1) class Demo11{
p. s. v. m (S qr[]){
S.o.pln ("Softwaves-1");
int x = Integer.parseInt (qr[0]);
S.o.pln ("Softwaves-2");
int y = Integer. parseInt(qr[1]);
S.o.pln ("Softwaves-3");
S.o.pln (x/y);
S.o.pln ("Softwaves-4");
}
}

o/p → 1) Demo11 10 2
o/p → Softwaves-1
Softwaves-2
Softwaves-3
5
Softwaves-4

2) 0 2
o/p → Softwaves-1
Softwaves-2
Softwaves-3
0
Softwaves-4

3) 10 0
o/p → Softwaves-1
Softwaves-2
Softwaves-3
Exception :- Arithmetic Exception:
(1) diviid by zero

4) 10 ab
o/p → Softwaves-1
Softwaves-2
Exception :- Number Formate Exception:
for input string
"ab"

5) 10
o/p → Softwaves-1
Softwaves-2
Exception :- Array Index Out Of Bounds:
for index 1

6)
o/p → Softwaves-1
Exception :- N.F.E.: for input String "ab"

7)
o/p →

Excepti

what is

An Ex
whic
proc
the

Excep

Suppos
you
exce
code
to

if
the

That

→ Exception is only occurs at Run time

DATE :
PAGE :

what is Exception?

An Exception is an unwanted or unexpected event. which occurs during the execution of a program i.e at Run time, that distrupts the normal flow of the program.

Exception Handling.

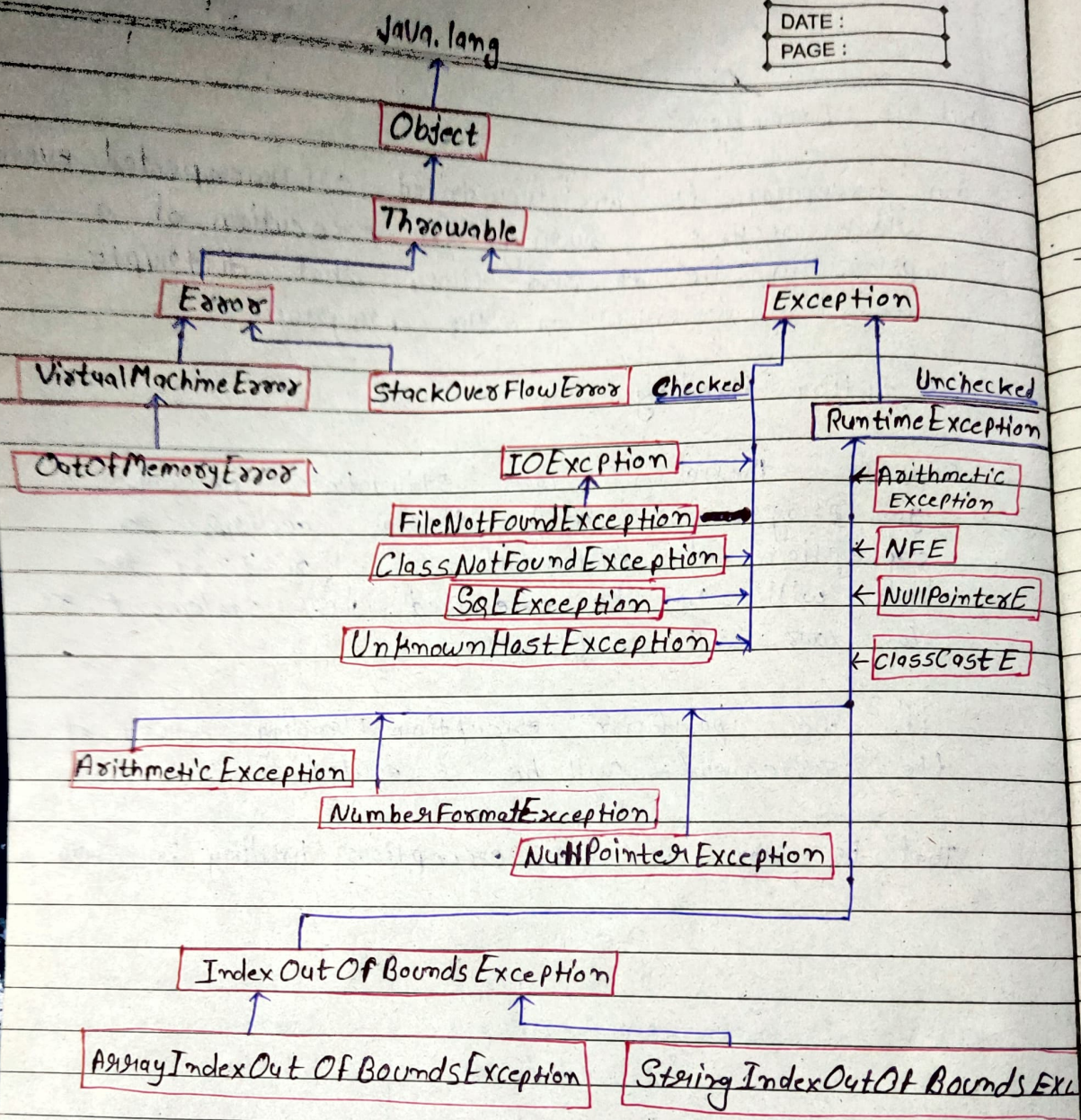
Suppose there is 1000 statements in your ~~you~~ program and there occurs an exception at statement 50, rest of the code will not be executed i.e. statement 51 to 1000 will not Run.

if we perform exception Handing, rest of the statement will be executed.

That is why we use exception handling in Java.

Exception Hierarchy

DATE :
PAGE :



DATE :
PAGE :

There are two types of exceptions.

1) Checked 2) Unchecked.

The main difference btween checked and unchecked Exception is that the checked exception are checked at compile-time while unchecked exception are checked at runtime.

Runtime and its subclasses are unchecked exception

- what is the super class for all Exception and Error

→ Throwable

→ what is the super class for all Exception?

