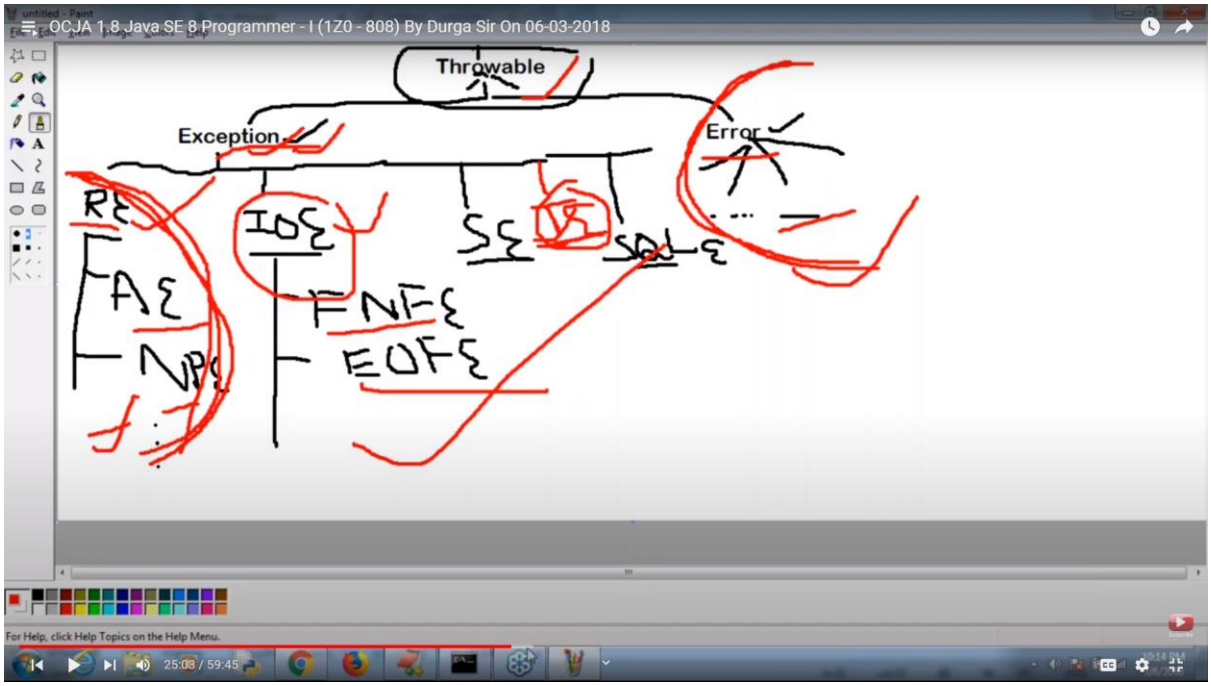
Exception Handling Root Class – Throwable

Throwable Child class are Exception and Error (Stack OverFlow Error)



Exception – Two Types Checked and Unchecked Exception – Needed for Overriding

RE and its child classes. Error and child class are Unchecked Exception

Rest are Checked Exception

Rule1: For Checked Exception

```
54 If child class method throws any checked exception then compulsory parent class
55 method should throw the same checked exception or its parent.
56 CE
57
```

For UnChecked Exception – No Rule

Child class does not throw an any exception – So its valid

Child throws Exception which is checked then Parent class method should also throw the same exception or parent exception- Below is Invalid. Overriden method(Parent Method) does not throw Exception

```
34 P: public void m1()
35 C: public void m1() throws Exception
36 ------------------------------------------------
```

Child Throws IOException parent throws parent of IOException->Exception valie

```
37 P: public void m1() throws Exception
38 C: public void m1() throws IOException
```

Child throws Exeption but parent throws child of Exception (Allowed Exception or its parent throwable only)- Fail

```
40 P: public void m1() throws IOException
41 C: public void m1() throws Exception
42 ------------------------------------------------
```

FileNotFound and EOFException are child of IOException - Valid

```
42 ------------------------------------------------
43 P: public void m1() throws IOException
44 C: public void m1() throws FileNotFoundException,EOFException
```

FileNotFoundException is a child of IOException , InterruptedException is not a child of IOException. Exception is the parent class

```
45 ------------------------------------------------------------------
46 P: public void m1() throws IOException
47 C: public void m1() throws FileNotFoundException,InterruptedException
48 ------------------------------------------------------------
```

Valid. Child classes are unchecked Exceptions, child method throws checked then the rule applies .

For unchecked exception no rules.

```
48 ------------------------------------------------------------
49 P: public void m1() throws IOException
50 C: public void m1() throws AE,NPE
```

```java
import java.io.*;
class P
{
   public void m1() throws IOException
   {
   }
}
class C extends P
{
   public void m1() throws ArithmeticException,FileNotFoundException
   {
   }
}
```

```
61 ------------------------------------------------------------
62 overriding wrt static modifier:
63 ------------------------------
```

Static Methods are class level methods, Instance methods are Object Level methods. So, we can't override static and non-static methods and vice versa. Static method can be overridden with static method only and Instance method with instance method only

CE. How static method with non static

```java
class P
{
   public static void m1()
   {
      System.out.println("Parent");
   }
}
class C extends P
{
   public void m1()
   {
      System.out.println("Child");
   }
}
```

```
D:\durgaclasses>javac P.java
P.java:10: error: m1() in C cannot override m1() in P
       public void m1()
                   ^
  overridden method is static
1 error
```

Invalid

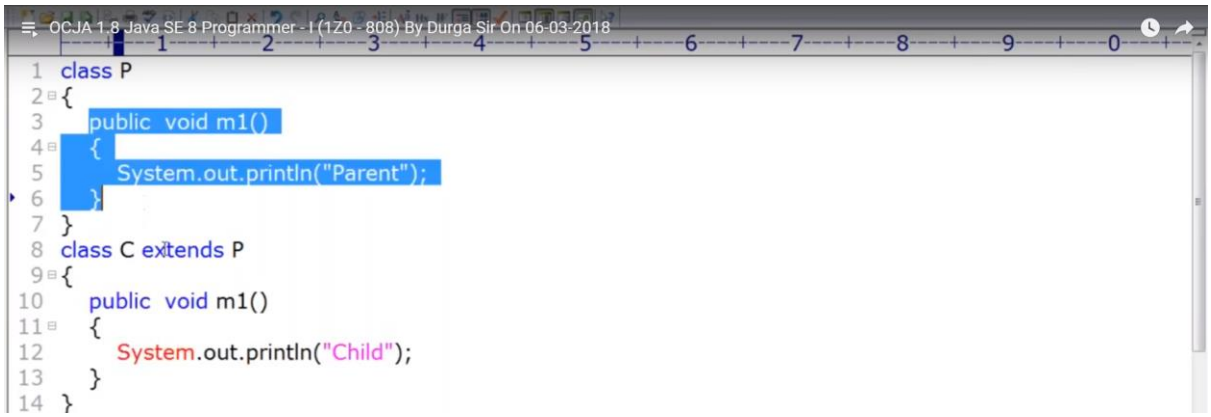
```
class P
{
   public  void m1()
   {
      System.out.println("Parent");
   }
}
class C extends P
{
   public static void m1()
   {
      System.out.println("Child");
   }
}
```

```
D:\durgaclasses>javac P.java
P.java:10: error: m1() in C cannot override m1() in P
       public static void m1()
                          ^
  overriding method is static
```

Static Method Overloading is not called overloading but Method Hiding

Method Hiding. Resolution is carried by ref type by compiler and compiler calls the ref var method

Consider non static methods below , this is overriding



```
15 class Test
16 {
17    public static void main(String[] args)
18    {
19       P p = new P();
20       p.m1();
21
22       C c = new C();
23       c.m1();
24
25       P p1 = new C();
26       p1.m1();
27
28    }
```

```
D:\durgaclasses>java Test
Parent
Child
Child
```

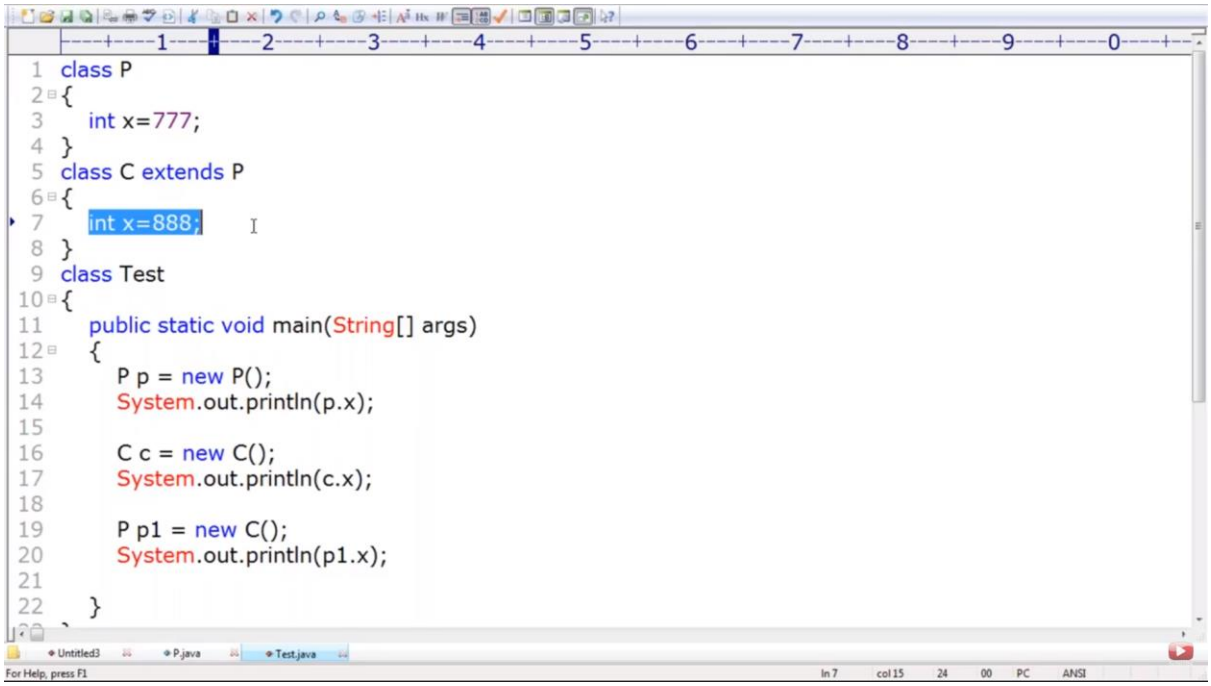
Static method Method Hiding carried out by compiler calls Ref Var Methods

```
1  class P
2  {
3     public static void m1()
4     {
5        System.out.println("Parent");
6     }
7  }
8  class C extends P
9  {
10    public static void m1()
11    {
12       System.out.println("Child");
13    }
14 }
```

```
15 class Test
16 {
17    public static void main(String[] args)
18    {
19       P p = new P();
20       p.m1();
21
22       C c = new C();
23       c.m1();
24
25       P p1 = new C();
26       p1.m1();
27
28    }
```

```
D:\durgaclasses>java Test
Parent
Child
Parent
```

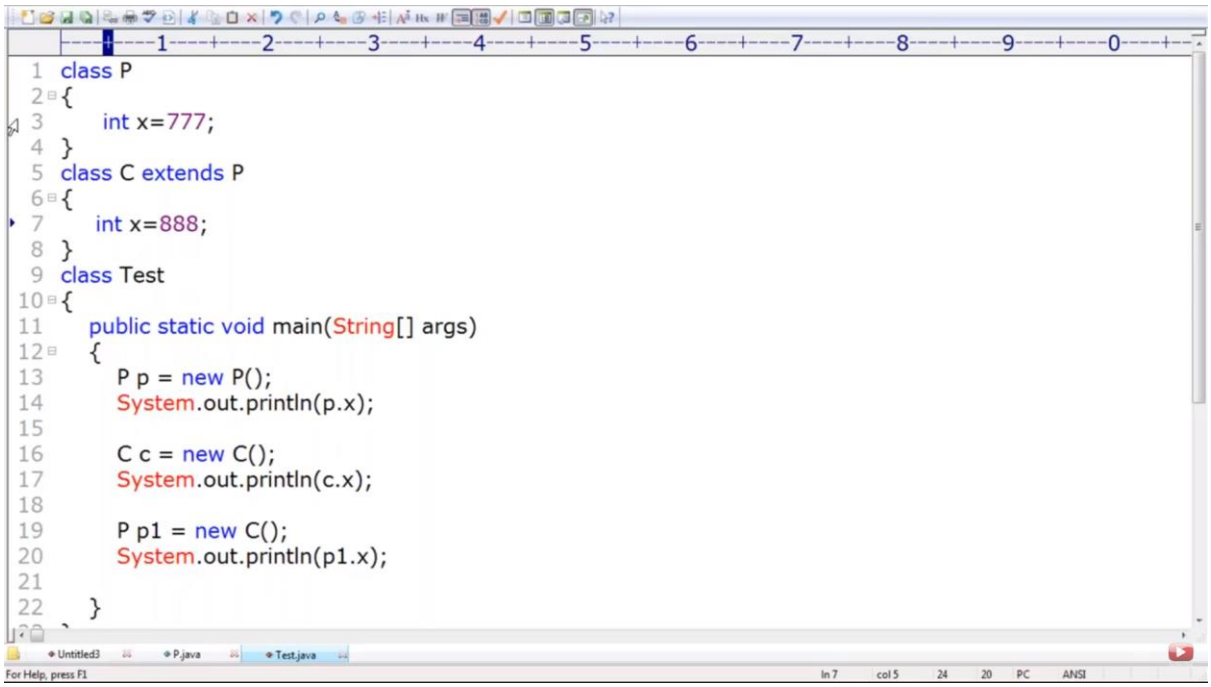
Overriding concept is applicable only for methods and not variables, How to resolve



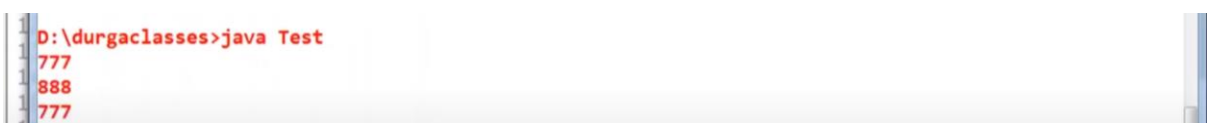
```
class P
{
   int x=777;
}
class C extends P
{
   int x=888;
}
class Test
{
   public static void main(String[] args)
   {
      P p = new P();
      System.out.println(p.x);

      C c = new C();
      System.out.println(c.x);

      P p1 = new C();
      System.out.println(p1.x);

   }
```

Variable resolution is always taken care by compiler based on ref type. JVM does not play a role here.

