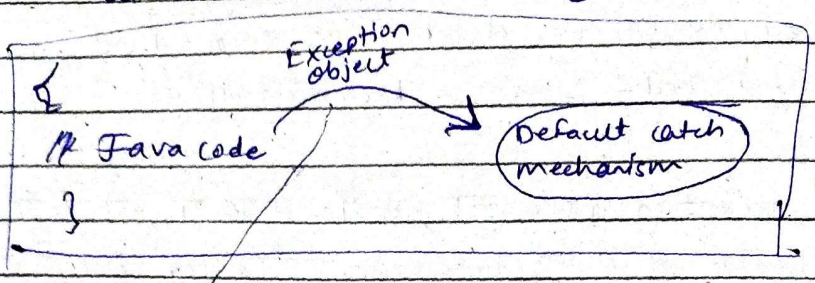
(Youtube - Lecture 32 Intro. to exception handling → channel - Java by Saurabh Shukla Sir)

Page No.:
Date:

Exception handling in java

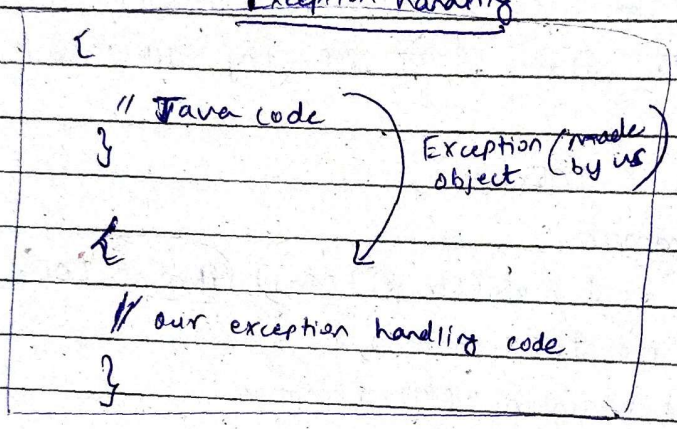
- at runtime

predefined situations in Java considered as exception



This object is by default ~~made~~ thrown by Java. This object describes the ~~error~~ exception.

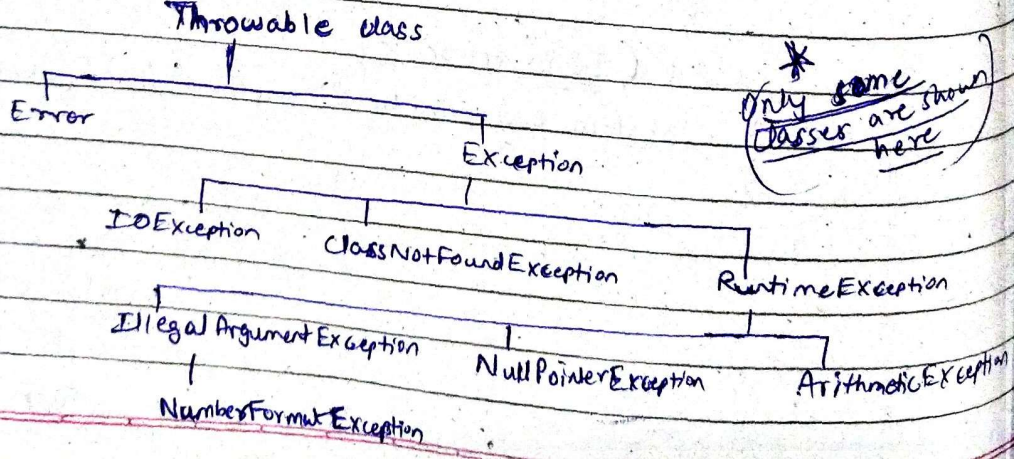
Exception handling



4 Options -

(i) default throw & default catch
(ii) default throw & our catch
(iii) our throw & default catch
(iv) our throw & our catch

default catch झाला तर program end होईल.



Page No.:
Date: / /

Java exceptions are raised with throw keyword and handled within a catch block.

```
String s1 = null;
S.O.ut (s1.length());
```
↑ null pointer exception

s1 is a reference variable and it's null. So, it's not pointing to object.