→ Exception

→ Exception Handling Keyword

try catch finally throw

throws

after try → catch
→ finally
→ resource declerations

1) class Demo11
{ p. s. v. m (S ar(1)){

int x = Integer. parseInt (ar(0));
int y = Integer. parseInt (ar(1));

try {
S. o. p ln (x/y);
}

S. o. p ln ("Softwaves - 1000");
}
}

o/p → Error!- 'try' without 'catch', 'finaly', or resource declerations

2) class Demo11 {
p. s. v. m(S ar(1)){

o/p → Error: else without 'if'

else { S. o. p ln ("softwaves - 1000");
}
}}

try block

programmer when observe that there is chance of coming exception in the code then that code is written inside the try block

we cannot use try block single, we must use try with the, 'catch', 'finally', or with resource declaration otherwise it give error.

DATE :
PAGE :

3) catch block

the catch block always come after the try block.

no statement or block can come b/w the try and catch. block:

```
try {
      //sssssss---
    }
catch (Exception_name e1)
    {
      //sita-----
    }
```

8) class demo11 {
p. s. v. m (S ar[]) {

o/p → Error : catch without 'try'

```
catch (Arithmatic Exception e)
{
  S.o.pln(" arithmetic -- Exception... hand");
}
}
}
```

4) class demo11 {
p. s. v. m(S ar[]) {

```
int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
try {
      s.o.pln(x/y);
    }
catch (ArithmeticException e) {
    S.o.pln("arithemetic--- exception -- hand");
}
S.o.pln(" softwaves-1000");
}
}
```

1- try can have multiple catch

DATE :
PAGE :

case-1 Demo11 10 2 5 Softwaves-1000	2- 0 2 0 Softwaves-1000	3→ 10 0 arithemetic-- Exception-- handl Softwaves-1000
4→ 10 ab Exception:-NFE for i/p string 'ab'	5→ 10 Exception: AIOOBE: 1	6→ Exception: AIOOBE: 0

→ when exception is generated in try block then catch block is executed.

→ we can use multiple catch block with one try block.

5]
```
class Demo11{
    p. s. v. m(S ar[]){
    try{
        int x = Integer.parseInt(ar[0]);
        int y = Integer.parseInt(ar[1]);
        S.o.pln(x/y);
    }
    catch(ArithmeticException e){
        S.o.pln("Arithmetic.Exception... hand");
    }
    catch(NumberFormatException e){
        S.o.pln("Number... Exception... hand");
    }
    S.o.pln("Softwaves-1000");
}
}
}
```

DATE :
PAGE :

o/p - case-1 demo 10 2 5 Softwaves-1000	2→ 0 2 0 Softwaves-1000	3→ 10.0 Arithematic Exception hand Softwaves--1000
4→ 10 ab NFE... hand Softwaves-1000	5→ 10 AIOOBE: 1	6→ ab Number... exception... hand Softwaves--- 1000

→ the type of exception is generated in try block that type of catch block is executed

→ case 6→ at a time only one exception is generated in try block because the program is terminated from that point and if its corresponding catch block is available then it is executed.

→ corresponding to one try block their is only one exception ~~still~~ can Generates at a time. but their are multiple possibilities of multiple exception but when the first exception arive the try block get termint

1) Class Demo6
{ public static void main(String a[]){

O/P → Demo6 10 20
0

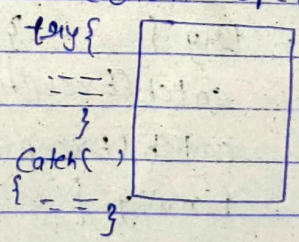
System.out.println("100 lines code");
try{
int x = Integer.parseInt(a[0]);
int y = Integer.parseInt(a[1]);
System.out.println(x/y);
}
catch(ArithmaticException e){
System.out.println("arithmatic ex.....");
}
}
}

finally

always executable block

finally :- it is always executable block.
catch block is executed when exception is generated in try block but the finally block is always executed irrespective of whether the exception is generated or not.

Use of finally block → ① Connection open



② Connection close

2) class Demo7{
public static void main(String a[]){
System.out.println("Connection open....");
System.out.println("100 line code");
try{ int x = Integer.parseInt(a[0]);
int y = Integer.parseInt(a[1]);
System.out.println(x/y);
}
catch(ArithmeticException e){
System.out.println("arithmetic ex.....");
}
System.out.println("7000 lines code");
System.out.println("Connection close");
}
}

o/p → case-1 → 10 2	→ 10 0	10 abhi
Connection open	Connection open	Connection open
100 line code	100 line code	100 line code
5	arithmetic ex....	NFE → Exception (termina
7000 lines code	7000 lines code	NFE for input String "abhi"
Connection close	Connection close	

→ The code which we want to execute irrespective of whether the exception is generated or not then we write that code in finally block.
ex. file closing, connection closing etc.

Syntax → try { } finally { }

→ try { } catch(Exception_1 e1){ } catch(Exception_1 e1){ } finally { }

→ we can use finally after the try block or after the catch block.

3] class Demo8
{ public static void main(String a[]){
System.out.println("connection open...");
System.out.println("100 lines code");
try{
int x = Integer.parseInt(a[0]);
int y = Integer.parseInt(a[1]);
System.out.println(x/y);
}
finally { System.out.println("Connection close...");
}
}}

Case → 10 2	→ 10 0	→ 10 abhi
Connection open...	Connection open	Connection open
100 lines code	100 lines code	100 lines code
5	Connection close...	Connection close...
Connection close	Exc→AE: \ by '0'	EX→NFE: for input String "abhi"

4] class Demo9{
public static void main(String a[]){
System.out.println("Connection open...");
System.out.println("100 lines code");
try{ System.out.println("Softwaves-1...");
int x = Integer.parseInt(a[0]); System.out.println("Softwaves-2...");
int y = Integer.parseInt(a[1]);
System.out.println(x/y);
System.out.println("Softwaves-3...");
} catch(ArithmeticException e){
System.out.println("arithmetic EE...");
} catch(ArrayIndexOutOfBoundsException e) { S.o.pln("AIex...");
} finally{
System.out.println("connection close...");
} System.out.println("Softwaves-4...");
}}

o/p- case → 10 2

#no exception is arise then exicated code is

→ before try block all code will execute
→ try code will execute
→ no catch block no execute
→ finally block will execute
→ code after try, catch finally will execute

o/p → connection open...
100 lines code
Softwaves-1...
Softwaves-2...
Softwaves-3...
Connection close...
Softwaves-4...

case → 10 0

if exception is arising and we had handled those exception

→ before the try block all code will execut
→ code before the line of exception will execute
→ corrcsponding catch block will execute
→ after catch finally block execute
→ code after try, catch, finally will execute

o/p → Connection open
100 lines code
Softwaves-1...
Softwaves-2...
→ Arithmatic ex...
Connection close...
Softwaves-4...

case → 10 abhi

if exception is generating but not handled.