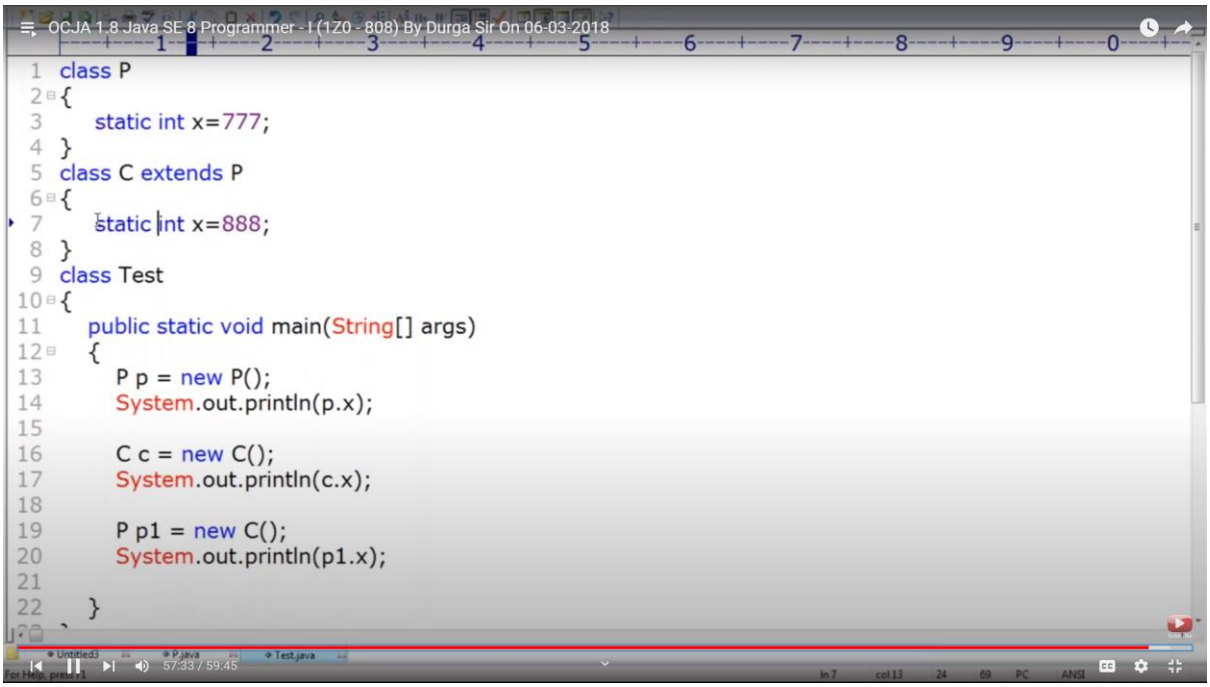
Rule is same if its static variable non static or not- based only on ref variable



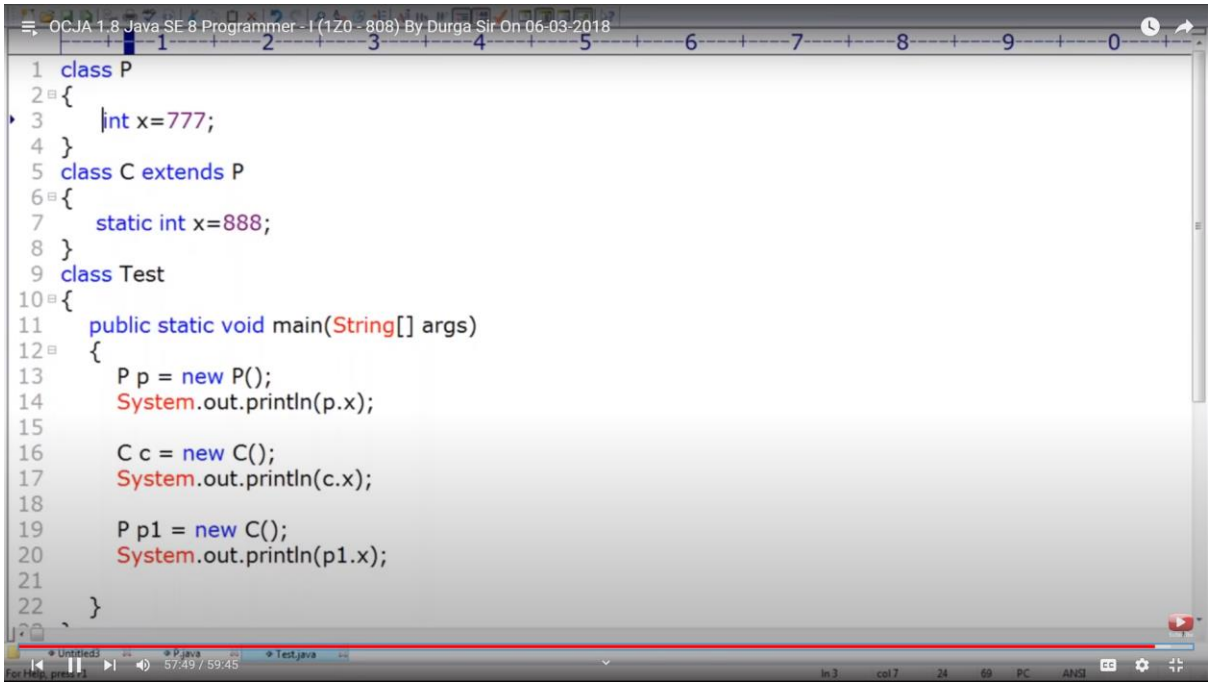
```
class P
{
    int x=777;
}
class C extends P
{
    int x=888;
}
class Test
{
   public static void main(String[] args)
   {
      P p = new P();
      System.out.println(p.x);

      C c = new C();
      System.out.println(c.x);

      P p1 = new C();
      System.out.println(p1.x);

   }
```



```
D:\durgaclasses>java Test
777
888
777
```




```
D:\durgaclasses>java Test
777
888
777
```



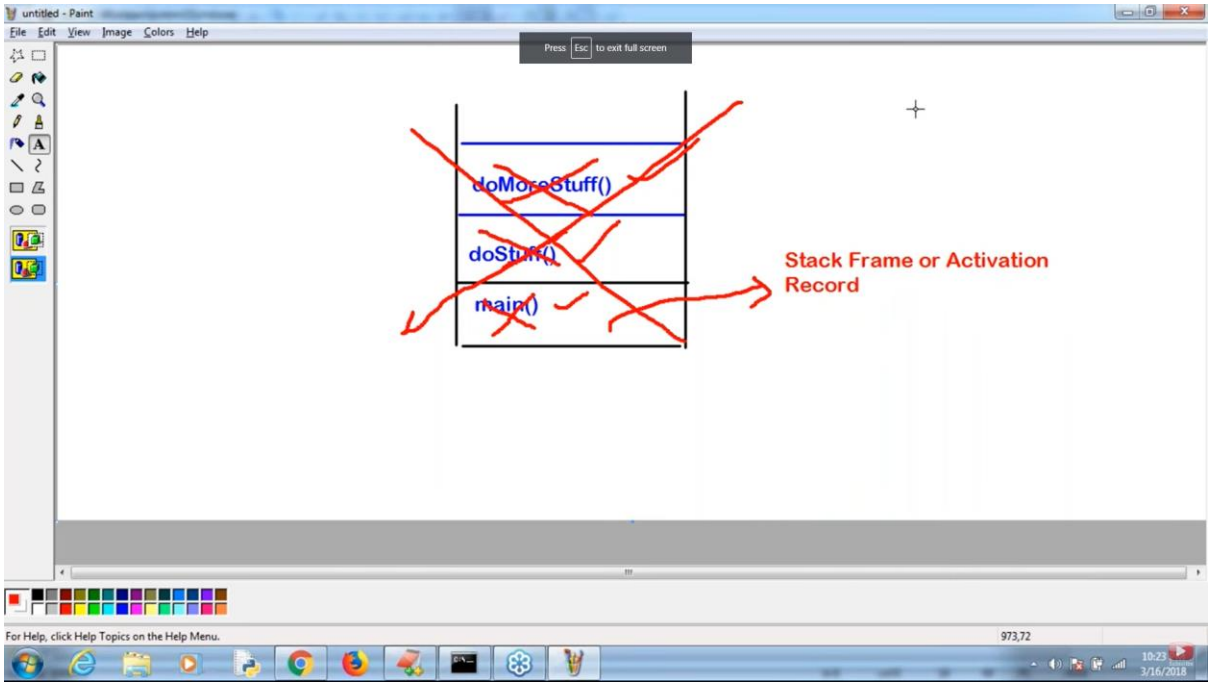

```
D:\durgaclasses>java Test
777
888
777
```

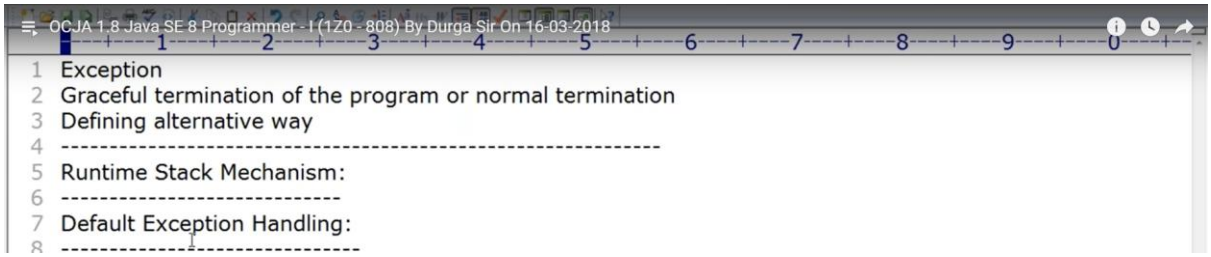
Exception

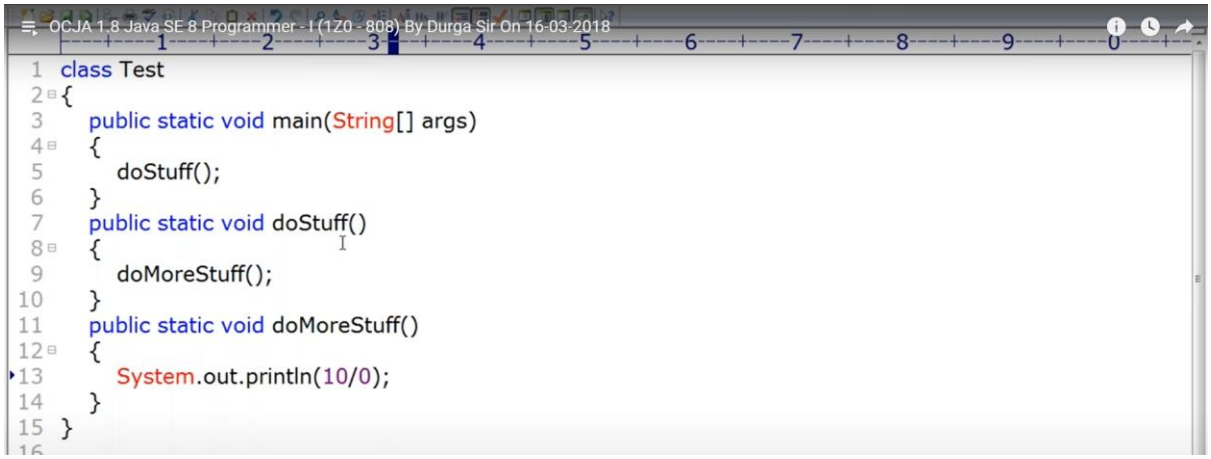
Every Thread has a stack

After main method is completed the, main thread is done and the stack will be destroyed

Every stack is called a Stack Frame or Activation Record







If DoMoreStuff fails what happens is defaultExecptionMechanism. It creates an Exception object as shown below and JVM checks if there is a handling code

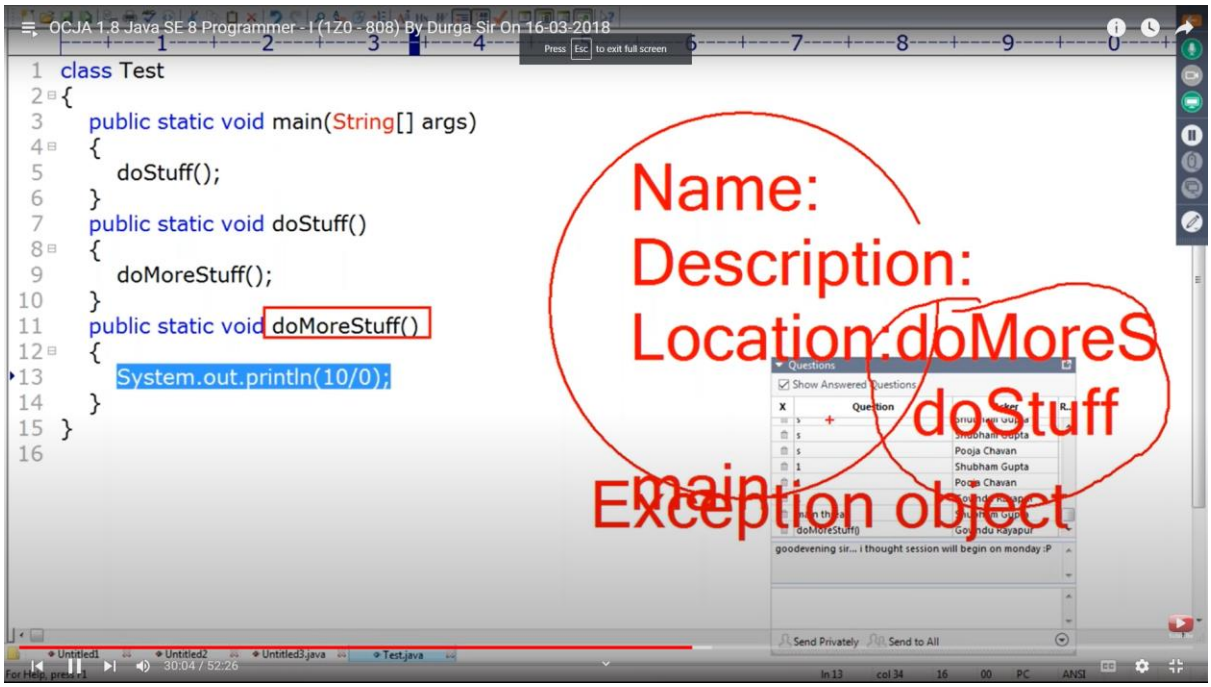
It checks if there is an handling code if not terminate the method.

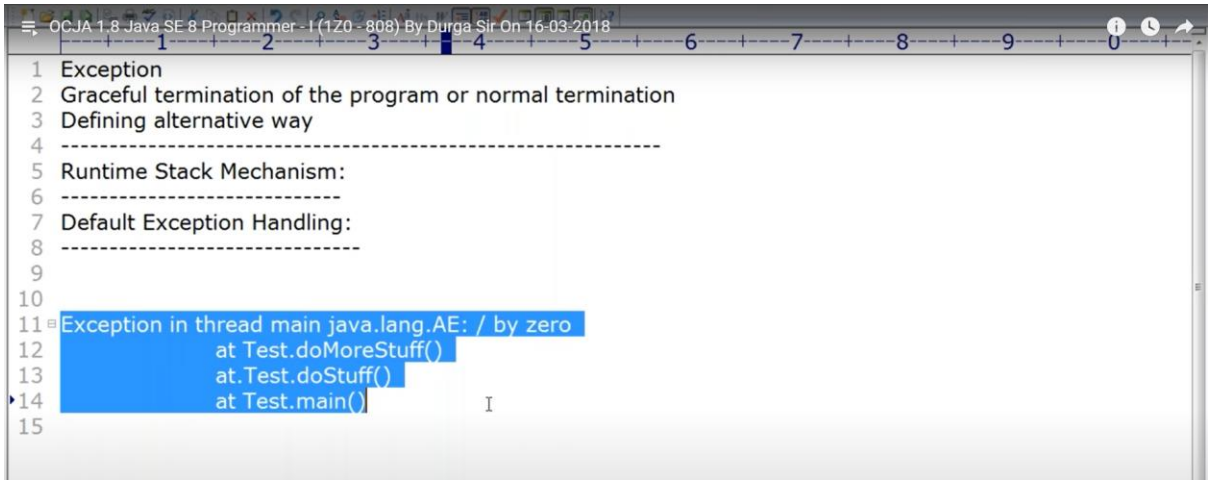
Checks who is the caller- Do u have an handling code ? Do stuff has no – Terminate DoStuff

At last

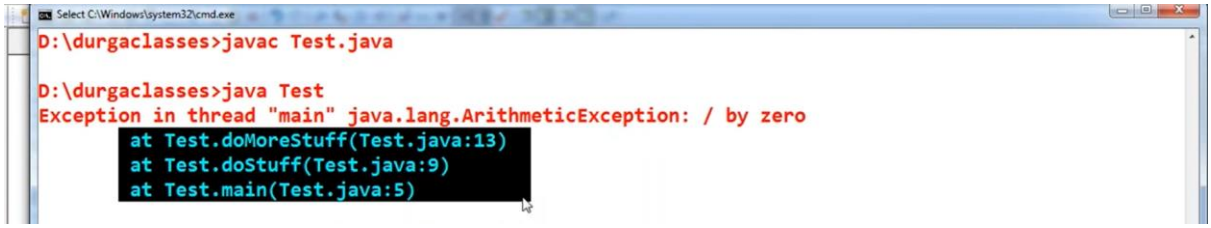
JVM checks if MainMethod , do you have any handling code? No -> Terminate Main Method

Caller of JVM method – JVM handles the exception -> DefaultExceptionHandler. -> Terminates the program abnormally.



