disp4();

}

class E implements D

{

void disp1()

{

System.out.println ("method of Interface A");

}

void disp2()

{

System.out.println ("method of Interface B");

}

void disp3()

{

System.out.println ("method of Interface C");

}

void disp4()

{

System.out.println ("method of Interface D");

}

public static void main (String args[])

{

E e=new E();

e.disp1();

e.disp2();

e.disp3();

e.disp4();

}

Output:

metho of Interface - A

B
C
D

Polymorphism :- यह दो words से मिलकर होता है।

Poly (many) + morphism (forms).

in object oriented programming का एक feature है जिसमें एक एक method को कई तरह से use करते ही संकेत होता है।

Polymorphism greek language का word है जिसमें poly means many वह morphism means many forms होता है। यह एक ऐसा concept है जिसके द्वारा कई काम को अलग-अलग तरीके से किया जा सकता है। यह दो प्रकार का होता है-

- (i) Compile time polymorphism
- (ii) Run time polymorphism

इसकी method overloading या method early binding का भी होता है।

method overloading में एक इकार के method name होती है। datatype of method & its parameter different - different होती है।

Example:-

class Sample

```
    {
        System.out.println("Value of x: "+x);
        System.out.println("Value of y: "+y);
    }
```

void disp (int a)

```
    {
        System.out.println("Value of x: "+x);
    }
```

public static void main (String args[])

Sample s = new sample();

s.disp(3, 5);

s.disp(4);

??

Output:

value of x = 3

value of y = 5

value of x = 4

Runtime Polymorphism :-

Runtime polymorphism का लिए लेट बाइंडिंग,
Dynamic binding के द्वारा method overriding
का लिए है।

Runtime परोल्यॉमॉर्फिशन के JVM एटी

method के runtime में call होता है।
method overriding में same method name
same data type तथा same parameter list
होता है। method overriding में
first method call के लिए second method
call होता है; मानव इस को अलग होता है।
first method के place के 2nd method के
एटी override. भूत जाता है।

Same as previous (overriding example.)

Class A

{

void display()

{

System.out.println("Class A method")

}

Class B Extends A

{

void display()

~~finalize()~~ method object class का method
जो父 class implementation object class
empty state का उत्तर नहीं करता है तो इसे बदलकर
method overridden का उत्तर देता है। इसका
उत्तर जो भी finalizer() method का
overridden method हो।

{ System.out.println ("class B method");

} public static void main (String args[])

{

A obj = new A();

B obj = new B();

}

}

Binding →

अगर program का method call का method
definition से उत्तर binding नहीं होता है।
Java में यह 2 तरीके होते हैं।

- method
- (i) static Binding
 - (ii) Dynamic Binding

difference b/w static binding, and dynamic binding

Static Binding

(1) Static binding की
early binding है।
compile time polymorphism
केरा जाता है।

इसमें binding compilation में होती है।

(2) Static binding method
overloading के स्क्रिप्ट archive
फॉरम जाता है।

(3) क्योंकि static binding
compile time में होती है।
एकल program के execution
execution fast होता है।

(4) static method call के
matching की method
definition के साथ compile
time पर होती है।

Dynamic Binding

(1) Dynamic Binding की
runtime and late binding and
runtime polymorphism
केरा जाता है।

(2) static binding runtime
में होती होती है।

(3) Dynamic binding की
method overriding के
script archive के
उपरा है।

(4) static binding
runtime में होती है।
के program के execution
static binding के
comparison में slow होता है।

(5) static method call की की
matching की definition के साथ
runtime पर होती है।

CO-3: Apply Multithreading and Exception handing feature in Java

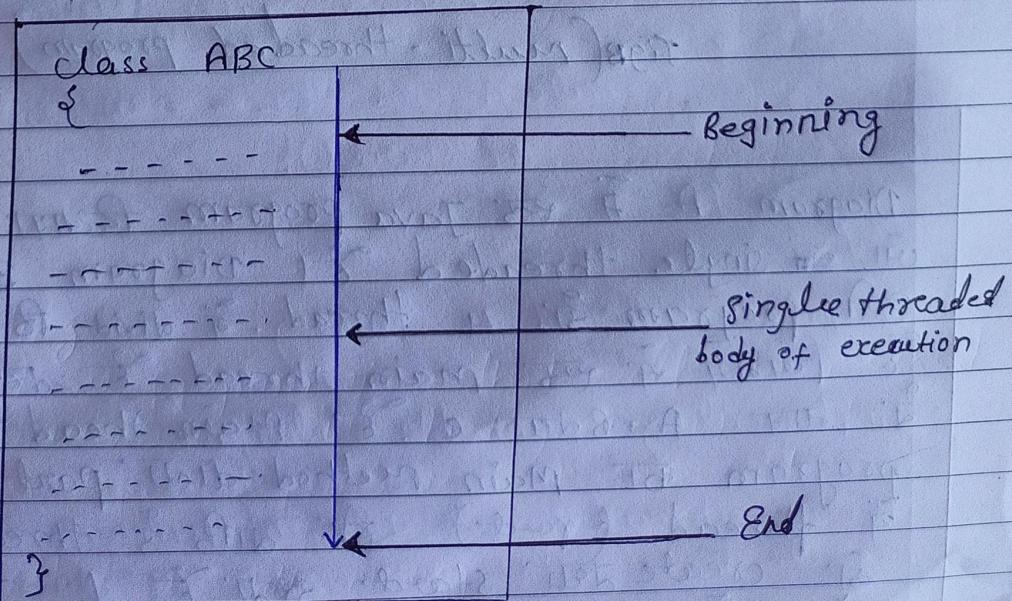
Multi threading - Multi-threading three ^{सह} paradigms
Concurrent programming ^{पारा}