→ before try block all code will execute
→ code before the line of exception will execute
→ exception not handled so no catch will execute
→ finally code will execute
→ try, catch, finally blocks next code will not execute
→ exception is not handled so after finally block exception will occurs.

→ o/p → connection open...
100 line code
Softwaves-1...
Softwaves-2...
Connection close...
E: NFE: for input string "abhi"

throw

The throw Keyword in Java is used to manually throw an exception from a method or any block of code.

We can throw either checked or unchecked exception.

→ The ~~to~~ throw Keyword is mainly used to throw custom exceptions.

```
1) class Demo10{
    public static void main(String ar[])
{
int age = Integer.parseInt(ar[0]);

if(age<18)
 {
    System.out.println("Invalid Age");
 }
else{
    System.out.println("~~soft~~Welcome");
 }
    System.out.println("SoftWaves-1...");
}
}
```

O/P→ Demo10 15	→ 25
→ Invalid age	- Welcome
SoftWaves-1...	SoftWaves-1...

if we want to generates any time of Exception as per our requirement. then we use throw Keyword

2) class Demo1
{ public static void main(String ar[])
{
int age = Integer.parseInt(ar[0]);

if(age<18)
{
ArithmaticException ae = new ArithmeticException();
throw ar;
}

else{
System.out.println("welcome");
}

System.out.println("Softwaves-1");
}}

o/p→ 25	→ 15
→ welcome	→ Exception: ArithmeticExection
Softwaves-1...	

3) class Demo2
{ public static void main(String ar[])
{
int age = Integer.parseInt(ar[0]);

if(age<18){
ArithmeticException ae =new ArithmeticException("Invalid age");
throw ae; }
else { System.out.println("welcome"); }

}System.out.println("Softwaves-1...");

o/p→ 15.	→ 25
→ Invalid age Exception	→ welcome
ArithmeticException:	Softwaves-1...
Invalid age	

```
class Demo {
    public static void main(String ar[]) {
        int age = Integer.parseInt(ar[0]);

        if(age<18) {
            ArithmeticException ae = new ArithmeticException("Invalid age
                                     is=" + age);
            throw ae;
        }
        else {
            System.out.println("walcome");
        }
        System.out.println("Softwaves_1...");
    }
}
```

o/p → 25
→ walcome
Softwaves-1:--

17
Ex: ArithmeticException: Invalid age is=17

* we wont to create user define exception. so--

→ we can create by help of Exception and their derived classes.

User define Exception

in Java we can create our own exception class and throw that Exception using throw keyword. These exception are known as User-define or custom exception.
Java provides us facility to create our own exception which are basically derived classes of Exception

folder → JavaDemo(EH)

EH → InvalidAgeException.java

```
class InvalidAgeException extends RuntimeException
{
    InvalidAgeException()
    {
    }
    InvalidAgeException(String s)
    {
        super(s);
    }
}
```

EH → Demo.java

```
class Demo
{
    public static void main(String ar[])
    {   int age = Integer.parseInt(ar[0]);
if(age<18){
InvalidAgeException ae = new InvalidAgeException();
        throw ae;
    }
else{
        System.out.println("welcome");
    }
    System.out.println("Softwaves_1...");
}
}
```

o/p → case → 25
→ welcome
Softwaves_1...

→ 15
Exception: InvalidAgeException

2)

```
class Demo {
  public static void main(String ar[]){
  int age = Integer.parseInt(ar[0]);
  if(age<18){
    InvalidAgeException ae = new InvalidAgeException
                             ("Invalid Age is="+age);
    throw ae;
  }
  else{
      System.out.println("welcome");
    }
    System.out.println("softwaves_1..");
  }}
```

o/p → case → 25
→ welcome
softwaves_1..

→ 15
E: InvalidAgeException: Invalid age is=15

3)

```
class Demo
  {
    public static void main(String ar[]) {
  int age = Integer.parseInt(ar[0]);
  if(age<18){
throw new InvalidAgeException("Invalid age is="+age);
    }
  else{ System.out.println("welcome"); }

    System.out.println("softwaves_1..");
  }}
```

o/p → case → 25
→ welcome
→ softwaves_1..

→ 15
E: InvalidAgeException: Invalid age is=15

throws

यदि हमारे program में RuntimeException (unchecked) generate हो रही है तो तो code compile हो जायेगा

लेकिन यदि program में checked exception आ रही है तो या तो try, catch से handle करना होगा

otherwise throws keyword से उसे method के बाहर करना पड़ेगा

जो कि बाहर होके Calling method के पास जायेगी

→ if any unchecked exception is generated in our program than compiler has no problem with that but,

if any checked exception generated in our program then we have to handle it with try, catch or we have to throw it out from the method using throws keyword.

```
import java.io.*;
class Demo15
{
    public static void main(String ar[])
    {   System.out.println("softwaves-1...");
        PrintWriter pw = new PrintWriter("xyz.txt");
        pw.print("Softwaves");  pw.close();
        System.out.println("softwaves-2...");
    }
}
```

o/p → unreportedException : FileNotFoundException : must be caugh or declared to be thrown

throws

यदि हमारे program में RuntimeException (unchecked) generate हो रही है तो भी code compile हो जायेगा