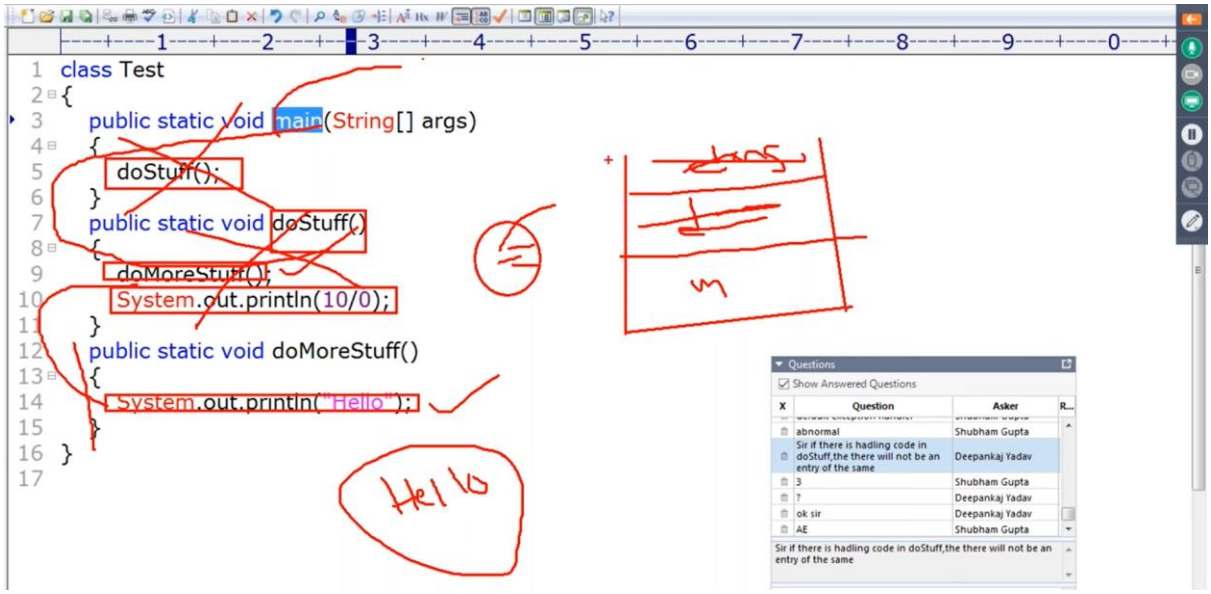


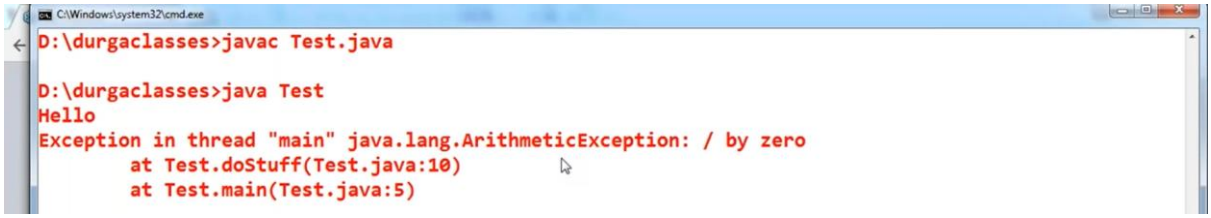
This is called Stack Trace, taken care by DefaultExceptionHandler



Here DomoreStuff executes successfully and doStuff line 10 will Throw Exception

At Test doMoreStuff will not be available in StackTrace





Error and child classes and Runtime Exception and child classes are Unchecked . Compiler is not going to throw an error.

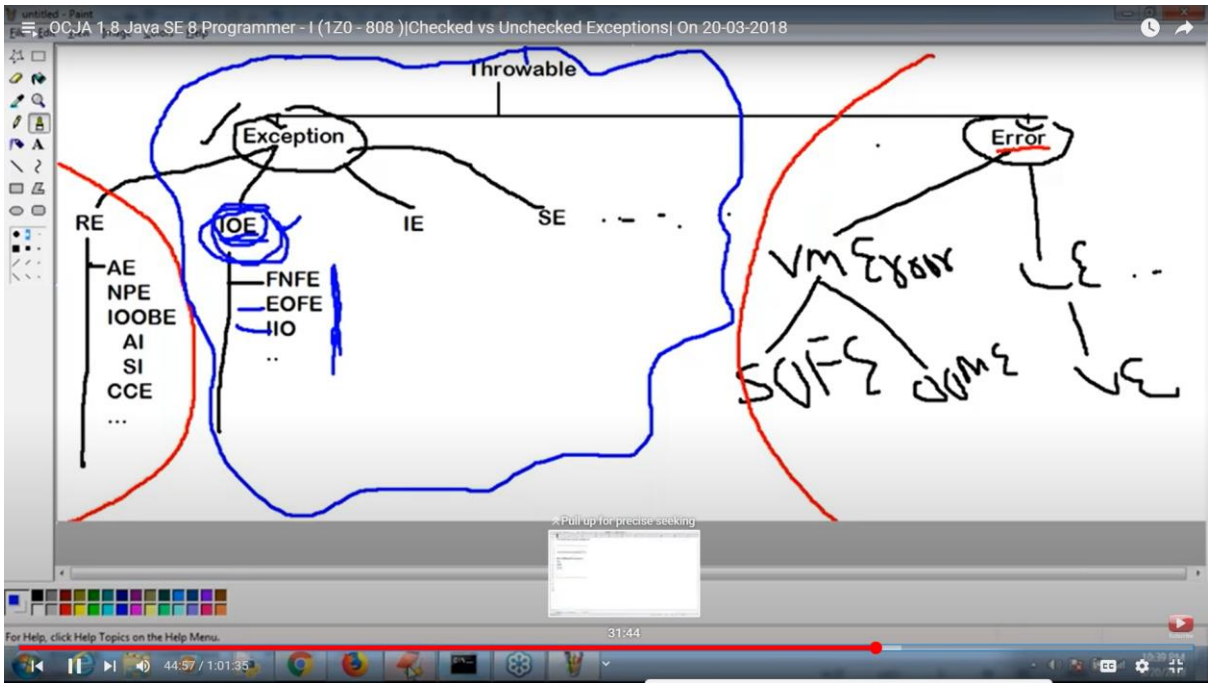
```
22 -------------------------
23 RE and its child classes
24 Error and its child classes
25 unchecked
26
27 FNFE===>checked
28 IOE===>checked
29 Exception==>checked
30 AE===>unchecked
31 RE==>unchecked
32 Object==>NA
33
```

Fully Checked and Partially Checked. Parent is checked and all childs are checked. Eg IOE, IE(interuppted).

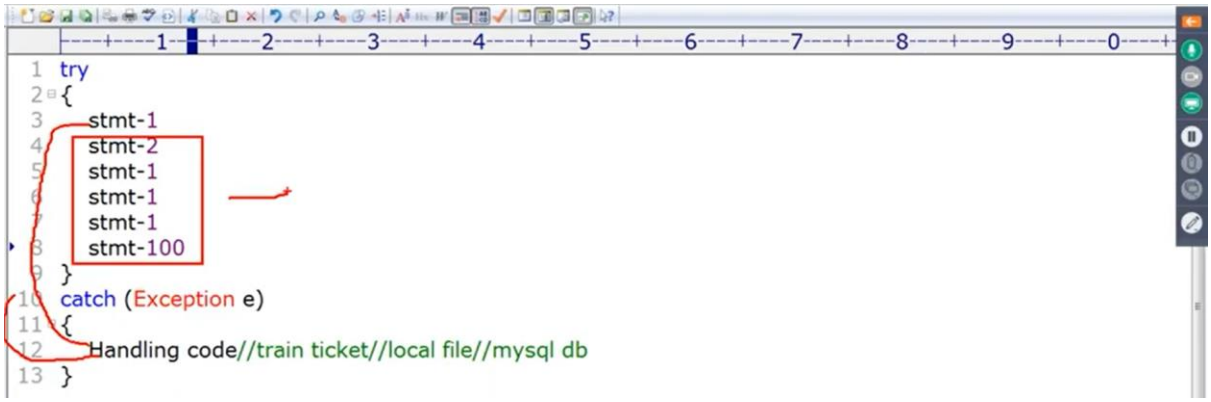
Exception class has RE and IOE. RE is unchecked and IOE is checked hence its partially checked

Throwable is partially Checked.Errors unchecked and RE is also unchecked.

Error is completely Unchecked Exception. Other than Throwable and Exception no other partially checked



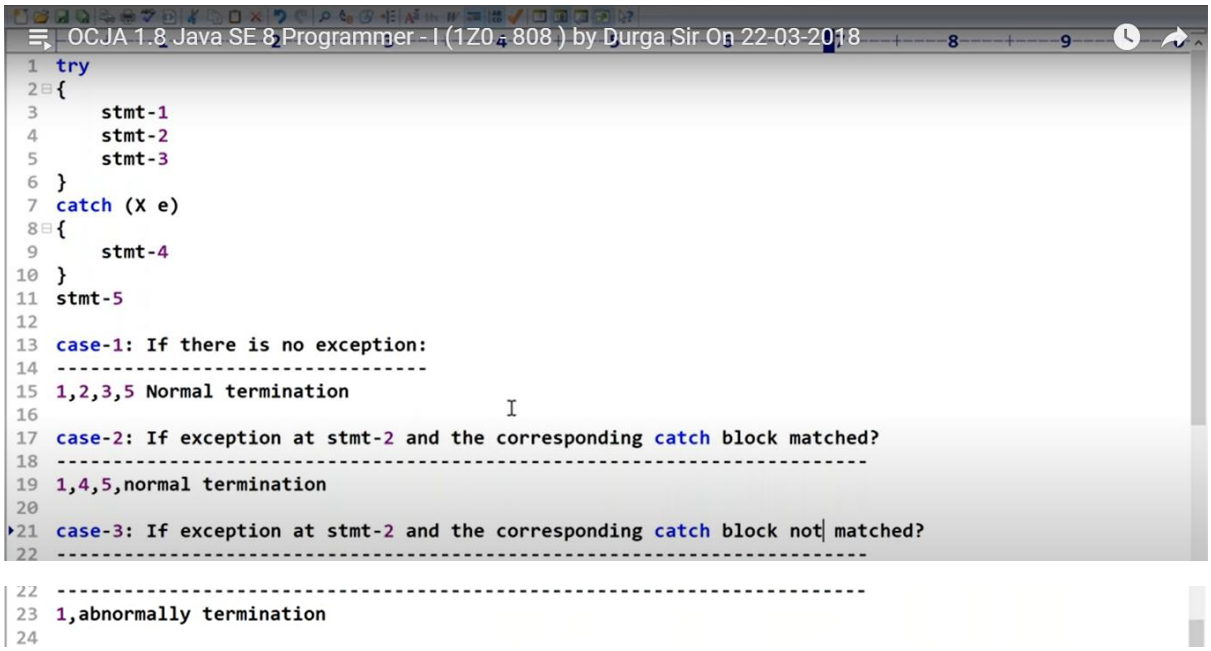
Not recommeded to write all code in one try block. Will stop all execution



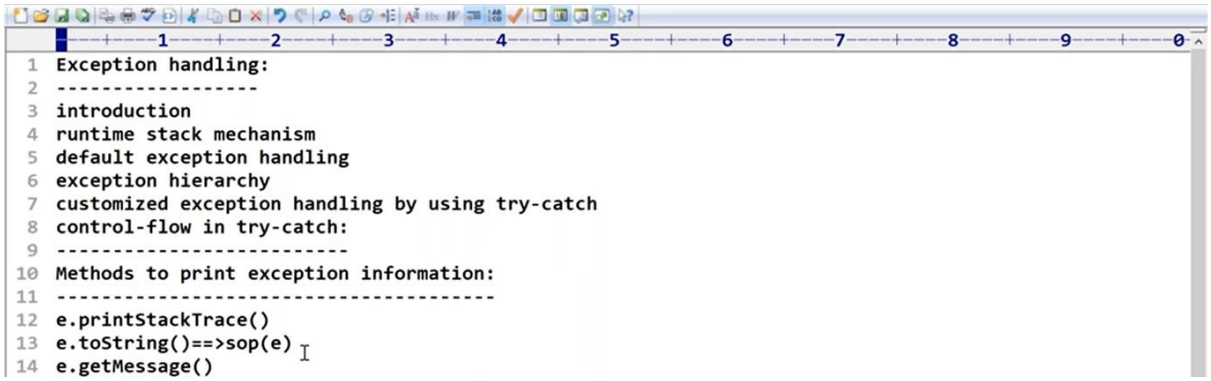
```
try
{
    stmt-1
    stmt-2
    stmt-1
    stmt-1
    stmt-1
    stmt-100
}
catch (Exception e)
{
    Handling code//train ticket//local file//mysql db
}
```

Exceptions are caused by program and recoverable

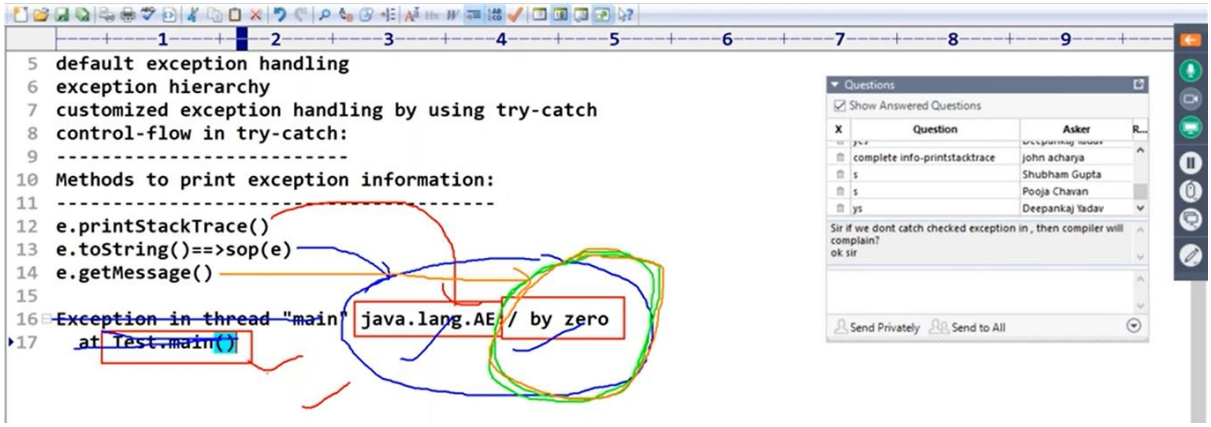
Errors are not caused by program and not recoverable




```
try
{
    stmt-1
    stmt-2
    stmt-3
}
catch (X e)
{
    stmt-4
}
stmt-5

case-1: If there is no exception:
---------------------------------
1,2,3,5 Normal termination

case-2: If exception at stmt-2 and the corresponding catch block matched?
------------------------------------------------------------------------
1,4,5,normal termination

case-3: If exception at stmt-2 and the corresponding catch block not matched?
------------------------------------------------------------------------
------------------------------------------------------------------------
1,abnormally termination
```



```
Exception handling:
-------------------
introduction
runtime stack mechanism
default exception handling
exception hierarchy
customized exception handling by using try-catch
control-flow in try-catch:
--------------------------
Methods to print exception information:
---------------------------------------
e.printStackTrace()
e.toString()==>sop(e)
e.getMessage()
```

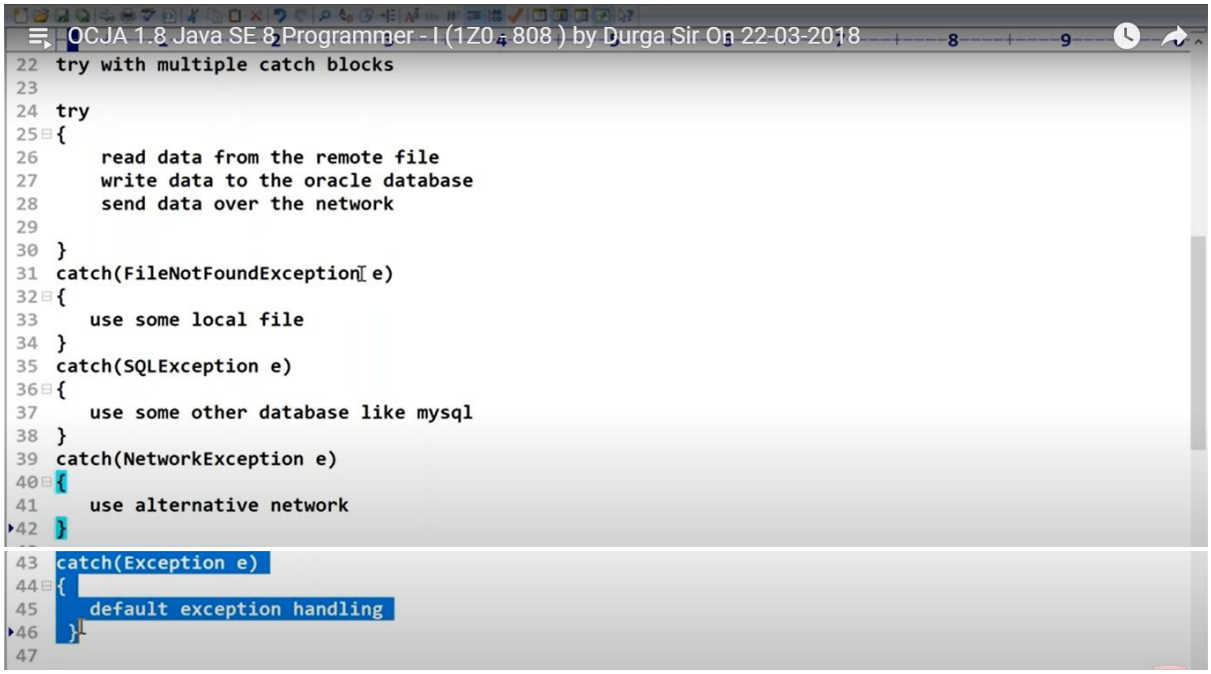


e.printStackTrace()

e.toString()

sysout(e)-> will call e.toString()