जैसे प्रोग्राम में एक program की multiple हाई-लोवर्स
part में divide किया जाता है। इनके
Subpart की thread का भाग है। वे सभी
thread CPU के SRT Simultaneously parallel
execute होते हैं।

Multithreading Java की unique property है कि

program develop करने के लिए multiple flow
flow of control की use करना यही यही
विशेषता है। लेकिन एक program की minimum एक thread
होता है।

thread program के similar जैसा होता है, तथा
thread may be program के single flow
of control होता है जिसके commands के sequentially
execute करता है।



Main method
module

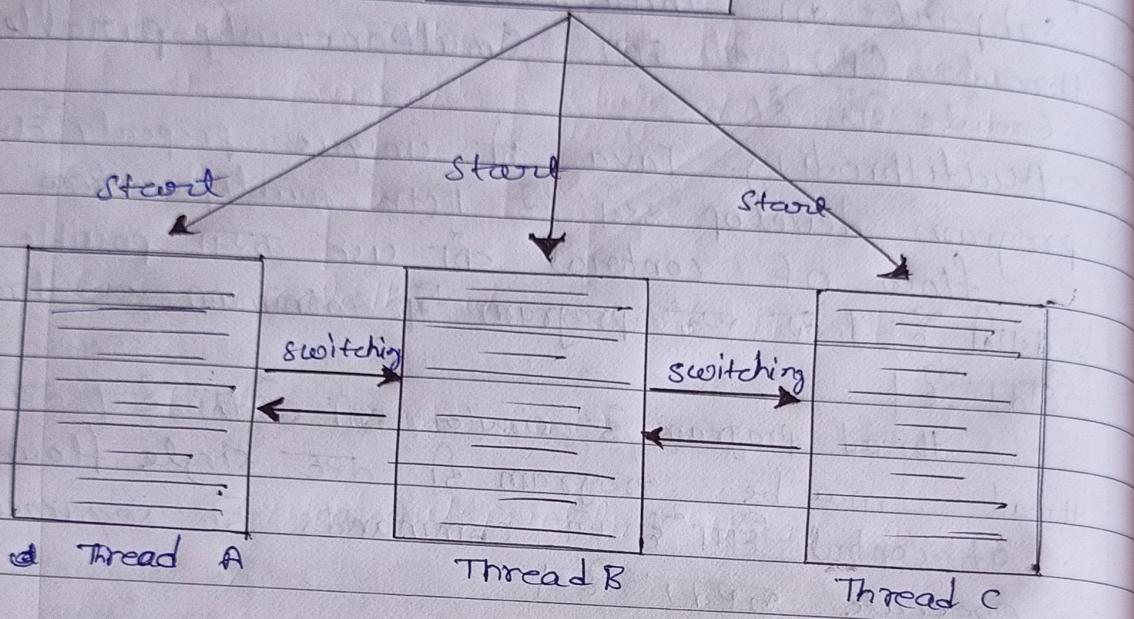


Fig B (multi-threaded program)

Diagram A ने Java program को बताया है।
जो single threaded है। Diagram B में
यही program को 4 thread के लिए बदला गया है।
जो बिन्दु से इस main thread के बाहर आते हैं
जो नाम A, B वा C हैं। Main thread के
program को Main method की part होती है।
ये thread (प्रैट) की तरफ से threads (प्रैट)
को create करते हैं। start करते हैं।
अब से पहले main thread run होता है। और
वार्ता के thread Resources को share करते हैं,
इसे, यहां Run होते हैं।