लेकिन यदि program में checked exception आ रही है तो या तो try, catch से handle करना होगा

otherwise throws keyword से उसे method के बाहर करना पड़ेगा

जो कि बाहर होके Calling method के पास जायेगी

→ if any unchecked exception is generated in our program then compiler has no problem with that but,

if any checked exception generated in our program then we have to handle it with try, catch or we have to throw it out from the method using throws keyword.

```
import java.io.*;
class Demo15
{
    public static void main(String ar[])
    { System.out.println("softwaves_1...");
PrintWriter pw = new PrintWriter("xyz.txt");
    pw.print("Softwaves"); pw.close();
    System.out.println("Softwaves_2...");
}
}
```

o/p → unreportedException : FileNotFoundException : must be caugh or declared to be thrown

5) import java.io.*;
class Demo
public static void main(String arg[]){

System.out.println("Softwaves-1...");

try{
PrintWriter pw = new PrintWriter("xyz.txt");
pw.print("Softwaves"); pw.close(); }
catch(FileNotFoundException e)
{
}
System.out.println("Softwaves-2--");
}}

o/p → Softwaves-1...
Softwaves-2--

xyz.txt → Softwaves

throws

6) import java.io.*;
class Demo{
public static void main(String arg[]) throws
FileNoteFoundException
{
System.out.println("Softwaves-1...");

PrintWriter pw = new PrintWriter("xyz.txt");
pw.print("Indore");
System.out.println("softwaves-2...");
pw.close();
}}

o/p → Softwaves-1...
Softwaves-2--

xyz.txt → Indore

Exception is always generate at the Run time

Lacture 6

1) import java.io.*;
class Demo
{
public static void main(String arg[])
{
System.out.println("softwaves-1...");
PrintWriter pw = new PrintWriter("xzy.txt");
pw.print("softwaves");
pw.close();
}
}

o/p → unreported E: FileNotFoundException

Lacture-6

Checked exception:- exception which is checked by compiler is called checked exception

unchecked exce. :- exception which is cannot be checked by compiler.
these are checked by JVM. is called unchecked exception

○ What is checked exception...?

Ex → यदि हम किसी बस में सफर करते हैं या तो कभी भीड़ वाले इलाके में जाते हैं वहाँ पर कई बार एक warning message लिखा हुआ रहता है।

कृपया आप अपने सामान की सुरक्षा खुद ही करें यदि आपका सामान चोरी होता है तो इसकी जिम्मेदारी आपकी स्वयं की है।

यह warning msg. इसलिए लिखा होता है क्योंकी इसी घटना घट सकती है। तो आप सावधान रहें।

ऐसा ही भीड़ वाली जगह पर लिखा होता है कि जब कैंची से सावधान---

देखे ही कुछ

xyz.
proble

इसी

Ex

क्योंकी

हो
हो

1) import
cla

{

}

o/p → E

जैसे ही Demo class में PrintWriter से xyz.txt file में कुछ write करना चाहते हैं लेकिन हो सकता है।

xyz.txt file उस folder में हो ही ना या कोई और problem के कारण file में write नहि कर पाये

इसी problem आ सकती है सालिए compile the पर जो Exceptions आती है। वो एक warning msg. होता है।

क्योंकी अगर exception अगर Run time पर आयी तो हो सकता है। आगे का code भी 100, 1000, 10000 lines का हो. और चले ही नहि

```
1) import java.io.*;
   class Demo{
      public static void main(String ar[])
      {
         System.out.println("Softwaves-1");
         PrintWriter pw = new PrintWriter("xyz.txt");

         System.out.println("Softwaves-2");
      }
   }
```

o/p → E: FileNotFoundException: use try, catch

2) import java.io.*;
class Demo{
public static void main(String arg[])
{
System.out.println("softwaves-1");
try{
PrintWriter pw = new PrintWriter("xyz.txt");
}
catch(FileNotFoundException e)
{
System.out.println("abhi");
}
System.out.println("softwaves-2");
}?

o/p → softwaves-1
softwaves-2

→ abhi msg नहीं आया क्योंकी try में exception नहीं आ रही तो एक warnning msg ही था।
अगर रसीलिस catch blocke नहीं देते तो चेला।

Lacture 7

1) class Demo{
public static void main(String arg[])
{
System.out.println("softwaves-1");
System.out.println("softwaves-2");
}}

o/p → softwaves-1
softwaves-2

2) class Demo{
public static void main(String arg[]){
System.out.println("softwaves-1");
System.out.println(10/2);
System.out.println("softwaves-2");
}}

o/p → softwaves-1
5
softwaves-2

3) class Demo{
public static void main(String ar[]){
System.out.println("softwaves_1");
System.out.println(10/0);
System.out.println("softwaves_2");
}
}

o/p → Softwaves_1
E: ArithmeticException: ~~e = new ArithmeticException~~ / by '0'

4) class Demo {
public static void main(String ar[]) throws AE
{
System.out.println("softwaves_1");
System.out.println(10/0);
System.out.println("softwaves_2");
}
}

o/p → Softwaves_1
ArithmeticException: / by '0'

Note ⇒ Unchecked Exception कभी भी throws keyword से method के बाहर throw नहीं होती only ~~un~~checked exception को ही throws से बाहर कर सकते हैं।

Run time stack mechanisim :

5) Class Demo {
public static void main(String ar[]){
System.out.println("Softwaves_1");
}}

o/p → Softwaves_1

6) class Demo{
Void Show1(){
System.out.println("softwaves_2.");
System.out.println("Softwaves_3..");
}
public static void main(String ar[])
{
System.out.println("Softwaves_1");
Show1();
System.out.println("Softwaves_4");
}}

o/p → Error: non-static method show1() cannot be reference from static content

7) class Demo{
static void show1(){
System.out.println("Softwaves_2");
System.out.println("Softwaves_3");
}
public static void main(String ar[]){
System.out.println("softwaves_1");
Show1();
System.out.println("Softwaves_4");
}}