- The class Throwable provides getMessage() function to retrieve an exception.
- Throwable class provides a string variable that can be set by the subclasses to provide a detailed message that provides more information of the exception occured.

Unchecked exceptions – Runtime exception.
– Subclasses of RuntimeException

Default throw and our catch

```
try
{
    <code>
} catch(<exception type> <parameter>){

}
finally {

}
```

- After try block, catch block or finally block should be written.
- multiple catches are allowed, but only one finally.

Page No.:
Date. / /

Default throw and our catch

```
class Example{
  public static void main(String[]args){
    try{
      System.out.println(3/0);
      System.out.println("Entry");
    }
    catch(Exception ArithmeticException e){
      System.out.println("Exception:" + e.getMessage());
    }
    System.out.println("Hello");
  }
}
```

If we would have written some other exception like ArrayIndex..., then Java's default catch would have worked

① ~~If catch doesn't~~ If the catch block that we've written, is not ~~handling the i/o~~ written for the correct exception class,

→ try works, if exception comes, then finally works, after finally, java's default catch mechanism works

② Even if there's no exception "try", "finally" will work.

Previous lectures are on [illegible] pages ahead Date.

Lecture 34 . Java Saurabh Shukla

Exception handling

throw <throwableInstance>;

> The exception reference must be of type throwable class or one of its subclasses.

> A detailed message can be passed to the constructor when the exception object is created.

Our throw default catch

```
class ---{
    p s v m(){
        int balance = 5000;
        int withdrawlAmount = 6000;
        if( balance < withdrawlAmount)
            throw new ArithmeticException("Insufficient balance");
        balance = balance - withdrawlAmount ;
    }
}
```

O/P:- Exception in thread "main" java.lang.ArithmeticException: Insufficient balance
at Example.main(Example.java:7)

Our throw our catch

```
class {
    p.s.v. m. () {
        int balance = ___ ;
        int withdrawlAmount = ___ ;
        try {
            throw new ArithmeticException("Insufficient balance");
        }
        catch( ArithmeticException e)
        {   s.o.ut ("Exception:" + e.getMessage());
        }
        System.out.println("program continued");
    }
}
```

Lecture 35 Use of throws in checked exception in java

checked exception - detected at compile time
Unchecked - compiler check करत नाही.

checked exception java ला handle करायला सांगायचं असेल तर, throws वापरणे; otherwise स्वतः try catch लिहून handle करणे.

checked exception मध्ये direct throw लिहू नये. त्याने error येते. throws वापरून throw करावे OR try catch throw करावे.

①
```
import java.io.IOException;
public class Example
{
public static void main(String[] args) throws IOException
{   throw new IOException();
    System.out.println("After Exception");
}
}
```

comma करून multiple classes लिहू शकतो.

Method() throws <ExceptionType1>, ----- <ExceptionTypeN>

②
```
class {
  p.s.v.m.() {
    try {
      throw new IOException();
    }
    catch (IOException e)
    {  System.out.println("Exception:" + e.getMessage());
    }
  }
}
```

File class in java.io package

```
import java.io.File;
File file = new File("C:\\data\\input-file.txt");
```

file.exists() - whether a file exists

file.getName() - returns the name of the file

There're a number of ways to read from a file.

1. Using Scanner class.

To read the contents of file, file object is passed to Scanner object.

```
try {
    File x = new File("path.txt");
    Scanner sc = new Scanner(x);
}
catch (FileNotFoundException e) {
}
```

try/catch block - If there's a chance of non-existent file.

```
while (sc.hasNext()){
    System.out.println(sc.next());
}
sc.close();
```

Scanner class inherits from Iterator.

next() method returns each word separately

close() - to close a file when finished working with it.

Formatter - used to create content and write it to files

```
import java.util.Formatter;

class {
    p.s.v. main(String[] args)
{

    try {
        Formatter f = new Formatter("path.txt");
        f.format("%s %s %s", "1", "John", "Smith \r\n");
        f.close();
    }
    catch (Exception e){
        s.out("error");
    }
}
}
```

file is created

\r\n is the newline symbol in Windows