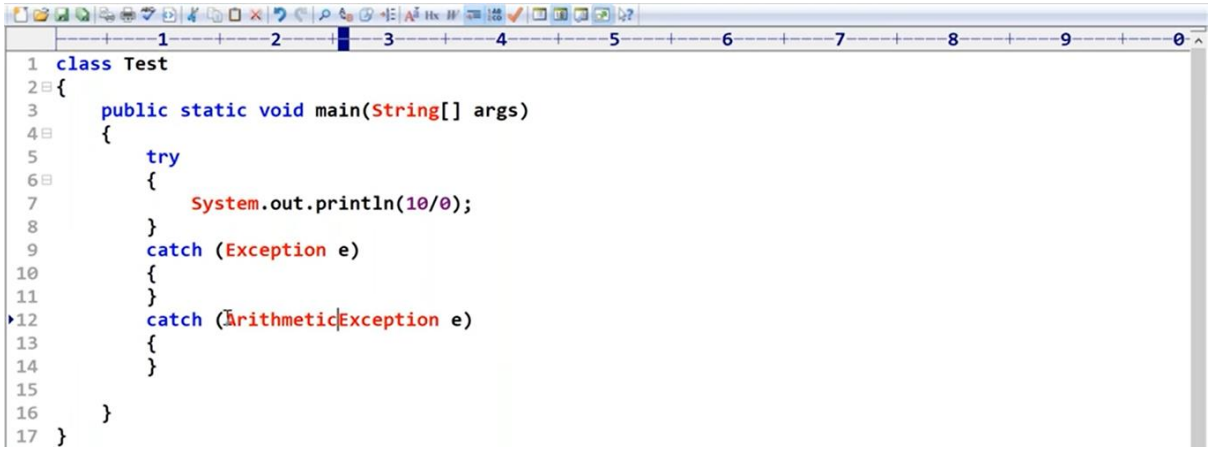
e.getMessage()



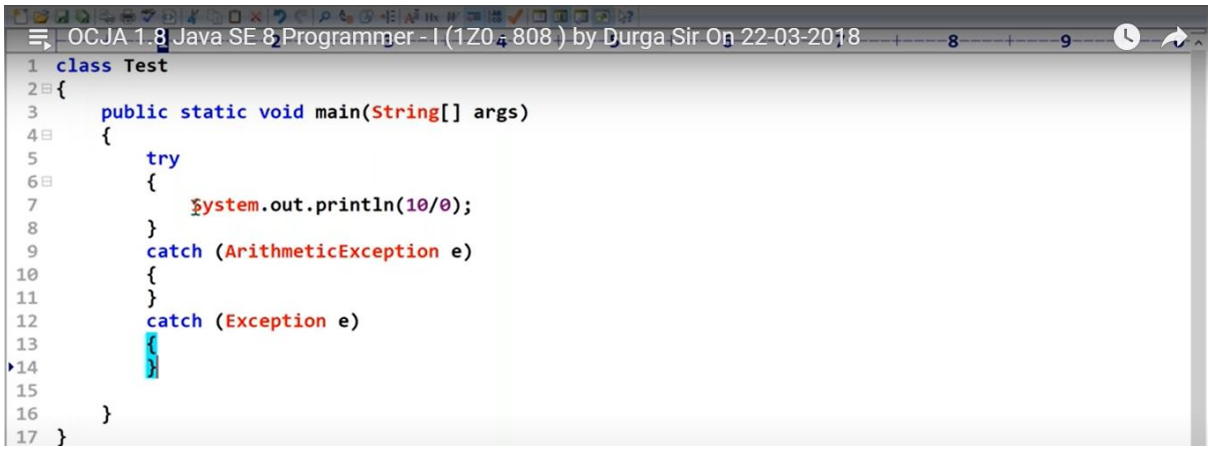
Don’t use throws unchecked exception -> Throws keyword is used for checked exception. You can use Throws for unchecked but still makes no sense.

Order of catch block is important

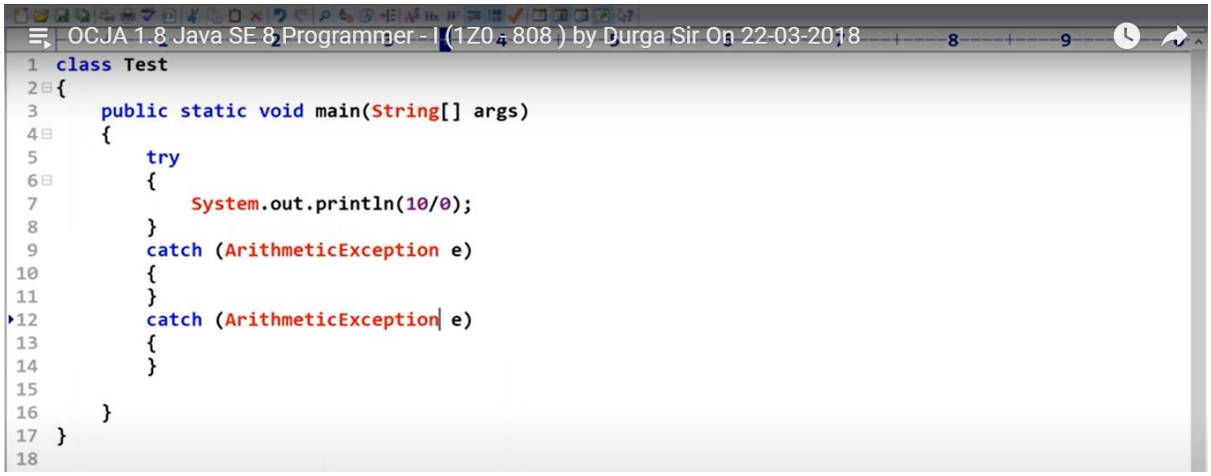
JVM considers top to botton. Exception e -> will handle all exception and will not handle arithmetic exception. First child followed by Parent

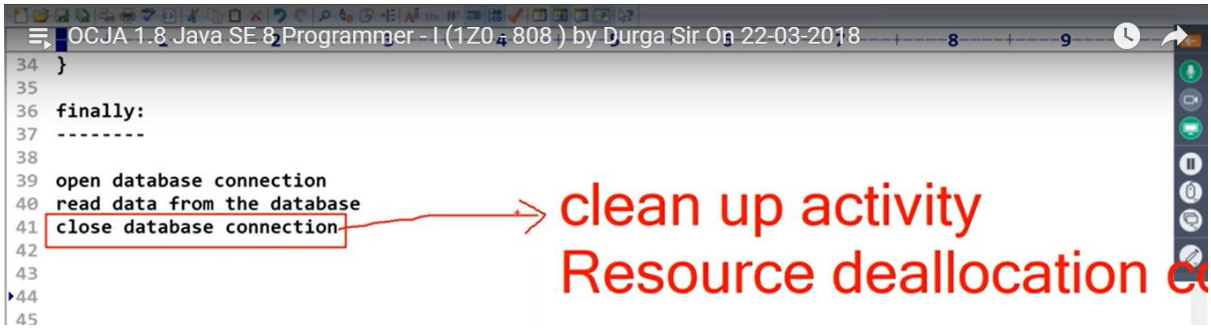


Below is valid

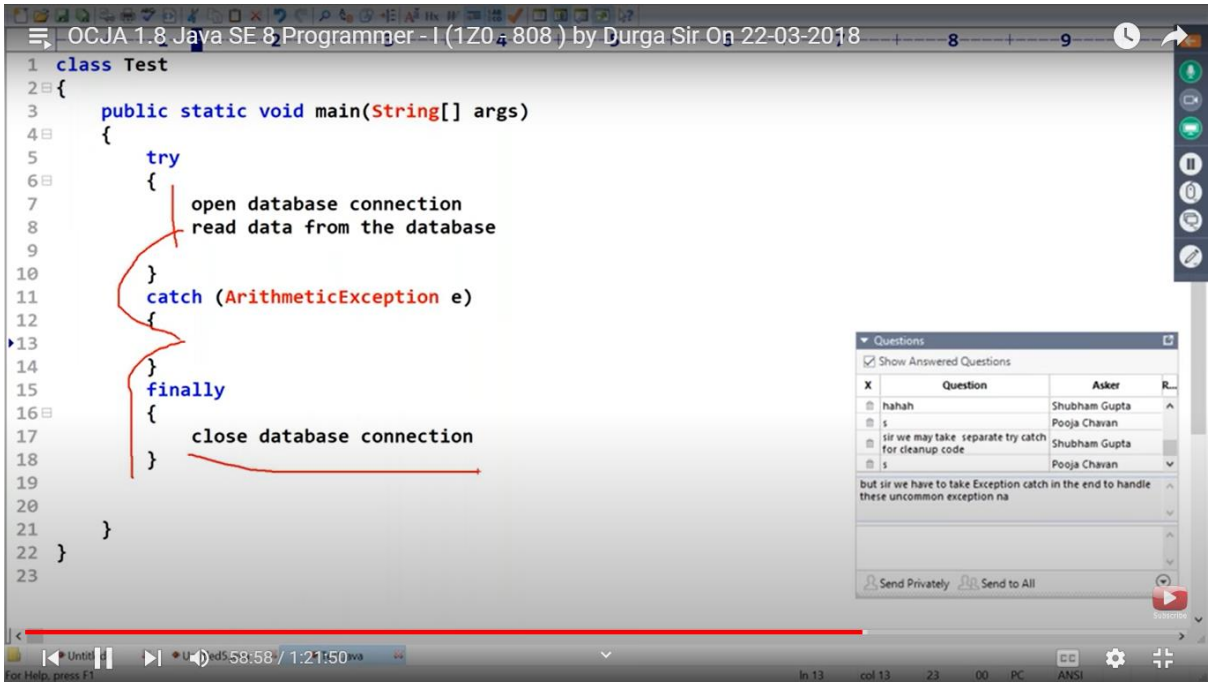


InValid





Finally will be executed during normal termination. Even if exception is handled or if exception is not handled and abnormal termination happens, finally block will be executed. Highly recommended



Only 1 finally for a try block is allowed

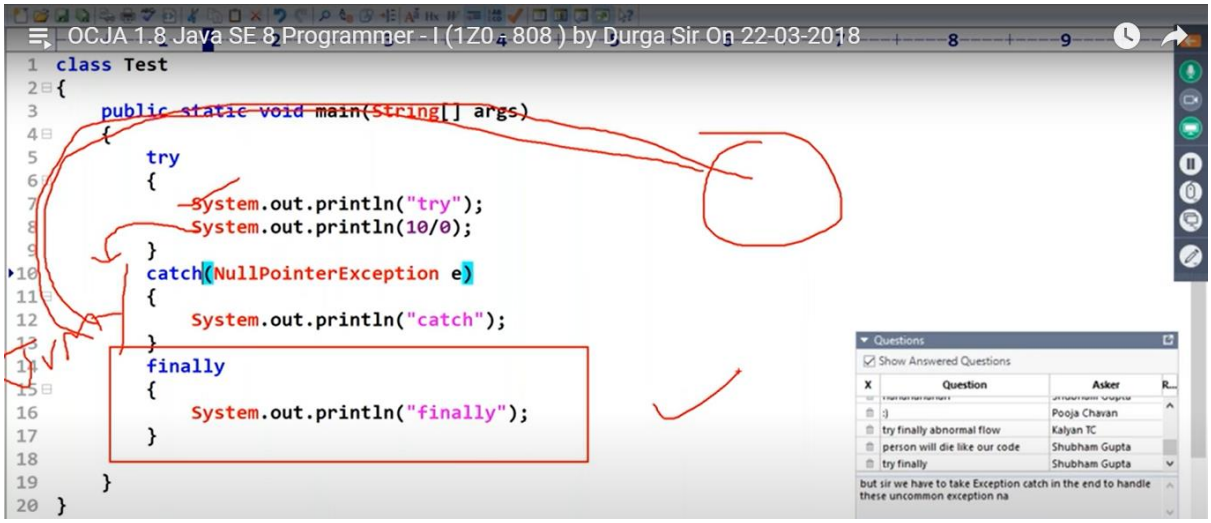
Exception Handled case

```
class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("try");
            System.out.println(10/0);
        }
        catch (ArithmeticException e)
        {
            System.out.println("catch");
        }
        finally
        {
            System.out.println("finally");
        }


    }
}
```

```
D:\durga_classes>javac Test.java

D:\durga_classes>java Test
try
catch
finally
```

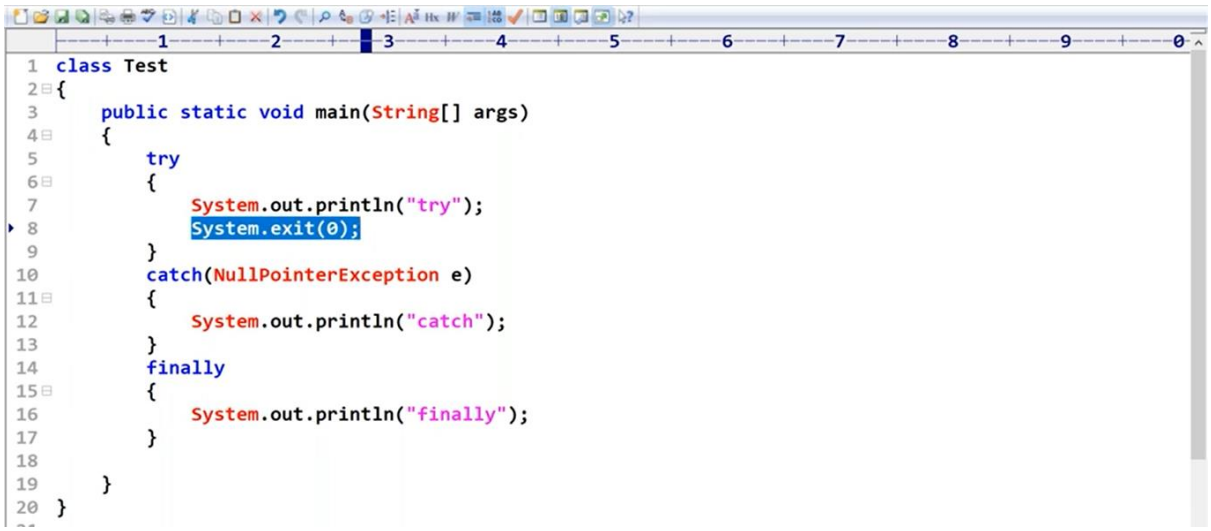
Exception UnHandled Case

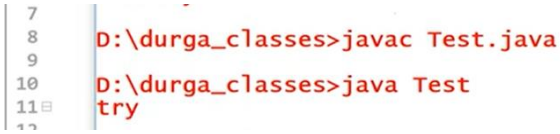
```
class Test
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("try");
            System.out.println(10/0);
        }
        catch (NullPointerException e)
        {
            System.out.println("catch");
        }
        finally
        {
            System.out.println("finally");
        }

    }
}
```



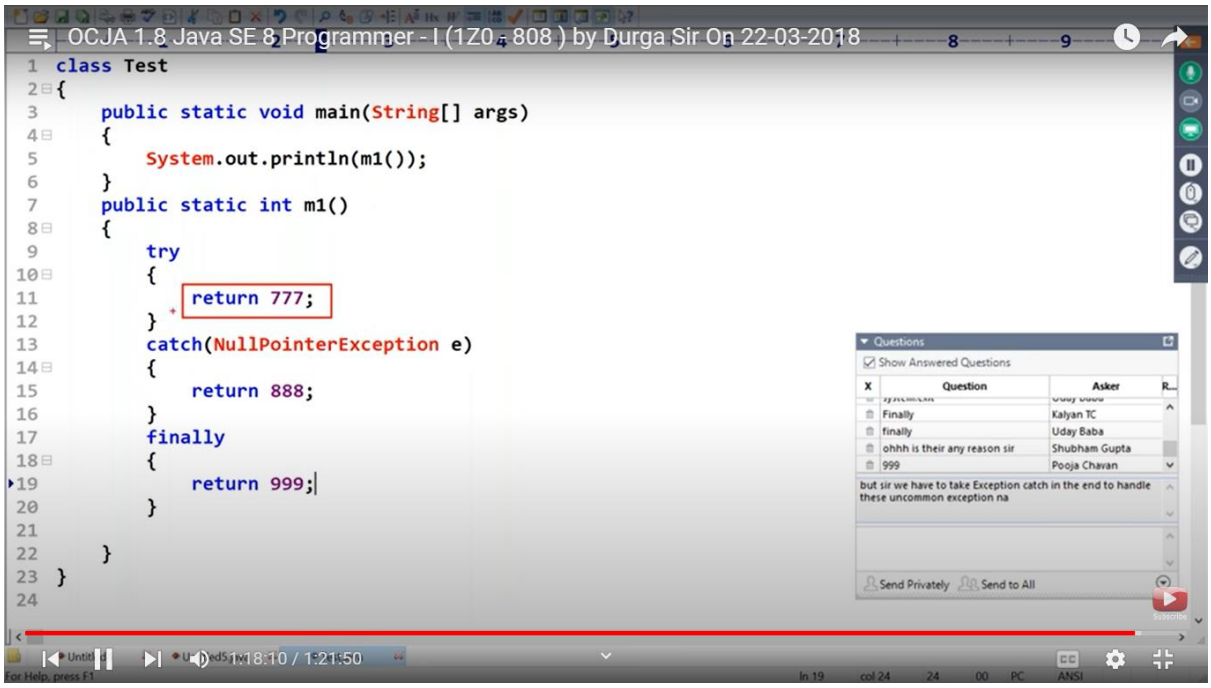
Finally Cant be written anywhere only after try catch .