o/p → Softwaves_1
softwaves_2
Softwaves_3
Softwaves_4

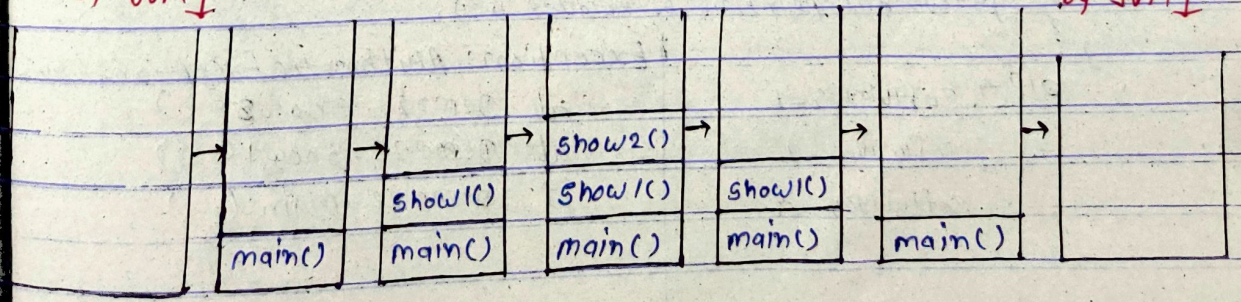
Java के हर program में bydefault एक Thread होती है।
जिसे main Thread करते है।

```
class Demo{
    static void show2(){
        System.out.println("softwaves-5");
    }
    static void show1(){
        System.out.println("softwaves-2"); show2();
        System.out.println("softwaves-3");
    }
    public static void main(String arr[])
    {
        System.out.println("softwaves-1");
        show1();
        System.out.println("softwaves-4");
    }
}
```

O/P → softwaves-1
softwaves-2
softwaves-5
softwaves-3
softwaves-4

Ex→ 8, सबसे पहले JVM ने main Thread के लिए stack create किया। और main method को call (entry) किया, then show1() की entry हुई फिर show1() ने show2() को call किया इसलिए show2() की entry हुई क्योंकी show2() में कोई दूसरी method call नहीं हो रही इसलिए पहले show2() complete execute होगी और फिर stack से remove होगी फिर show1() का remaining code execute होगा then उसे भी JVM stack से remove करेगा। and then main() को। then stack के empty होने पर JVM stack को destroy कर देगा

Stack created by JVM ↓



Stack destroyed by JVM ↓

Run-Time stack mechanism in Java

Lacture-8

- for every Thread Jvm creates a run-time stack.
- Jvm store every methods calls performed by that Thread in the stack
- After completing every method call by the thread is removed from the stack
- after completing all methods, the stack will be empty and that Run-time stack will be destroyed by the Jvm before terminating the thread.

Lacture-8

```
1) class Demo22{
     static void show2(){
       System.out.println("softwaves-5...");
       System.out.println(10/0);
       System.out.println("softwaves-6...");
     }
     static void show1()
     {
     System.out.println("softwaves-2...");
     show2();
     System.out.println("softwaves-3...");
     }
   public static void main(String ar[]){
     System.out.println("softwaves-1");
       show1();
     System.out.println("softwaves-4");
   }}
```

O/P → Softwaves-1
Softwaves-2
Softwaves-5

Exception: Arithmetic Exception: / by '0'
at Demo22.show2( )
at Demo22.show1( )
at Demo22.main( )

NOTE

→ when o
metho
whet
meth

→ if the
except

→ if the
code
and

→ now
handl

→ if th
then
and

→ the
meth

→ now
code

→ if
the
before
excep
defa
the

Sym

NOTE

→ when an exception is generated in a method than the method handover it to the JVM then JVM will check whether the exception handling code is in the method or not

→ if the method contain exception handling code then the exception is handld.

→ if the method does not contain the exception handling code than JVM will terminate the method abnormally and remove its entry from the runtime stack.

→ now JVM will check whether the calling method has handled the exception or not.

→ if the exception is not handled by calling function then JVM will terminate that method abnormally and remove its entry from the runtime stack.

→ the process is continue untill it reach upto the main method.

→ now JVM will check that the main method contain the code for exception handling or not.

→ if main method does not handle the exception then the main method is terminated abnormally. But just before the program terminate the JVM forword the exception to the default exception handler and the default exception handler prints the exception in the ~~jvm~~ form of stack.

Syntex of printing Exception ⇒

Name of Exception : description

stackTrace

2]
```
class Demo23{
    static void show2(){
      System.out.println("softwaves_5..");
      System.out.println(10/0);
      System.out.println("softwaves_6");
    }
    static void show1(){
      System.out.println("softwaves_2..");
      try{
          show2();
        }
      catch (ArithmeticException e)
        {
          System.out.println("abhi");
        }
      System.out.println("softwaves_3");
    }
  public static void main(String ar[])
    {
      System.out.println("softwaves_1");
       show1();
      System.out.println("softwaves_4");
    }
}
```

o/p →
```
Softwaves_1
Softwaves_2
Softwaves_5
abhi
Softwaves_3
Softwaves_4
```

3)
```
class A{
    static void show2(){
     System.out.println("Softwaves-5");
     System.out.println(" 1010);
     System.out.println("Softwaves-6");
    }
}

class B{
    static void show1()
    {
     System.out.println("Softwaves-2");
      A.show2();
     System.out.println("Softwaves-3");
    }
}

class Demo24{
   public static void main(String ar[])
   {
     System.out.println("Softwaves-1");
     B.show1();
     System.out.println("Softwaves-4");
   }
}
```

o/p → Softwaves-1
Softwaves-2
Softwaves-5

Exception :- ArithmaticException : / by zero
at A.show2()
at B.show1()
at Demo24.main()

Exception propagation

Lacture - 9

1) import java.io.*;

```
class Demo
 {
  public static void main(String ar[])
   {
    PrintWriter pw = new PrintWriter("xyz.txt");
    System.out.println("Softwaves-1");
   }
 }
```

o/p → error: unreported exception FileNoteFoundException: must be callght or declared to be thrown

2) import java.io.*;

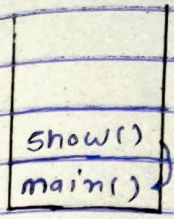
```
class Demo{
public static void main(String ar[]) throws FileNot
                                    : FoundException
 {
  PrintWriter pw = new PrintWriter("xyz.txt");
  System.out.println("Softwaves-4");
 }
}
```

o/p → Softwaves-4

3) import java.io.*;

```
class Demo{
 static void Show1()
  {
   PrintWriter pw = new PrintWriter("xyz.txt");
  }
public static void main(String ar[])
 {
  Show1();
  System.out.println("Softwaves-4");
 }
}
```

o/p → error:- Unreported Exception FineNFE: must be caught or declared to be thrown



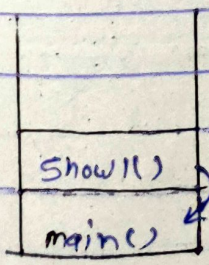
show() में throws का use करेंगे तो Exception calling method main() के पास जायेगी और हमें main() में भी throws करना है।

show() से FNFE को throws keyword से throw कराने पर throws उस Exception को drop down कर देता है। (Send Exception to calling method को।

→ throws keyword does not handle Exception it delegates the Exception to calling method.