Creating a thread :-

Java में thread create करना बहुत easy होता है।
thread object के form में implement किये जाने हैं जिसकी साथी प्राप्ति पर method होती है। इस
method की Run method जैसा भारा है।
Run method जैसा thread की Body implement
की जाती है। Run method की तरफ इस
से Declare किया जाता है।

Syntax : `public void run()`

Statement for implementing thread

Run method को thread के object के द्वारा
invoke (call) किया जाता है। Run method को call
करने के लिए start method का use किया
जाता है।

Thread के लिए कैसे create किया जा सकता

- 1). By Extending thread class
- 2). By Implementing Runnable interface

(1) By Extending Thread Class

- Thread create करने के लिए एड class को create
किया जाता है। फिर इस class को thread class
को extend किया जाता है।
- Run method को override किया जाता है। इस
method को start method के लिए
call किया जाता है।

• main method में class का object लिया जाता है इस object के द्वारा start method को call किया जाता है।
start method को call किये पर thread का execution start हो जाता है।

Start method thread का new state
& से Runnable state नहीं जाता है।

Example :

```
class CreateThread extends Thread
{
    public void run()
    {
        System.out.println("Thread is running.");
    }

    public static void main (String args[])
    {
        CreateThread t1 = new CreateThread();
        t1.start();
    }
}
```

Output : Thread is running.

Implementing Runnable Interface

There Thread create किये के लिए यह class को create किया जाता है। तथा उस class को Runnable interface को किया implement किया जाता है। इसके बाद run method को override किया जाता है। तथा class को object create किया जाता है। तथा thread class का object create किया जाता है।

जब thread class के object ने start method को override किया है। Thread class के object के create करने के समय, thread class के constructor की class के object की reference भेज देता है।

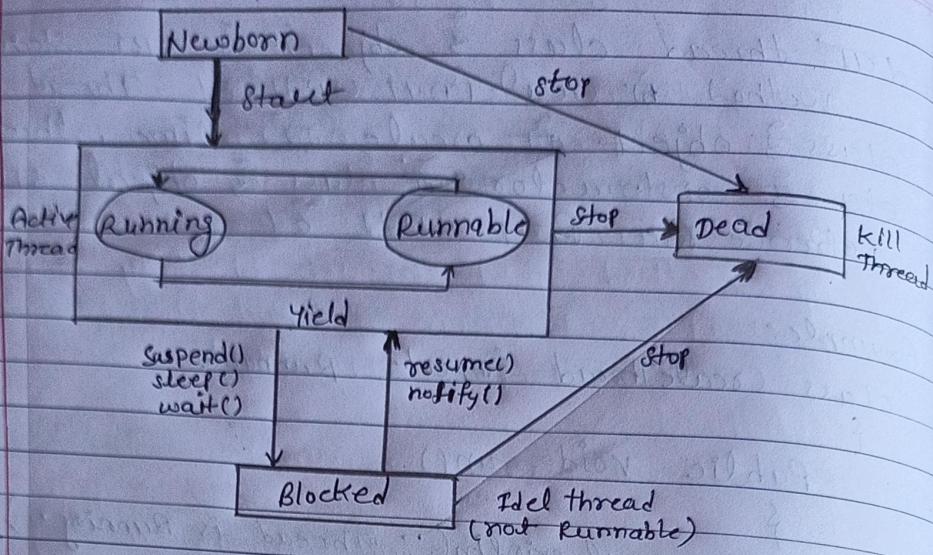
Example:

```
class CreateThread implements Runnable  
{  
    public void run()  
    {  
        System.out.println("Thread is Running").  
    }  
    public static void main(String args[])  
    {  
        CreateThread c = new CreateThread();  
        Thread t1 = new Thread(c);  
        t1.start();  
    }  
}
```

LIFE CYCLE OF THREAD

एक thread का life cycle में 5 steps होते हैं।
एक thread अपने पूरे life time में इन 5 stages के होते हैं जैसे कि Java Thread का life cycle जो JVM के बीच control लिया जाता है।
5 states निम्नलिखित हैं—

- 1). New born state
- 2). Runnable state
- 3). Running state
- 4). Blocked state
- 5). Dead state



Dia. State Transition of Thread

Newborn State

यह एक new thread की # Create किया जाता है यह newborn state होता है इस state में thread alive नहीं होता है और पर execution के लिए ready नहीं होता है जैसे कि start method की call किया जाता है thread newborn state से Runnable state में चला जाता है। Runnable state से thread newborn state में वापस नहीं आ सकता है।

Runnable State:

Start method के call के पर thread newborn से Runnable state में आ जाता है। Runnable का meaning कि thread run होने के लिए ready के लिए यह अभी processor के free wait के रहा है। Thread Scheduler की thread के scheduling algorithm (Round Robin) first serve, shortest Job first & according processor available की ओर।