When we use System.exit(0) bcos JVM will stop execution and the Finally will not be executed. That's the only situation where a Finally will not be executed

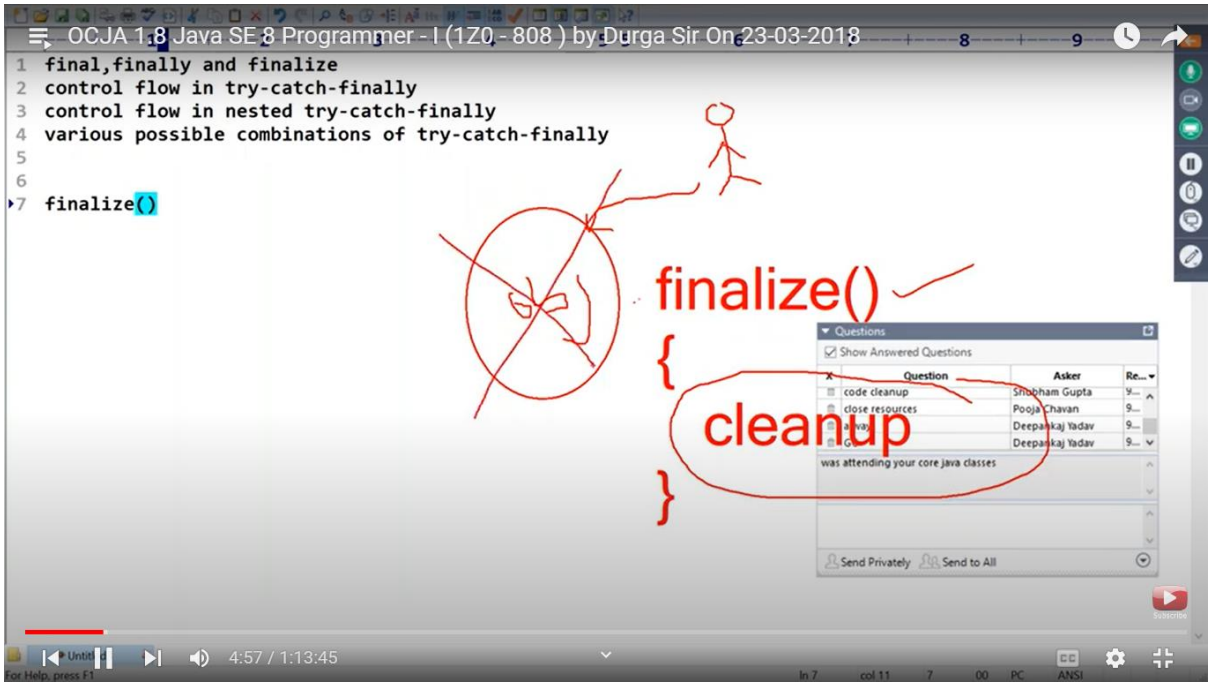




When return is there as bellow finally will be executed before returning and , the return in finally will stop return and takes higher priority . Hence return 999 will execute and not Return 777

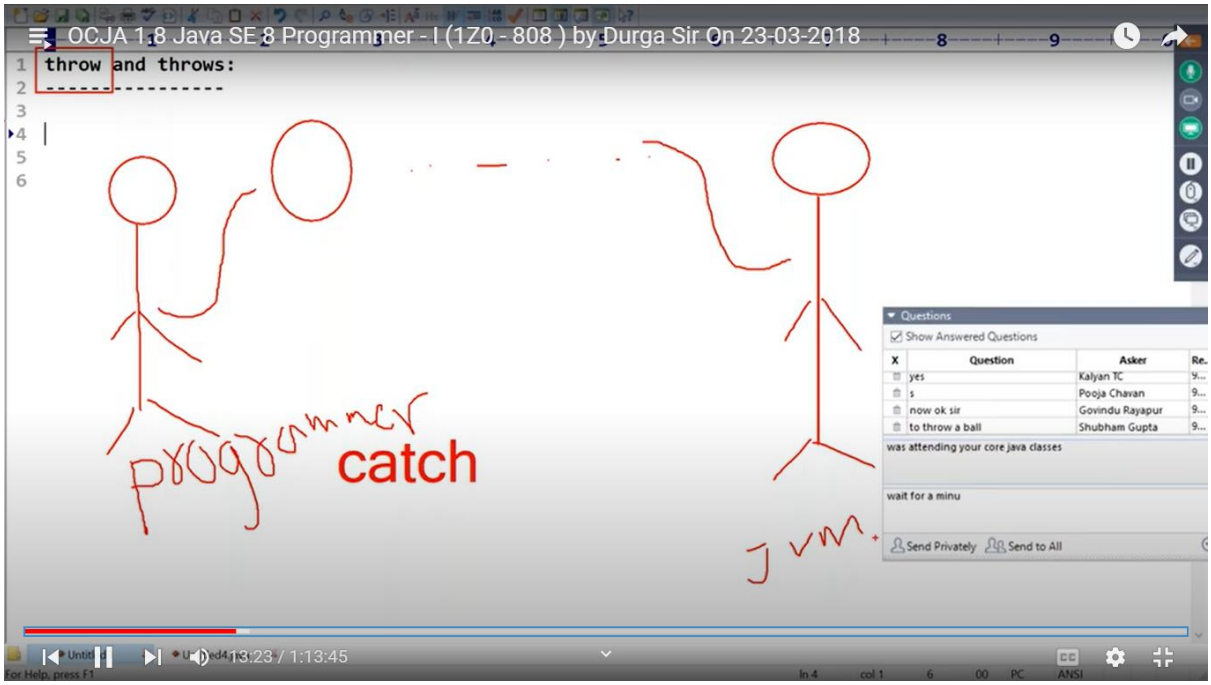


Garbage Collector -> Finalize Method- Before destroying an object- To Perform clean up activities

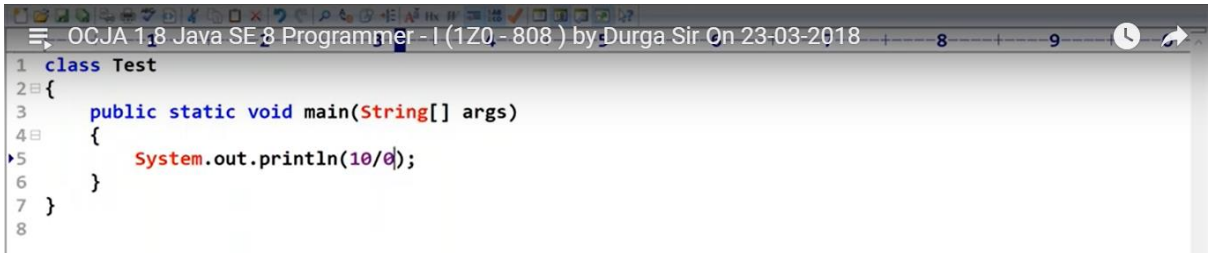


Throw and Throws

Programmer throws Exception and JVM catches.

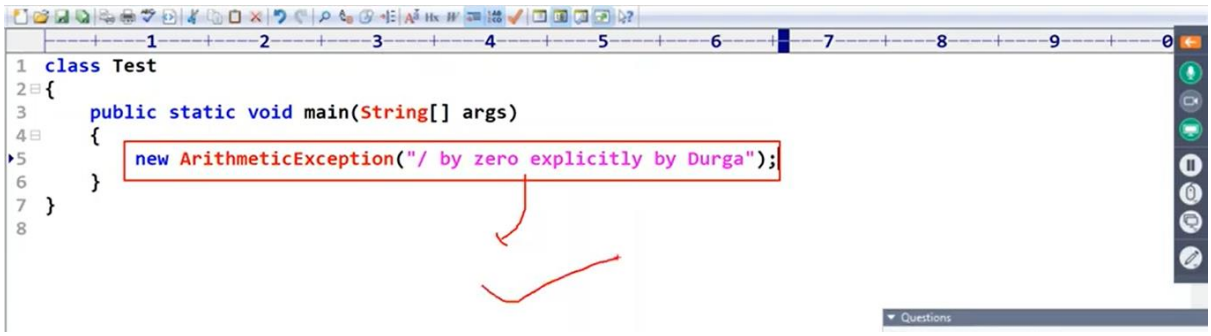


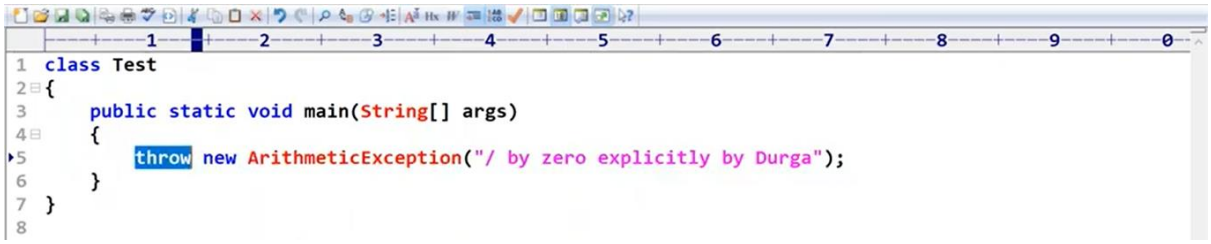
AE -> The method Main is responsible to create the Exception Object (with help of JVM only). JVM will see if the exception is being handled. And if not then it calls the default exception handler.



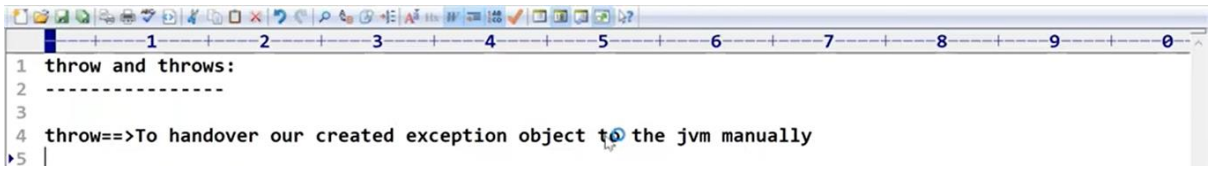
The Main method created the exception Object and handed over by the method to JVM

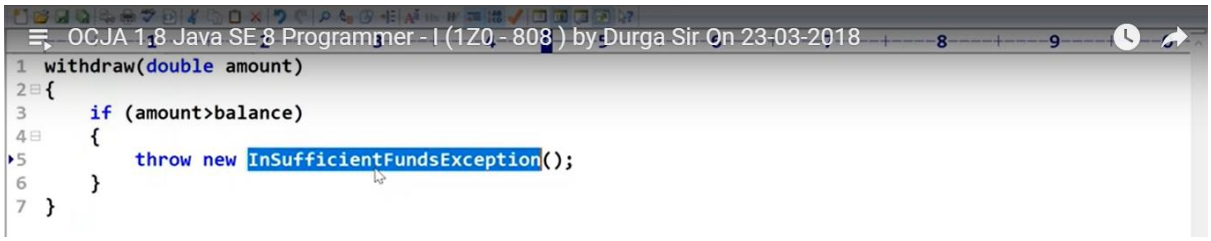
Sometimes we can create and exception object explicitly and hand it over to JVM. Then we will use throw keyword, after creating the Exception Object hand it over to the JVM





See the Explicitly by Durga in the below





```
Matrinomial website:
--------------------

Age: 99

throw new TooYoungException("plz wait some more time definetely u will get best match")
```

Throws

If there are any chance of Checked Exception being thrown then we need to handle them by try catch or throws Keyword. Eg below

There could be a chance or file not found exception but u r not handling it, CE

```
import  java.io.*;
class Test
{
    public static void main(String[] args)
    {
        PrintWriter pw= new PrintWriter("abc.txt");
        pw.println("Hello");
    }
}
```

```
Command Prompt
D:\durga_classes>{
'{' is not recognized as an internal or external command,
operable program or batch file.

D:\durga_classes>javac Test.java
Test.java:6: error: unreported exception FileNotFoundException; must be caught or
declared to be thrown
                PrintWriter pw= new PrintWriter("abc.txt");
                                ^
1 error
```

Eg2

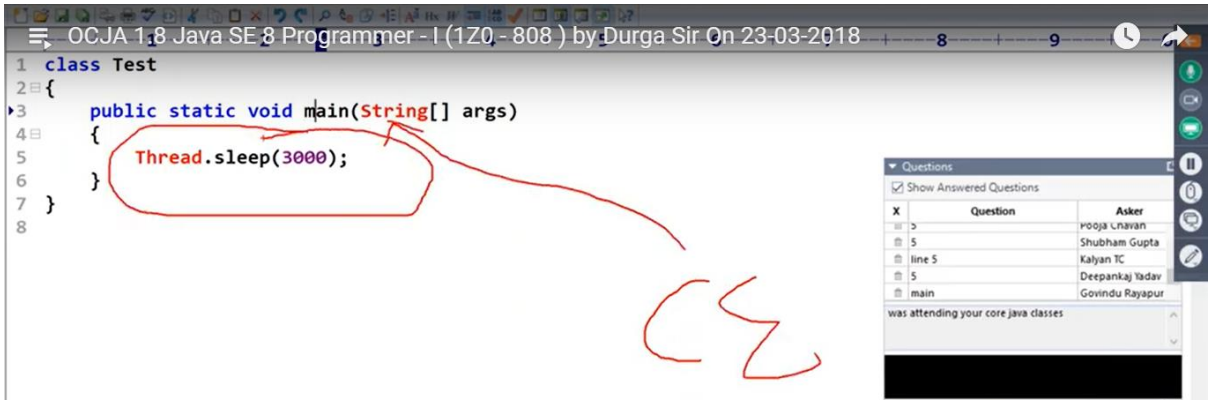

```
throws:
-------
InterruptedException=>checked
```

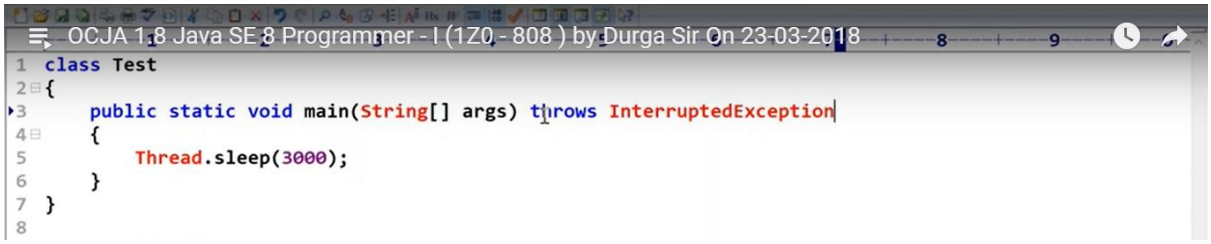
CE is say the exception is not reported

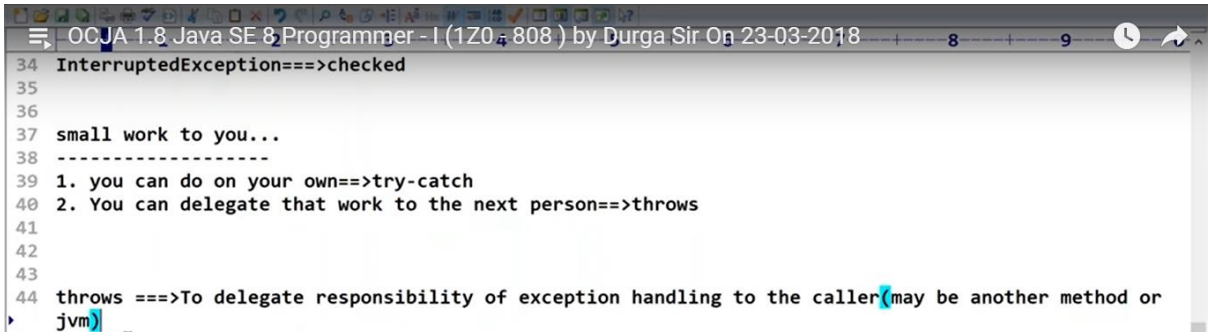
```
D:\durga_classes>javac Test.java
Test.java:5: error: unreported exception InterruptedException; must be caught or d
eclared to be thrown
                Thread.sleep(3000);
                             ^
1 error

D:\durga_classes>_
```

```
28
29
30
31
32  throws:
33  -------
34  InterruptedException===>checked
35
36
37  small work to you...
38  -------------------
39  1. you can do on your own==>try-catch
40  2. You can delegate that work to the next person==>throws
41
42
43
```