4) import java.io.*;
class Demo
{
static ~~public~~ void ~~pvsvl~~ show() throws FileNotFoundException
{
printWriter pw = new PrintWriter("xyz.txt");
}
public static void main(String arg[])
{
show();
System.out.println("Softwaves 4");
}
}

o/p → error: unreported Exception:
FileNotFoundException: must be caught or declared to be thrown



5) import java.io.*;

```
class Demo
{
  static void show1() throws FileNotFoundException
  {
  PrintWriter pw = new PrintWriter("xyz.txt");
  }
}
public static void main(String a[]) throws FileNotFoundException
{
   show1();
  System.out.println("softwaves_4");
}
}
```

o/p → softwaves_4

6) import java.io.*;

```
class Demo
{
static void show2()
   {
    PrintWriter pw = new PrintWriter("xyz.txt");
   }
static void show1(){
   show2();
  }
public static void main(String a[])
{
   show1();
  System.out.println("softwaves_4");
  }
}
```

o/p → unreported Exception (show2()) FNFE : must be caugth or declered as thrown

o/p →

show2() throws FNFE
show1() throws FNFE
main() throws FNFE .

7) import java.io.*;

class Demo
{
static void show2() throws FileNotFoundException
{
PrintWriter pw = new PrintWriter("xyz.txt");
}
static void show1() ~~throws FileNotFoundException~~
{
show1();
}
public static void main(String a[])
{ show1();
S.o.p("software=1");
}
}

o/p → ↗ show1()
unreported Exception:
FNFE:

8) import java.io.*;
class Demo{
static void show2() throws FNFE
{
PrintWriter pw = new PrintWriter("xyz.txt");
}
static void show1() throws FileNotFoundException
{
show2();
}
static public void main(String ar[]) ~~throws~~
{
show1();
S.o.p("software=1");
}
}

↗ main()
o/p → unreported Exception
FNFE

9) import java.io.*;
class Demo{
static void show2() throws FileNotFoundException
{
PrintWriter pw = new PrintWriter("xyz.txt");
}
static void show1() throws FileNotFoundException
{
show2();
}
public static void main(String a[]) throws FileNotFoundException
{
show1();
S.o.p("softwaves_4");
}
}
o/p → softwaves_4

यदि हमारे program में check Exception आती है तो वह compile time पर ही warning के तौर पर display होती है।

Unchecked Exception:- को compiler compile time पर check नहि कर पाता है। इसे JVM cheack करता है।

checked Exception को compiler compile time पर check कर लेता है और warning दे देता है। इसलिए इसे checked Exception को solve करना होता है।

checked Exception: को या तो try catch block से handle कर सकते हैं या throws keyword की Help से exception को throw कर सकते हैं calling method को

Exception propagation / Exception Delegation

When a checked exception is generated in the top of the stack then we have to handle it either by try-catch or we have to throw it from the method using throws keyword.

If we throw the checked exception using throws keyword then it is dropdown to the calling method/previous method

then if it is not handled in the calling method and thrown from the method using throws keyword then it is again dropdown to the previous method and the process is continue until it reach upto the main method.

this process is called exception propagation

10) class Demo {
public static void main(String arg[]){
PrintWriter pw = new PrintWriter ("xyz.txt");
System.out.println("software-4");
}}

o/p → cannot find symbol : PrintWriter

11) import java.io.*;
class Demo{
void show2(){
PrintWriter pw = new PrintWriter("a.txt");}
static void show1(){ show2(); }
public static void main(String arg[]){
show1();
S.O.Pln("software14");
}}

o/p → Error: non static method show2() cannot be reference for static context

difference b/w try, catch & throws keywords Lecture - 10

1)
```
import java.io.*;
class Demo
{   static void show2(){
        PrintWriter pw = new PrintWriter("xyz.txt");
    }
    static void show1(){
        show2();
    }
    public static void main(String arg[])
    {
        show1();
        System.out.println("softwaves_4");
    }
}
```

O/P → Exception: FileNotFoundException ← (show2())

2)
```
import java.io.*;
class Demo{
    static void show2() throws FileNotFoundException
    {
    PrintWriter pw = new PrintWriter("xyz.txt");
    }
    static void show1(){
    } show2();

    public. static void main(String a[]){
        show1();
        S.O.pln("softwaves_4");
    }
}
```

O/P → Exception: FileNotFoundException → (show1())

3) import java.io.*;
class Demo{
static void show2() throws FileNotFoundException
{
PrintWriter pw = new PrintWriter("xyz.txt");
}
static void show1() throws FileNotFoundException
{
show2();
}
public static void main(String ar[])
{
show1();
System.out.println("softwaves-4");
}
}
o/p → Exception : FileNotFoundException - (main())

4) import java.io.*;
class Demo{
static void show2() throws FileNotFoundException
{ PrintWriter pw = new PrintWriter("xyz.txt");
}
static void show1() throws FileNotFoundException
{ show2();
}
public static void main(String ar[]) throws FileNotFound
Exception
{
show1();
System.out.println("softwaves-4");
}
}
o/p → ~~Exception~~ : ~~fil~~ softwaves-4

5) import java.io.*;
```
class Demo {
  static void show2()
  {
    try {
    PrintWriter pw = PrintWriter("xyz.txt");
    }
    catch(FileNotFoundException e)
    {
    }
  }
  public static void show1(){
      show2();
  }
  public static void main(String a[])
  {
     show1();
    System.out.println("Softwaves_444");
  }}
```
o/p → Softwaves_444

try, catch → these are use to handle the exeception

throws → this is used to perogate (drop down) the exception to calling method

→ difference: throws Keyword does not handle the exeception. it only forword the checked exception from that method to the calling method.

but try, catch does not delegate the exception. it can handle both check and unchecked exception.

