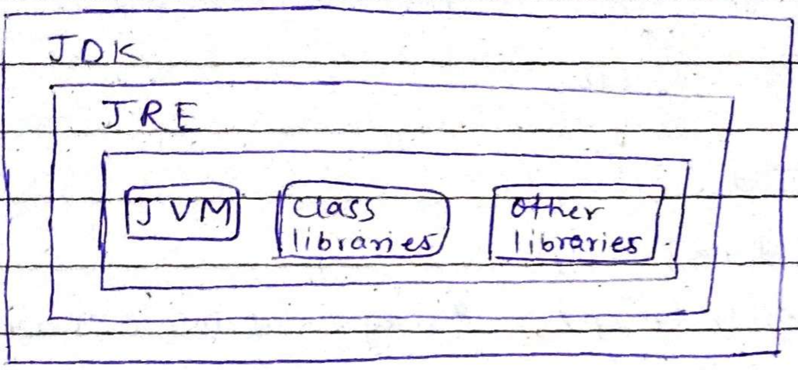
① To install Java on computer, the developer must download the JDK and set up the JRE.

② JDK - technically an implementation of either Java Standard edition or Java enterprise edition
- software development tools & supporting libraries



③ JVM - software tool responsible for creating run-time environment for the java source code to run.
- stays right on top of the host OS & converts the java source code into ByteCode.

④ JRE - software platform where all the java source codes are executed
- responsible for integrating the software plugins, jar files, and support libraries necessary for the source code to run.

⑤ Java APIs - integrated pieces of software that come with JDKs
- provide interface between two different applications & establish communication

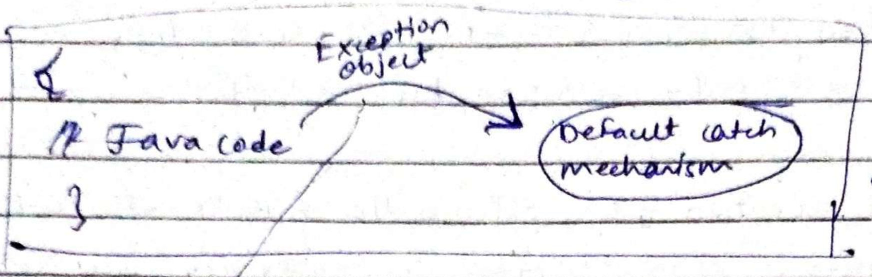
⑥ Different APIs have different service protocols
- rules & protocols guide the functionality of the Java API.

e.g. Rules of RESTful API service protocol →
Stateless, Uniform Interface, Client-server, cache, layered

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

Exception handling in java
- at runtime

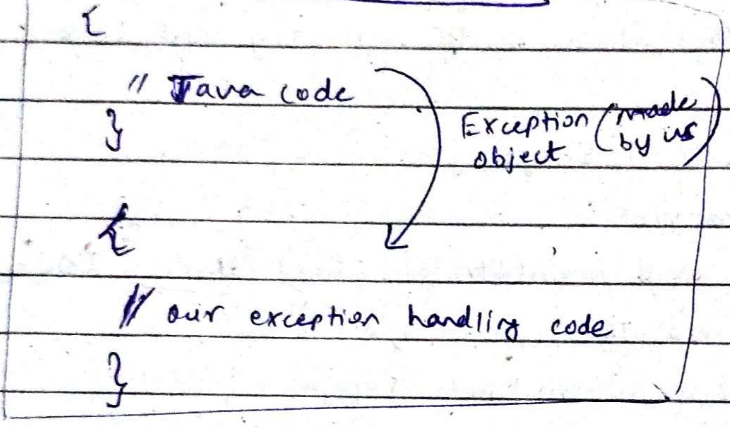
Predefined situations in Java considered as exception



This happens without exception handling by ourselves.
← by default happens in java

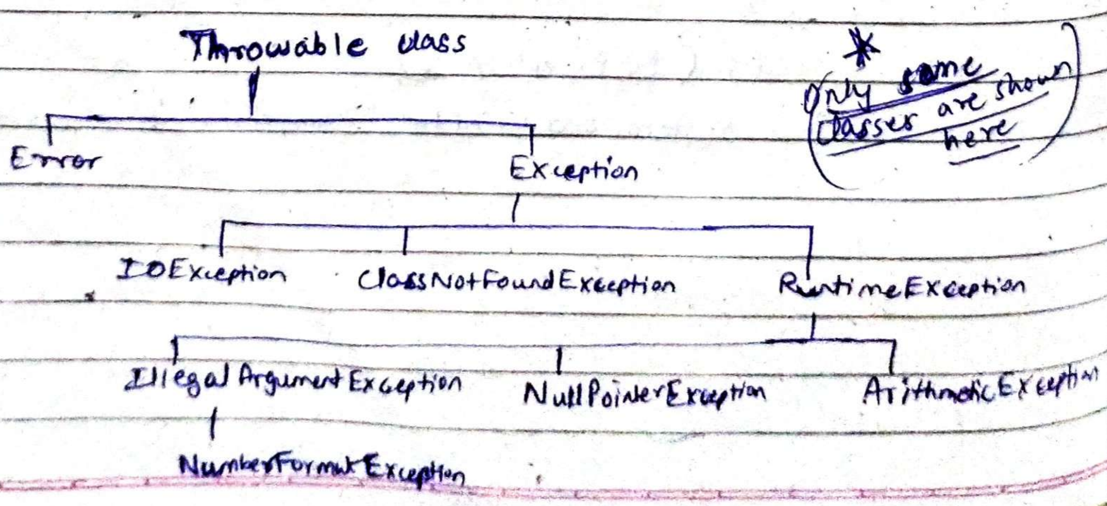
This object is by default ~~made~~ thrown by java. This object describes the exception.

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.



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

Date. / /

Default throw and our catch

```
class Example {
    public static void main(String[] args){
      try {
        System.out.println(3/0);
        System.out.println("Intry");
      }
      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 doe~~ ~~If~~ the catch block that we've written, is not ~~handling the ri~~ written for the correct exception class,

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

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

Lecture 34 · Java Saurabh Shukla

Exception handling

throw <throwable Instance>;

> 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 withdrawl Amount = 6000;
        if( balance < withdrawlAmount)
            throw new ArithmeticException("Insufficient balance");
        balance = balance - withdrawl Amount ;
    }
}
```

```
O/P:- Exception in thread "main" java.lang.ArithmeticException; Insufficient
      balance
      at Example.main
      <Example.java>
```

Our throw our catch

```
class  {
    p.s.v. m. () {
        int balance = __ ;
        int withdrawl Amount = __ ;
    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 (IO Exception e)
      {  System.out.println("Exception:" + e.getMessage());
      }
    }
}
```