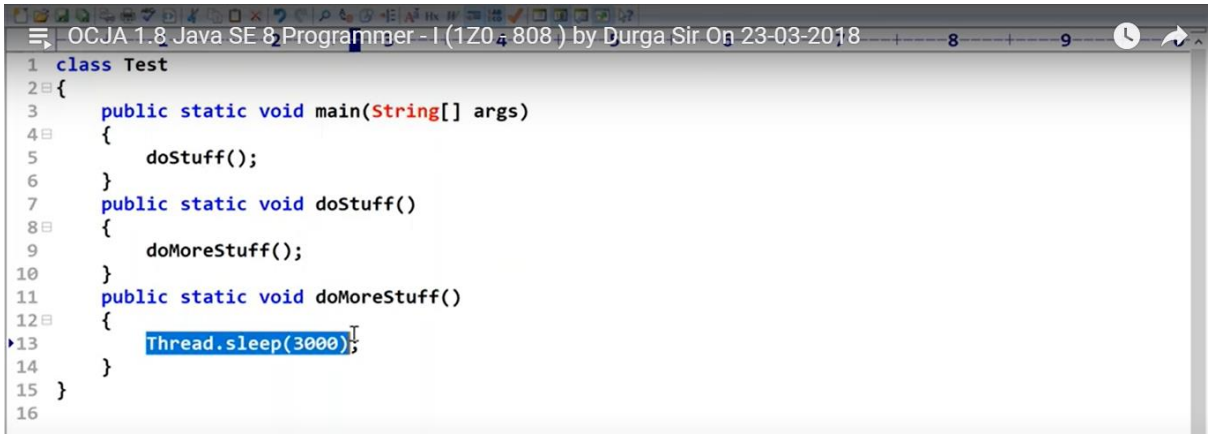
Line no 5 throws the exception, which is in main method. Main Method can delegate its called JVM to handle the exception by throwing the exception





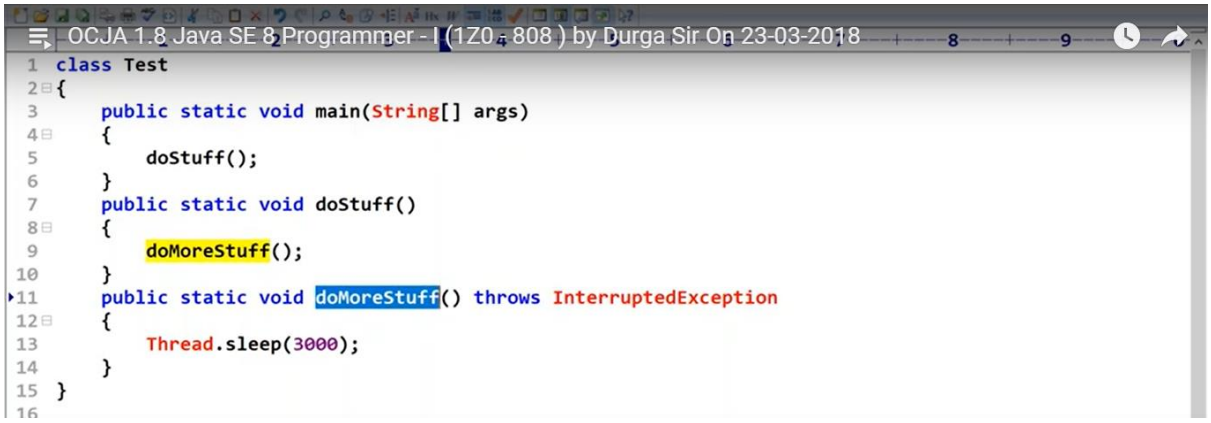


Three levels of method calls

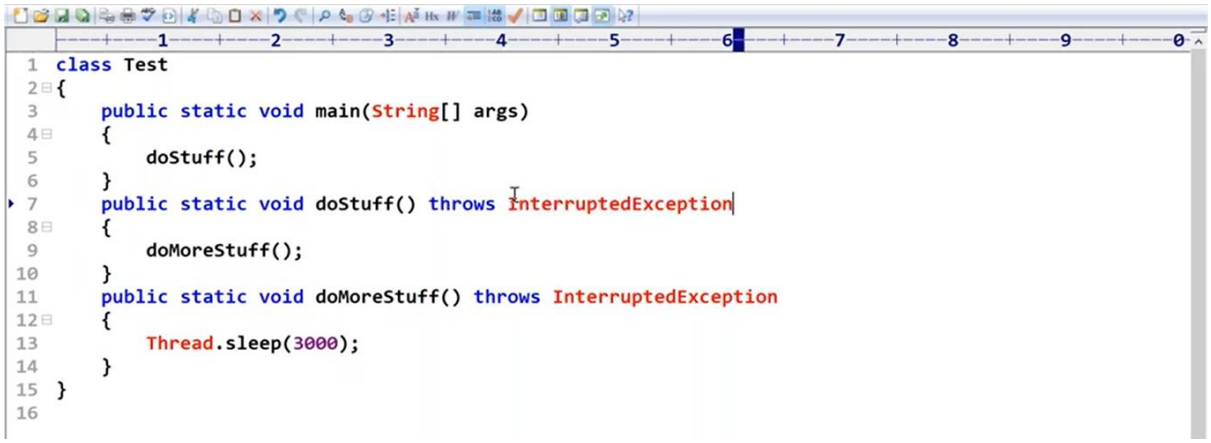


Line13 triggers the exception. CE asks doMoreStuff for Handling

Either use try catch or delegate the responsibility to caller (do Stuff). So u can use throws to ask CE compiler to check with doStuff

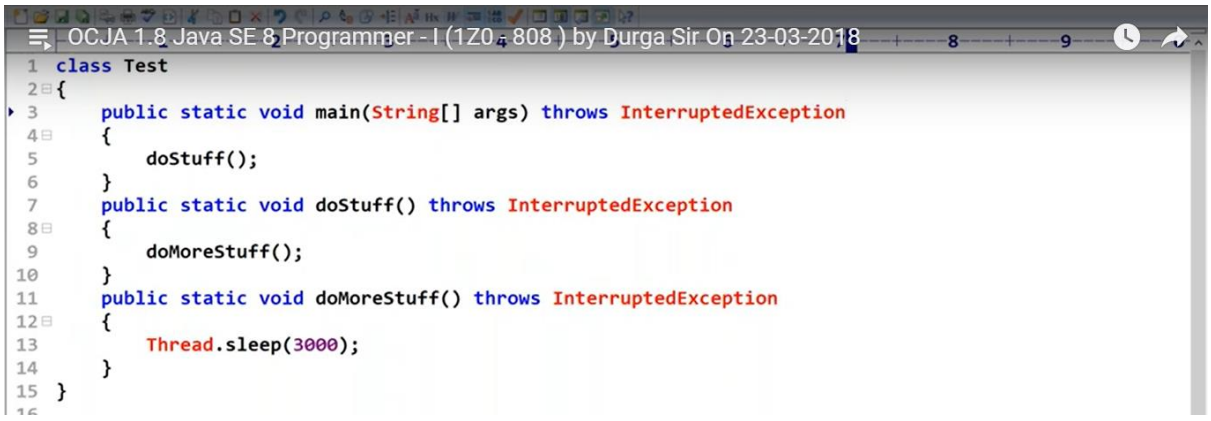


Check the error its in line 9 . Now do stuff can either use Try catch or delegate the resposbility to the caller (Main Method)_

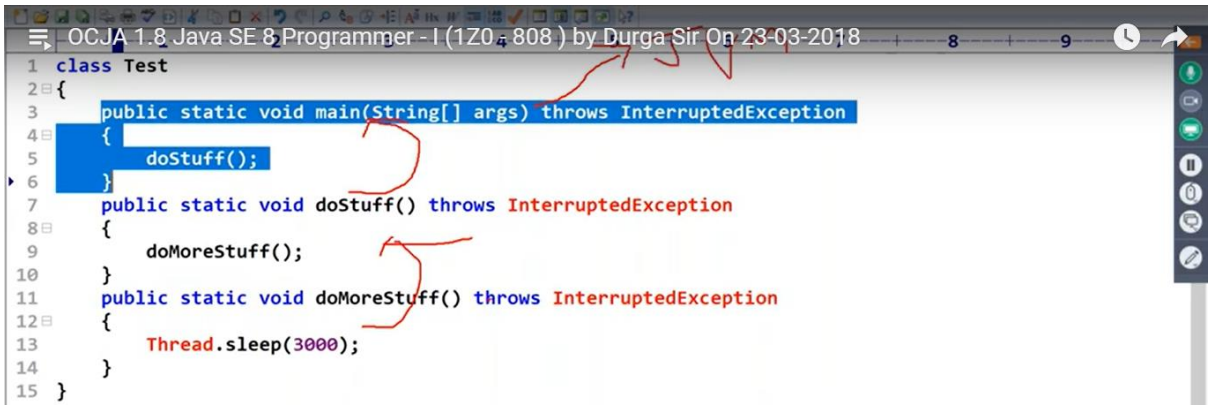


The exception will be thrown at line 5

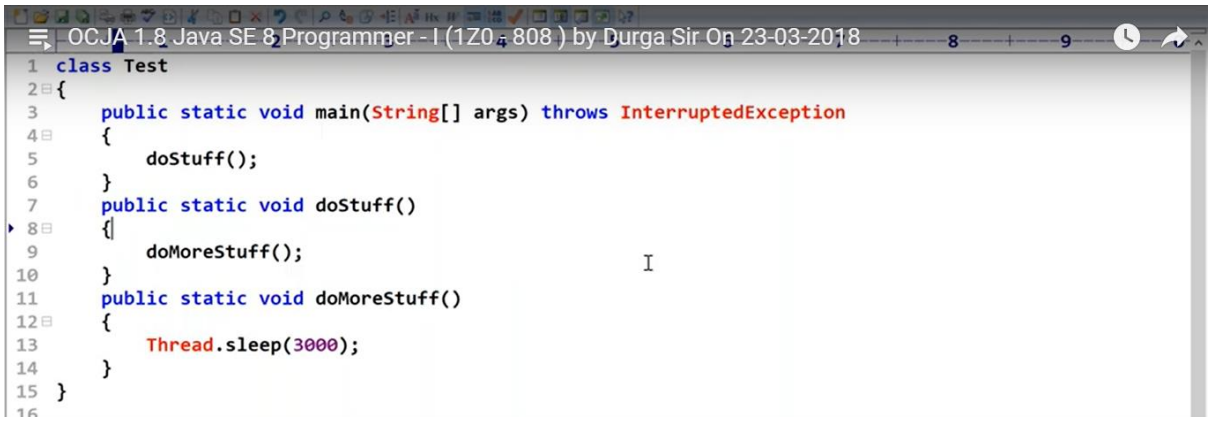
Now Main Method has delegated the exception handling to JVM. CE will now ignore it



Delegating



InValid- Compiler will check doMoreStuff to handle it , if we don't delegate it by throws in Line 11 how would it check with doStuff and then to Main Method . CE knows that Main is throwing this but u have not delegated it to Main Method. First Delegate



The ball comes to DoStuff

```
class Test
{
    public static void main(String[] args) throws InterruptedException
    {
        doStuff();
    }
    public static void doStuff()
    {
        doMoreStuff();
    }
    public static void doMoreStuff() throws InterruptedException
    {
        Thread.sleep(3000);
    }
}
```

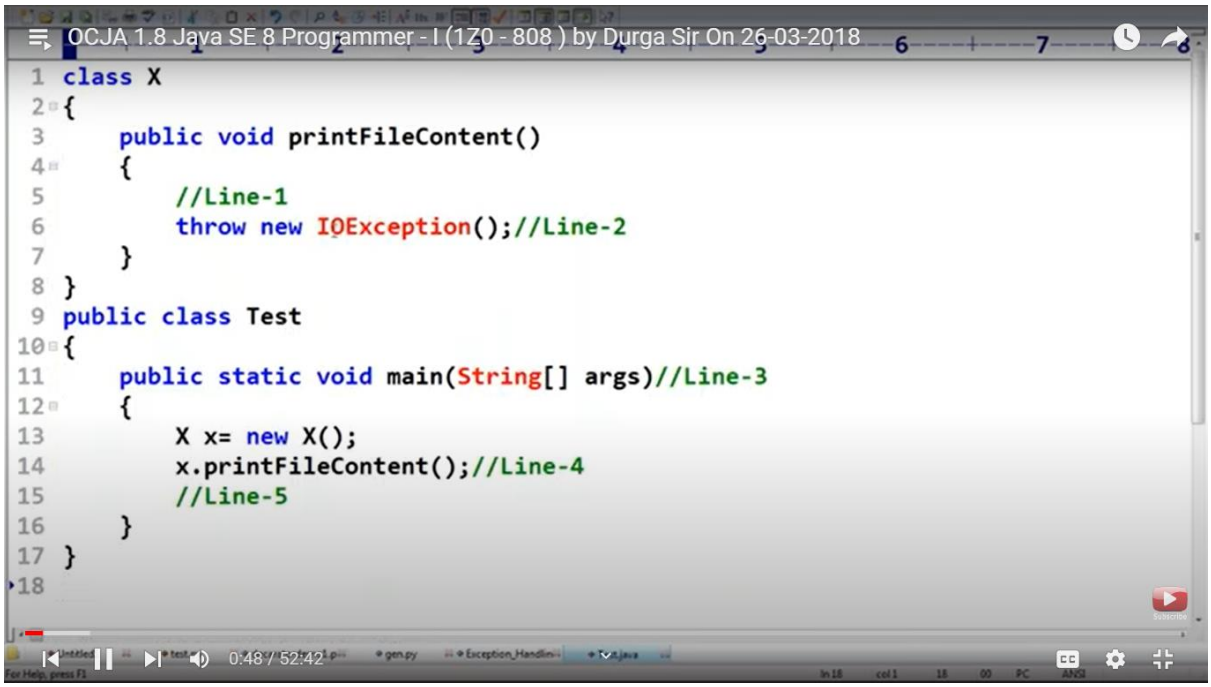
```
D:\durga_classes>javac Test.java

D:\durga_classes>javac Test.java
Test.java:13: error: unreported exception InterruptedException; must be caught or
declared to be thrown
                Thread.sleep(3000);
                             ^
1 error

D:\durga_classes>javac Test.java
Test.java:9: error: unreported exception InterruptedException; must be caught or d
eclared to be thrown
                doMoreStuff();
                              ^
1 error
```

It compiles now

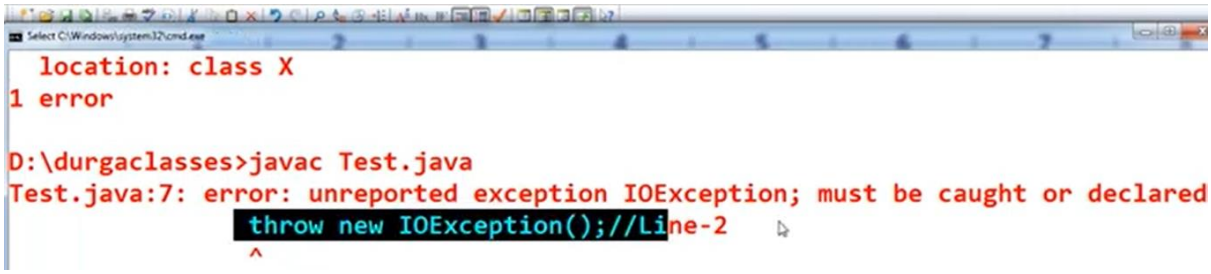
```
class Test
{
    public static void main(String[] args) throws InterruptedException
    {
        doStuff();
    }
    public static void doStuff() throws InterruptedException
    {
        doMoreStuff();
    }
    public static void doMoreStuff() throws InterruptedException
    {
        Thread.sleep(3000);
    }
}
```

Practice

Prob1: If a program is throwing any exception it should be handled or caught by throws



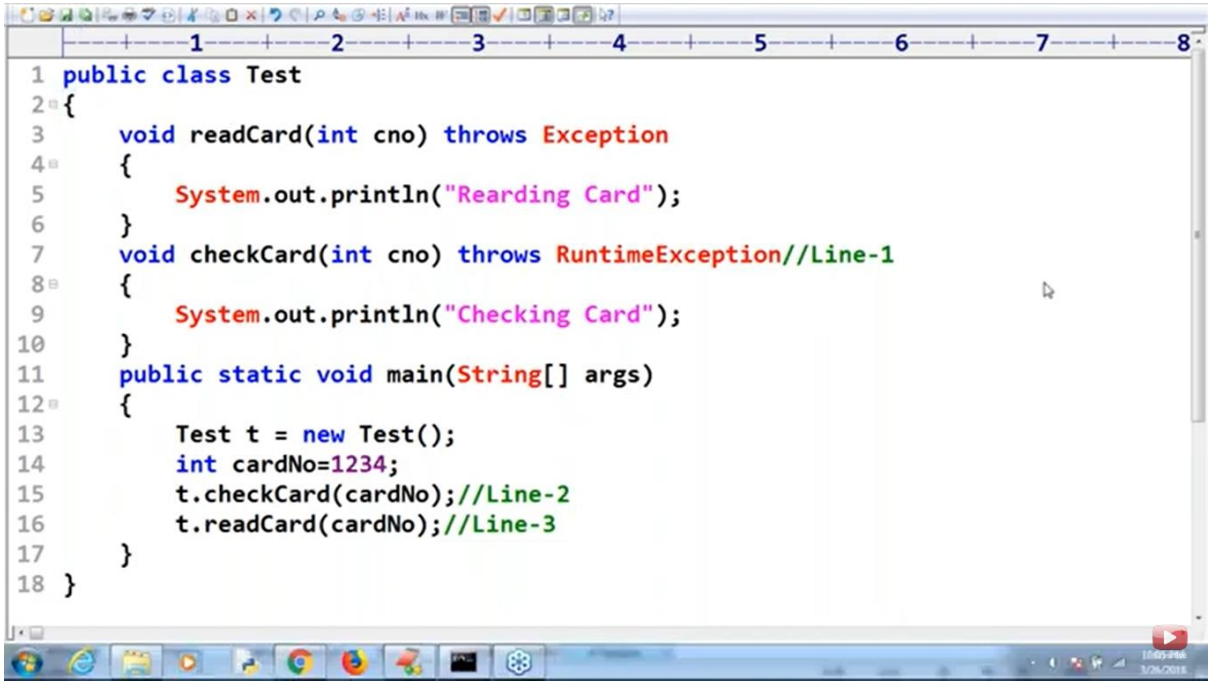

```
class X
{
    public void printFileContent()
    {
        //Line-1
        throw new IOException();//Line-2
    }
}
public class Test
{
    public static void main(String[] args)//Line-3
    {
        X x= new X();
        x.printFileContent();//Line-4
        //Line-5
    }
}
```



```
  location: class X
1 error

D:\durgaclasses>javac Test.java
Test.java:7: error: unreported exception IOException; must be caught or declared
                throw new IOException();//Line-2
                ^
```