Note: → throws Keyword only forward the checked exception from the method to calling method not unchecked exception.

difference b/w throw & throws Keyword, Lacture.11

1)
```
class Demo
{
    public static void main(String arg[])
    {
        System.out.println("Softwaves_1");
        int age = Integer.parseInt(arg[0]);
        if(age<18)
        {
            throw new ArithmeticException();
        }
        else
        {
            System.out.println("welcome");
        }
        System.out.println("Softwaves_2");
    }
}
```

o/p → Demo 25
Softwaves_1
→ welcome
Softwaves_2

15
o/p → Softwaves_1
~~welcome~~ Exception : ArithmeticException
~~Softwaves_2~~

if we want to generate any Exception manualy as per our requirements then we have to use throw Keyword.

```
import java.io.*;
```
2)
```
class Demo
{
    public static void main(String arg[])
    {
        System.out.println("Softwaves_1");

        PrintWriter pw = new PrintWriter("xyz.txt");
        System.out.println("Softwaves_2...");
        pw.close();
    }
}
```

o/p → Exception:- FileNotFoundException: must be caugth or declered as thrown

3) import java.io.*;
class Demo{
public static void main(String ar[]) throws FileNotFoundException
{
System.out.println("Softwaves_1");
PrintWriter pw = new PrintWriter("xyz.txt");
System.out.println("Softwaves-2");
pw.close();
}
}
o/p → Softwaves-1
Softwaves-2

throws Keyword is use for throw the exception outside the method. (forword)
→ it only throw the check Exception.

⇒ diff b/w throw & throws

1) ⇒ if we want to generate the checked, unchecked, predefined or user defined, exception manully then we have to use throw Keyword.

throws Keyword is used to forword the checked exception from that method to calling method.

Ex → import java.io.*;
throw → class Demo{
public static void main(String ar[])
{
System.out.println("Softwaves_1");
int age = Integer.parseInt(ar[0]);
if(age<18){
throw new ArithmeticException();
}
else{ System.out.println("welcome"); }
System.out.println("Softwaves_2");
}
}

throws Ex → import java.io.*;
class Demo
{
public static void main(String ar[]) throws FileNotFoundException
{
System.out.println("softwaves-1");
PrintWriter pw = new PrintWriter("xyz.txt");
System.out.println("softwaves-2");
pw.close();
}
}

2) ⇒ throw Keyword is always used inside the method.
throws Keyword is always used with method signature

3) ⇒ When we use throw Keyword then we have to create the object of exception we want to generate manully.
When we use throws Keyword then we use the exception class name.

4) ⇒ throw Keyword can generate only one exception at a time
throws Keyword can throw multiple checked exception at a time

5) →

Can we handle more then one Exception in single ~~else~~ catch block

Lacutre 12

1)
```
import java.io.*;
class Demo{
public static void main(String ar[])
{
int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
System.out.println(x/y);
System.out.println("Softwaves_2..");
}}
```

o/p → case Demo 10 2	10 0	10 abhi
→ 5 Softwaves_2	→ Ex: ArithmeticException: \ by zero	→ Ex: NFE: For input String "abhi"

2)
```
import java.io.*;
class Demo{
public static void main(String ar[])
{
try{
int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
System.out.println(x/y);
}
catch(ArithmeticException e)
{
System.out.println(e);
}
System.out.println("Softwaves_2");
}
}
```

o/p → 10 2	10 0	10 abhi
5 Softwaves2	AE: / by zero Softwaves_2	Excep: NFE: for input String "abhi"

| → Pipe Symbol

3) import java.io.*;
class Demo{
public static void main(String ar[])
{
try{
int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
System.out.println(x/y);
}
catch(ArithmeticException e)
{ S.o.p(e);
}
catch(NumberFormatException e)
{
System.out.println(e);
}
System.out.print("softwaves-2");
}
}

o/p → 10 2
5
softwaves-2

10 0
AE: / by zero
softwaves-2

10 abhi
NFE: for i/p string "abhi"
softwaves-2

4) import java.io.*;
class Demo{
public static void main(String ar[]){
try{ int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
System.out.println(x/y);
}
Catch(ArithmeticException | NumberFormateException e){
S.o.p(e);
} S.o.p("softwaves-222");
}
}

o/p → 10 2
5
softwaves-222

10 0
AE: / by zero
softwaves-222

10 abhi
NFE: for i/p string "abhi"
softwaves-222

Note

1> we can handle multiple exception with one catch block using a pipe(|).

5)
```
import java.io.*;
class Demo{
public static void main(String ar[])
{
try{ int x = Integer.parseInt(ar[0]);
     int y = Integer.parseInt(ar[1]);
     System.out.println(x/y);
   }
catch(ArithmeticException | NumberFormateException e){
     System.out.println(e);
   }
System.out.println("Softwaves_222");
}
}
```

o/p→ 10 2	10 0	10 abhi	10
5	AE: / by zero	NFE: for i/p str "abhi"	Exception: ArrayIndexOutofBounds Exception: 1
Softwaves_222	Softwaves_222	Softwaves_222	

→ Exception class use kre child classes ko all Exception ko handle kr skti h.