Using throws will not handle the exception, if the exception occurs it will terminate the program.
Using throws will not prevent abnormal termination of the program, just stops CE error

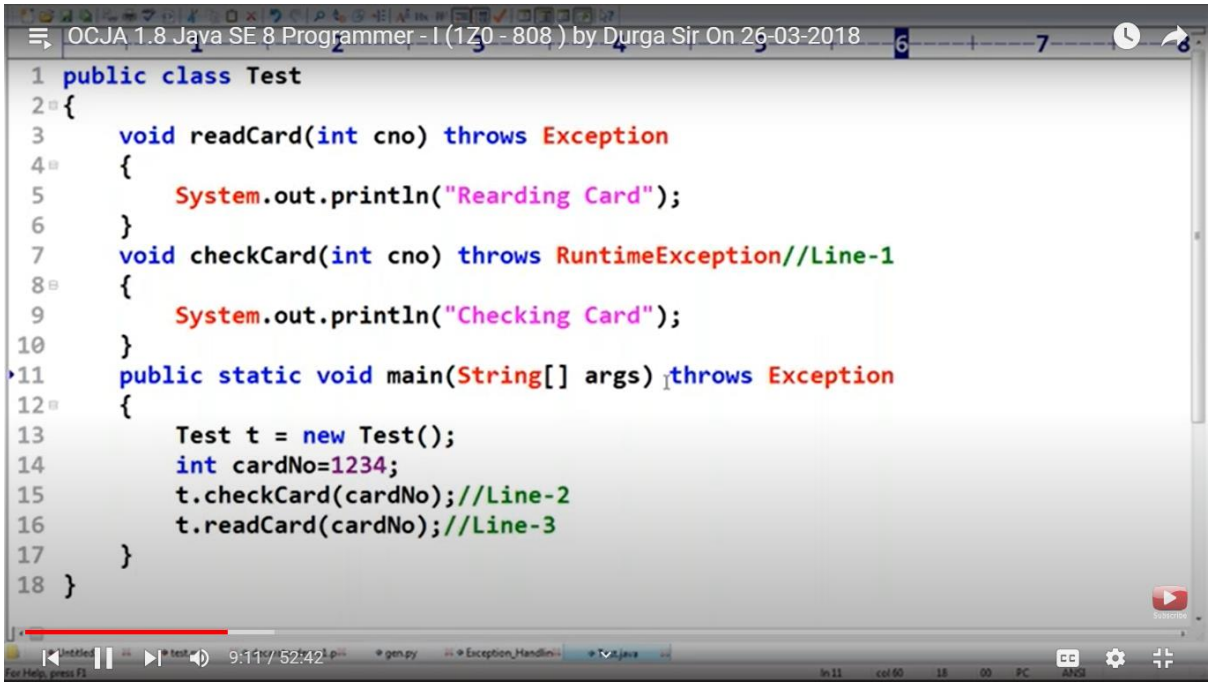
Throws Keyword can be used for Checked Exception for being more meaningful.



```
1  public class Test
2  {
3      void readCard(int cno) throws Exception
4      {
5          System.out.println("Rearding Card");
6      }
7      void checkCard(int cno) throws RuntimeException//Line-1
8      {
9          System.out.println("Checking Card");
10     }
11     public static void main(String[] args)
12     {
13         Test t = new Test();
14         int cardNo=1234;
15         t.checkCard(cardNo);//Line-2
16         t.readCard(cardNo);//Line-3
17     }
18 }
```

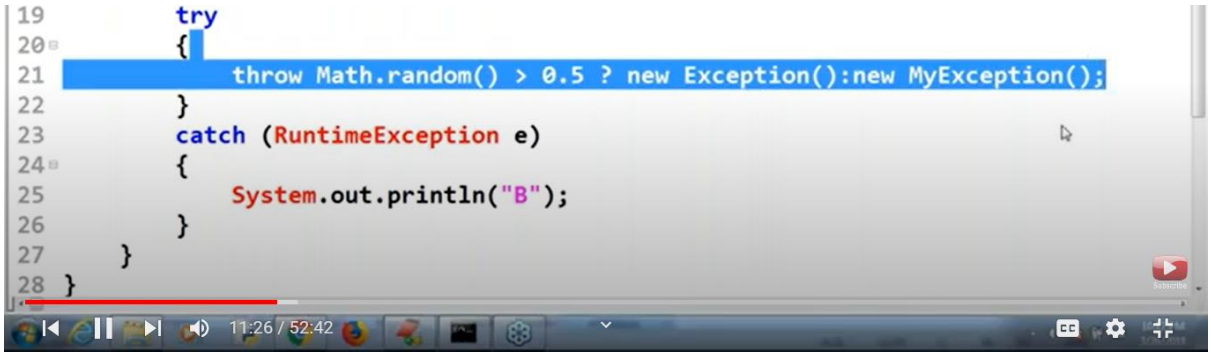
RunTimeException is unchecked Exception – So we don't need handle it so main need not throw the RunTimeException neither do checkCard. Having it is ok does not provide any meaning

ReadCard – throws Exception, its compulsory as its Partially Checked Exception. So, it has to be handled compulsory and thrown by Main Method, either use try catch or throws

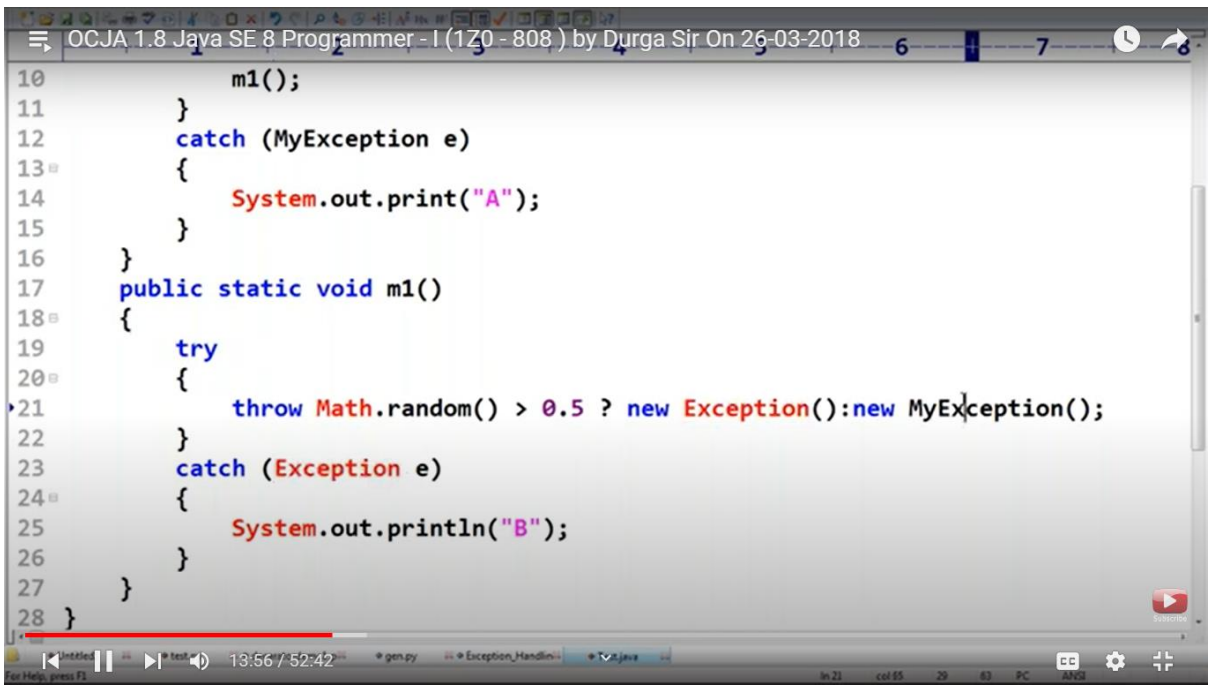



```
1  public class Test
2  {
3      void readCard(int cno) throws Exception
4      {
5          System.out.println("Rearding Card");
6      }
7      void checkCard(int cno) throws RuntimeException//Line-1
8      {
9          System.out.println("Checking Card");
10     }
11     public static void main(String[] args) throws Exception
12     {
13         Test t = new Test();
14         int cardNo=1234;
15         t.checkCard(cardNo);//Line-2
16         t.readCard(cardNo);//Line-3
17     }
18 }
```

```java
public class MyException extends RuntimeException
{
}
public class Test
{
    public static void main(String[] args)
    {
        try
        {
            m1();
        }
        catch (MyException e)
        {
            System.out.print("A");
        }
    }
    public static void m1()
    {
        try
        {
            throw Math.random() > 0.5 ? new Exception():new MyException();
        }
        catch (RuntimeException e)
        {
            System.out.println("B");
        }
    }
}
```



Line21 throws Exception or MyException. Both or possible , We are handling RunTimeException but not Exception and MyException



```
D:\durgaclasses>javac Test.java

D:\durgaclasses>javac Test.java
Test.java:1: error: class MyException is public, should be declared in a file na
public class MyException extends RuntimeException
       ^
1 error
```

```
1  public class Test
2  {
3      public static void main(String[] args)
4      {
5          String[] s= new String[2];
6          int i=0;
7          for(String s1: s)
8          {
9              s[i].concat("element"+i);
10             i++;
11         }
12         for(i=0; i<s.length;i++)
13         {
14             System.out.println(s[i]);
15         }
16     }
17 }
18
```

We have just declared, the array and not initialized. On null s[i] we cant call any method. Null Pointer Exception will be thrown. At run time not at compiler Time,

```
D:\durgaclasses>javac Test.java

D:\durgaclasses>java Test
Exception in thread "main" java.lang.NullPointerException
        at Test.main(Test.java:9)
```

```java
public class Test
{
    public static void main(String[] args)
    {
        String[] names={"Thomas","Bunny","Chinny"};
        String[] pwds=new String[3];
        int i =0;
        try
        {
            for (String n: names)
            {
                pwds[i]=n.substring(2,6);
                i++;
            }
        }
        catch (Exception e)
        {
            System.out.println("Invalid Name");
        }
```

SubString(Begin to n-1)

Thomas->(2,6)= 2 to 5 = omas,

Bunny-> 2 to 5 – no six index

This is the exception that will be thrown

The below error is from another program to show u what would be the error

```
D:\durgaclasses>java Test1
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String ind
        at java.lang.String.substring(Unknown Source)
        at Test1.main(Test1.java:5)
```

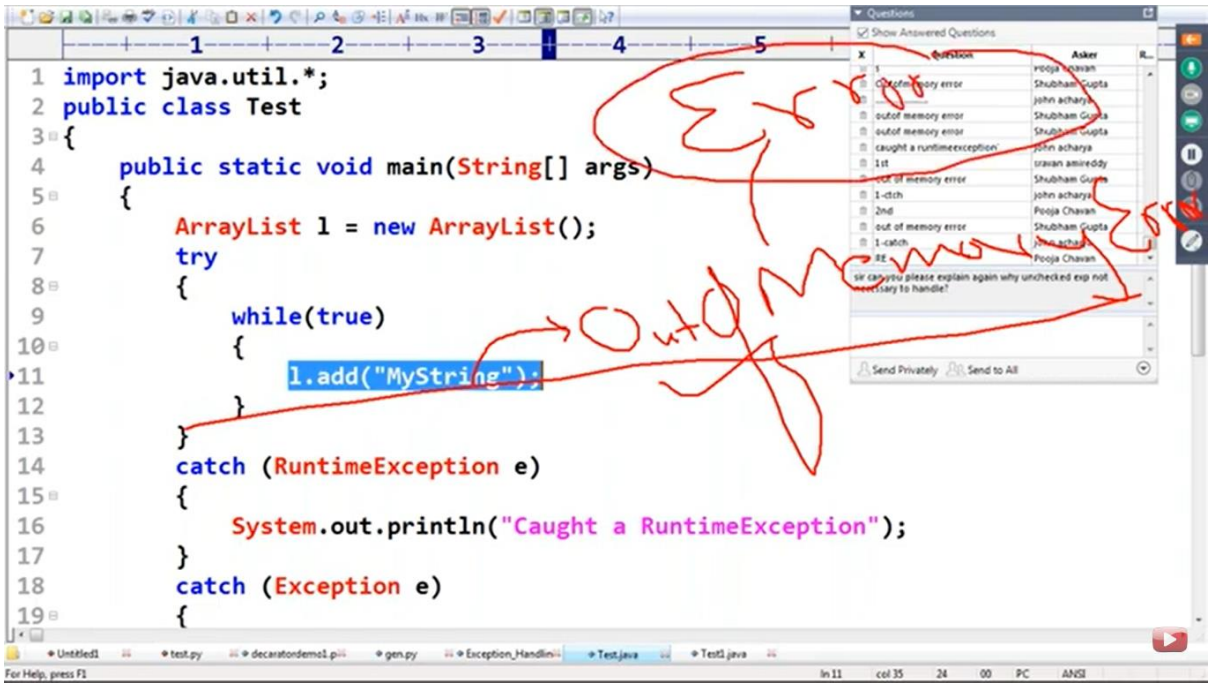
But the exception is handled . So pwds will be Omas, NULL , NULL

So the out put will be “invalid Name” -> which is in catch block

```
D:\durgaclasses>javac Test.java

D:\durgaclasses>java Test
Invalid Name
omas
null
null
```

```java
import java.util.*;
public class Test
{
    public static void main(String[] args)
    {
        ArrayList l = new ArrayList();
        String[] s;
        try
        {
            while(true)
            {
                l.add("MyString");
            }
        }
        catch (RuntimeException e)
        {
            System.out.println("Caught a RuntimeException");
        }
        catch (Exception e)
        {
            System.out.println("Caught an Exception");
        }
        System.out.println("Ready to use");
    }
}
```

Its not an Exception not an Error and cant be handled. Program terminates abnormally



```
D:\durgaclasses>java Test
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at java.util.Arrays.copyOf(Unknown Source)
        at java.util.Arrays.copyOf(Unknown Source)
        at java.util.ArrayList.grow(Unknown Source)
        at java.util.ArrayList.ensureExplicitCapacity(Unknown Source)
        at java.util.ArrayList.ensureCapacityInternal(Unknown Source)
        at java.util.ArrayList.add(Unknown Source)
        at Test.main(Test.java:11)
```

C – RunTime Error

```
200 }
201
202 What is the result?
203
204 A. Caught a RuntimeException printed to the console
205 B. Caught an Exception printed to the console
206 C. A runtime error is thrown at runtime
207 D. Ready to use printed to the console
208 E. The code fails to compile because a throws keyword required
```

A- True
B- False- Does not Cover all possible Errors
C- True
D- False – must be handled not must. Can be delegated
E- True

```
209
210 Q7. Which three are advantages of the Java Exception Mechanism?
211
212 A. Improves the program structure because the error handing code is separated
    from the normal program function.
213
214 B. Provides a set of standard exceptions that covers all possible errors
215
216 C. Improves the program structure beacuase the programmer can choose where to
    handle exceptions
217 D. Imporves the program structure because exceptions must be handled in the
    method in which they occurred.
218 E. Allows the creation of new exceptions that are tailored to the particular
    program being created.
219
```

```
Q8. Which 3 statements are true about exception handling?

A. Only unchecked exceptions can be rethrown
B. All Subclasses of the RuntimeException are recoverable
C. The parameter in catch block is of throwable type
D. All subclasses of RuntimeException must be caught or declared to be thrown
E. All Subclasses of the Exception except RuntimeException class are checked exceptions
F. All subclasses of the Error class are checked exceptions and are recoverable
```

Rethrown you can rethrow another exception from another exception. Throw can be used to rethrow

A- False , Any Exception can be rethrown

B- Exception and child classes are recoverable, Errors and its child class are not recoverable. So Runtime exception and its child classes are recoverable

Unchecked Exception does not throw compile time error so using throws means nothing to the compile. Because Compiler is not going to check if there is throws keyword there are not. Its being triggered at run time. Errors cannot be even handled at RunTime and will fail

So all Subclasses of RunTime Exception are recoverable by using try catch

c- TRUE – The argument in catch block is of throwable type which is the root

D-False. RE are unchecked and is not mandatory to be declared to be thrown or to be handled by try catch

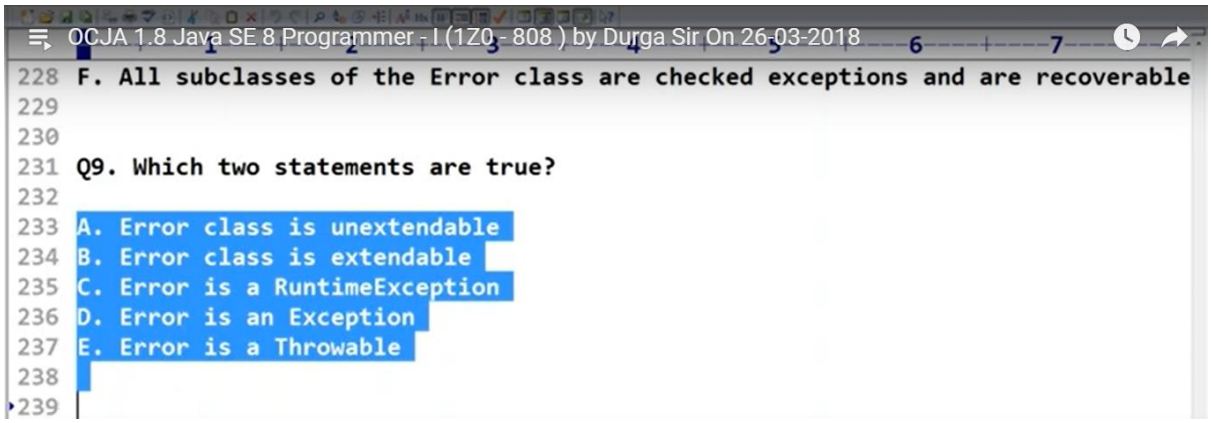
E= True , RE and Sub classes are unchecked and all subclasses of Exception other than RE are checked

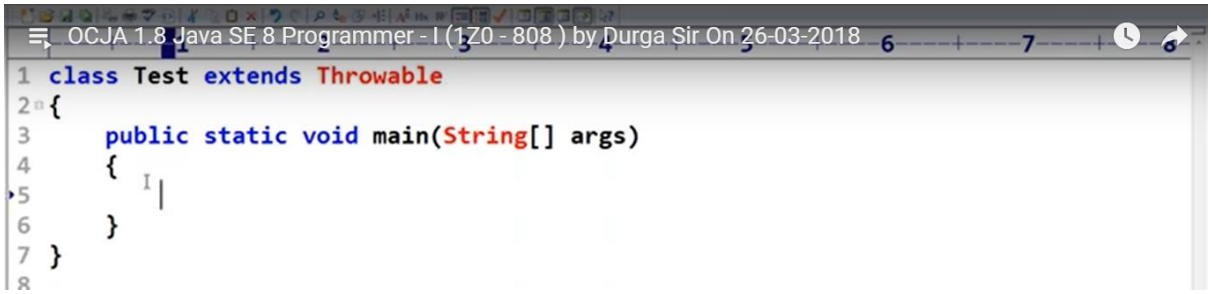
F- False. Error are unchecked and are not recoverable

B,C,E

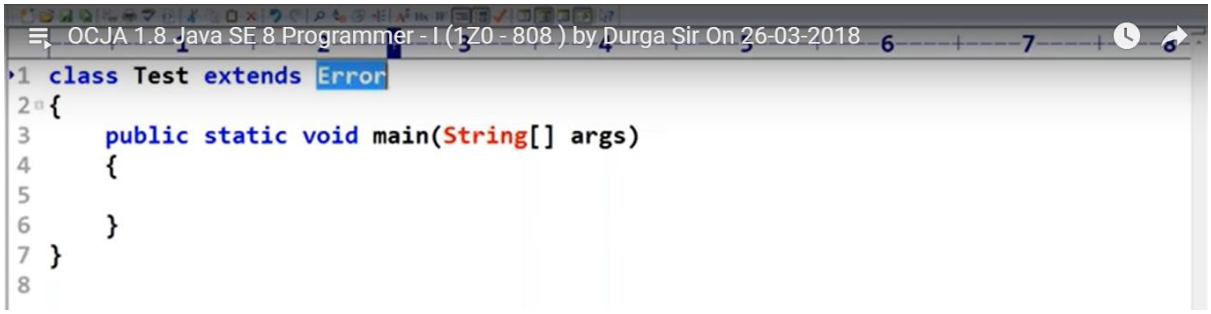
C- Error class is extendable can create child class of Error

D- Error is a throwable

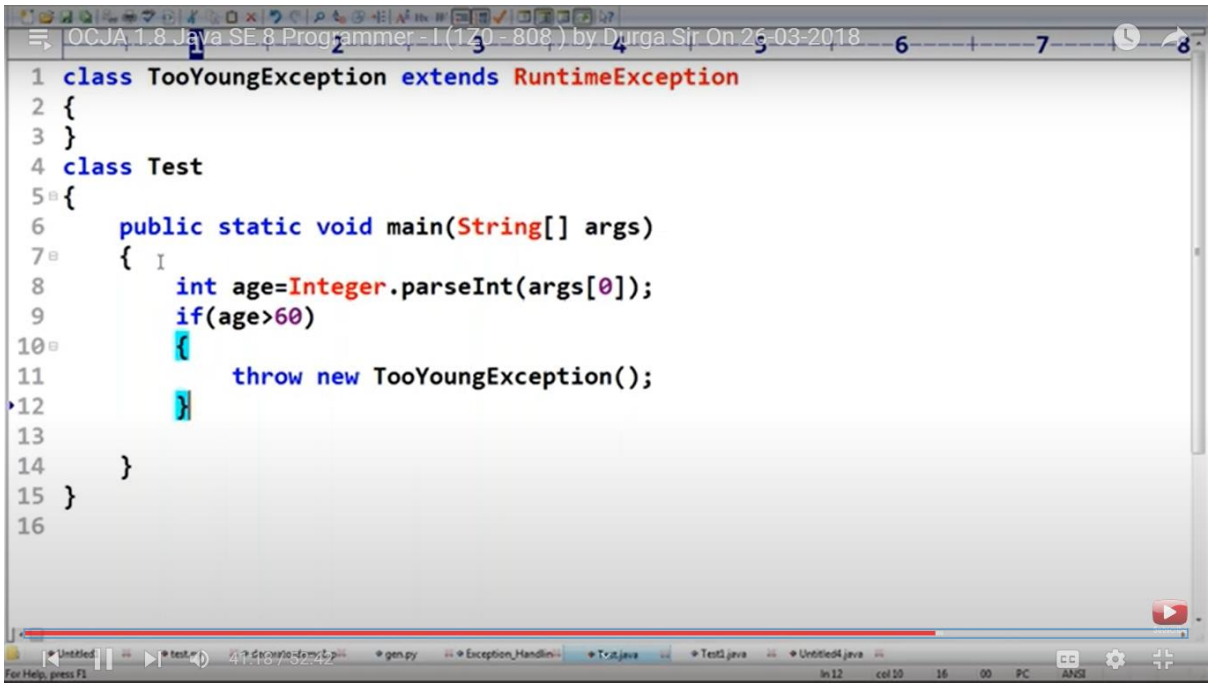


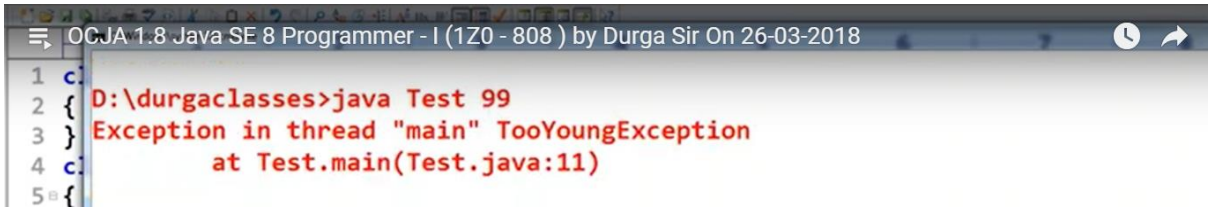


We can extend Error RE and Throwanle

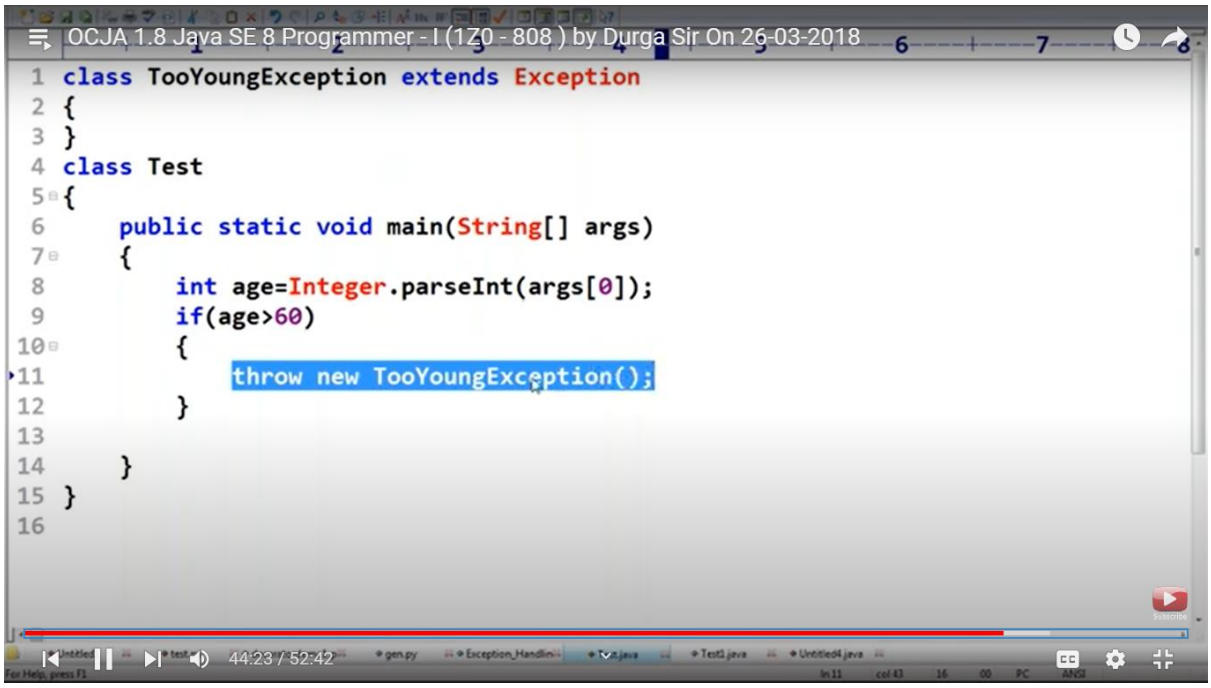


Why should we use RE.. The below is extending RE, its unchecked , if its unchecked even if its throwing compiler never gonna object even if its thrown or not, but at run time it will throw the exception.



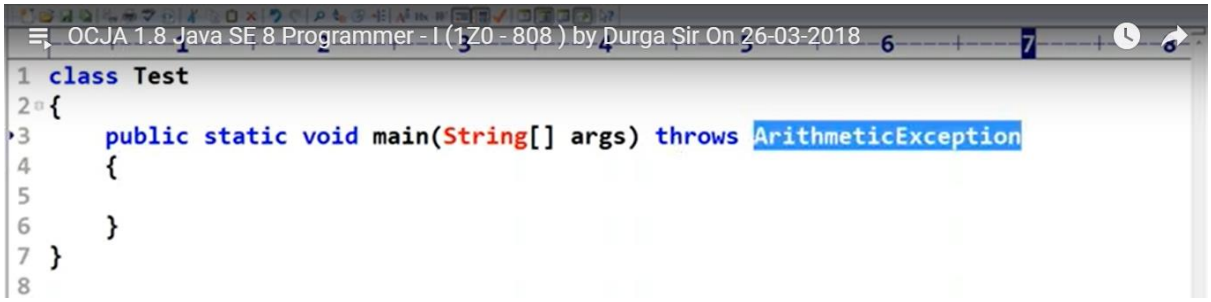


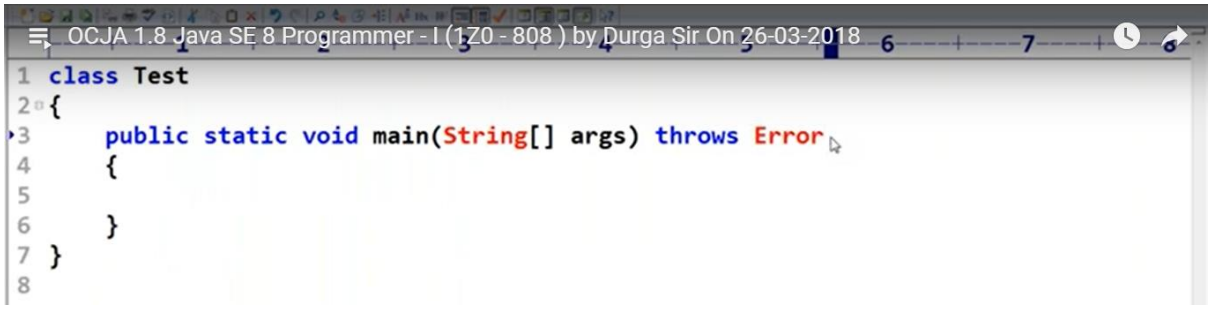
Here the class is Exception ,which is checked and Compiler will throw and error that this is not handled or uses throws



When I am creating user defined exception We always prefer RE because we are creating the exception to fail.. if u want it to fail why do you want to handle it… if you can handle it handle it in the code only you don't have to throw exception

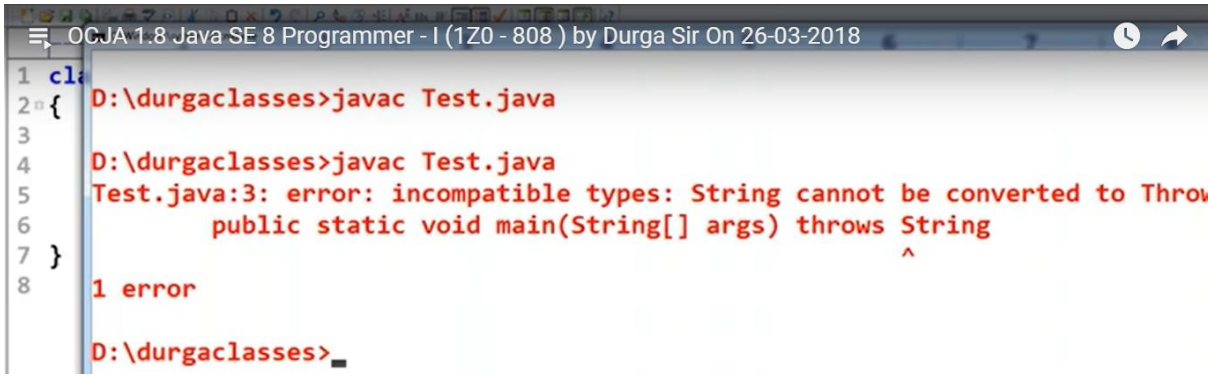
The below is valid only ..but does not has any meaning , Any throwable can be used with throws keyword



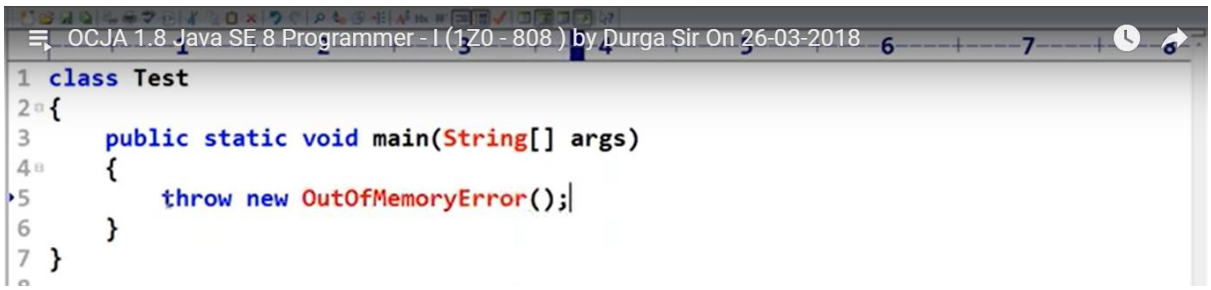


```
1 class Test
2 {
3     public static void main(String[] args) throws Throwable
4     {
5
6     }
7 }
```

```
1 class Test
2 {
3     public static void main(String[] args) throws String
4     {
5
6     }
7 }
```



Throw keyword can be used for any throwable type



```java
class Test
{
    public static void main(String[] args)
    {
        throw new Throwable();
    }
}
```