6)
```
import java.io.*;
class Demo{
public static void main(String ar[]){
try{ int x = Integer.parseInt(ar[0]);
     int y = Integer.parseInt(ar[1]);
     System.out.println(x/y);
   }
catch(Exception e)
{  S.o.p(e);
}
System.out.println("Softwaves_222");
}
}
```

o/p→ 10 5	10 0	10 abhi	10
2	AE: \ by zero	NFE: for i/p string	AIOOBE: 1
softwaves_222	softwaves_222	"abhi"	softwaves-222
		softwaves_222	

1,7] import java.io.*;
class Demo{
public static void main(String ar[]){
try{ int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
System.out.println(x/y);
}
catch(ArithmeticException e){
System.out.println("abhi");
}
catch(Exception e){ System.out.println("jain"); }
System.out.println("Softwaves-2");
}
}

o/p→ 10 2	10 0	10 abhi	10
5	abhi	jain	jain
softwaves-2	softwaves_2	softwaves-2	softwaves_2

8] import java.io.*;
class Demo{
public static void main(String ar[]){
try{ int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
System.out.println(x/y);
}
catch(Exception e){ System.out.println("jain"); }
catch(ArithmeticException e){ S.o.p("abhi"); }
System.out.println("softwaves-2");
}
}

o/p→ Error: exception ArithmeticException has already been caught

2)
```
import java.io.*;
class Demo{
public static void main(String ar[]){
try{ int x = Integer.parseInt(ar[0]);
     int y = Integer.parseInt(ar[1]);
     System.out.println(x/y);
   }
catch(ArithmeticException e){
  System.out.println("jain");
}
catch(ArithmeticException e){
  System.out.println("abhi");
}
  System.out.println("softwaves_2");
}
}
```

o/p → error: exception ArithmeticException has already been caught

Note

2) when we use multiple catch block with one try block then we can use super class only in last catch block. because if use it any other place then it will give error because super class can handle all the exception.

3)
```
import java.io.*;
class Demo{
public static void main(String ar[]){
try{ int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
  System.out.println(x/y);
}
catch(ArithmeticException/Exception e){
  S.o.pln("jain");
  S.o.pln("softwaves_2");
}
}
```

e = e.toString()

Lacture - 14

o/p → error: Alternatives in a multi-catch statement cannot be related by subclassing.
Alternative ArithmeticException is a subclass of Alternative Exception

Note

3) when we handle multiple exception with one catch block using a pipe(|) then it should not contain any parent-child relationship btween the exceptions.

Lacture 14

Q How many ways to print Exception message

→ 3

1) import java.io.*;
class Demo{
public static void main(String ar[]){
try{
int x = Integer.parseInt(ar[0]);
int y = Integer.parseInt(ar[1]);
System.out.println(x/y);
}
catch(Exception e){ System.out.println("abhi"); } e.toString()

System.out.println("Softwaves_2");
}}

o/p → 10 2 5 Softwaves-2	10 0 main, E: AE: \ by zero Softwaves_2	10 abhi NFE: for I/p St: "abhi" Softwaves_2

toString() → → 10 . 2 5 Softwaves_2	10 0 AE: \ by zero Softwaves_2	10 abhi NFE: for I/p String "abhi" Softwaves_2

diff. ways

1> toString() method: by using this method, we will only get exception name and description of an exception.

2> getMassage() : : : by using this method; we will get description of exception.

3> printStackTrace() : by using this method, we will get exception name and description of an exception separated by colon, and stacktrace in the next line

getmassage() =>

2)
```
import java.io.*;
class Demo{
public static void main(String ar[]){
try{
    int x = Integer.parseInt(ar[0]);
    int y = Integer.parseInt(ar[1]);
      System.out.println(x/y);
  }
catch(Exception e){
    System.out.println(e.getMassage());
  }
System.out.println("softwaves_2");
}
}
```

o/p→ 10 2	10 0	10 abhi
5	\ by zero	for I/P string "abhi"
softwaves_2	softwaves_2	softwaves-2

3) import java.io.*;
class Demo36{
public static void main(String arr[]){
try{
Int x = Integer.parseInt(arr[0]);
Int y = Integer.parseInt(arr[1]);
System.out.println(x/y);
}
catch(Exception e){
~~System.out.println~~ e.printStackTrace();
}
System.out.println("Softwaves_2");
}}

O/P→	10 2	10 0	10 abhi
	5	AE: / by zero	NFE: for i/p string "abhi"
	Softwaves_2	at Demo36.main();	at Demo36.main()
		Softwaves_2	Softwaves_2

4) class Demo37{
public static void main(String arr[]){
try{
System.out.println(10/0);
}
catch(Exception e){
e.printStackTrace();
}
System.out.println("Softwaves_2");
}}

O/P→ ~~E~~ E: AE: / by zero
at Demo37.main()

5] class A{
Void show1(){
try{
System.out.println(10/0);
}
catch(Exception e){
// e.printStackTrace();
System.out.println(e);
}}
}

class B{
Void show2(){
A a = new A(); a.show1();
}}

class Demo38{
public static Void main(String ar()){
B b = new B();
b.show2();
System.out.println("Softwaves_2");
}}

o/p→ E: AE: / by zero
Softwaves_2

6] class A{
Void show1(){
try{
System.out.println(10/0);
}
catch(Exception e){
System.out.println(e.toString());
}}

class B{ Void show2(){ A a = new A();
a.show1();
}
}

```
class Demo{
  public static void main(String ar[])
  {   B b = new B();
        b.show2();
      System.out.println("softwaves_2");
  }}
```

o/p → Arithmetic Exception:
/ by zero
softwaves_2

7] ~~class demo {~~
~~public static void main (String ar[]) {~~

7] class A{
```
  void show1(){
    try{
        System.out.println(10/0);
      }
    catch(Exception e){
        System.out.println(e.getMassage());
  }}  }
```

```
class B{
  void show2(){
    A a = new A();
      a.show1();
  }  }
```

```
class Demo38{
  public static void main(String ar[]){
      B b = new B();
        b.show2();
      System.out.println("softwaves_2");
  }}
```

o/p → / by zero
softwaves_2

```
class A{
    void show1(){
        try{
            System.out.println(10/0);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }}
class B{
    void show2(){
        A a = new A();
        a.show1();
    }
}

class Demo38{
    public static void main(String arg[]){
        B b = new B();
        b.show2();
    }
}
```

O/P → java.lang.ArithmeticException : / by zero
at A.show1()
at B.show2()
at Demo38.main()

softwaves - 2

⇒ जब JVM Exception को main() method से Default Exception Handler को pass करता है तो DefaultExceptionHandler internally printStackTrace() method का use करता है।
इसलिए यह stack की form में print होता है।
message

fully checked Exception

जिसकी Exception class की सभी sub classess में checkedException ही हो तो ऐसी class को fully checkedException कहते है।

ex → IOException

partially checked Exception

the exception which have both checked and unchecked subclasses is called partially checked exception.

ex → Throwable
→ Exception