22/07/2023

One try, Multiple catch →

① One try can have multiple catch blocks.

② Within a single try block, there are chances of occurence of different types of problems. Every problem requires its own solution, so that best way is to write multiple catch blocks for one try.

③ ➤ If Exception class has relationship, then child Exception class should be upper catch block & parent Exception class should be lower catch block.

➤ If Exception class has no relationship, then we can write any of the Exception class in any of the catch block.

If you write the catch block which is not related to actual Exception, then program will terminate.

catch (Exception e) ← this should be problem related class or its parent class.

* If the problem is about ArithmeticException we can write RuntimeException also, which is its parent class.
or Exception class which is parent class of RuntimeException

Date: 22/07/23 YOUVA

Eclipse IDE → Right click in code → Run configuration → Java Application → Arguments

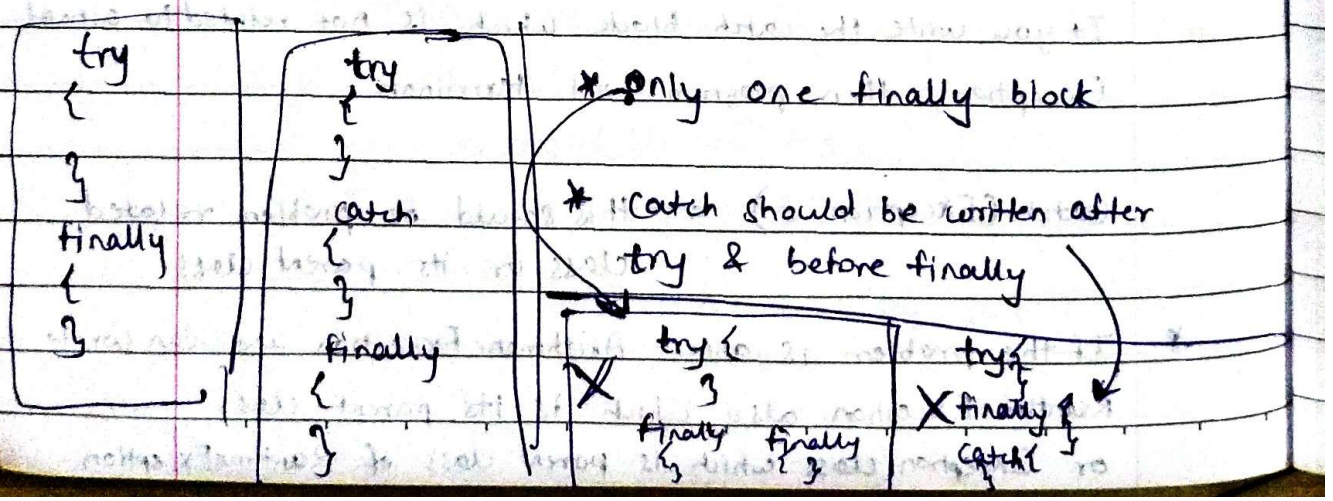
to provide the arguments
(it will be taken by String[] args)

```
public class Test
{
    p.s.v.m(String[] args)
    {   try {
            String s = args[0];
            int x = Integer.parseInt(s);
            int a = 10/x;
        }
    catch (ArrayIndexOutOfBoundsException e)
    {
    }   System.out.println("Enter input value");

catch (NumberFormatException e)
{
}   System.out.println("Enter number");

catch (ArithmeticException e)  ← if 10/0
{
}   System.out.println("Enter non-zero value");
    }
}
```

if we don't enter any argument

If the argument doesn't have integer, then it can't be parsed to integer

finally



Date: 22/07/23

finally - resource release करण्यासाठी code is written.
- e.g. we should close the file, close the database connection
- always executes
- ~~try~~ execute
- try मध्ये control गेल्यास finally मध्ये जाईल

① finally block's code is always executed whether there is a problem in try or not a problem in try.
② In both situations finally block's code will execute. Code which needs to execute compulsorily, that code should be written inside finally block.
③ Resource release code should be written inside finally block.
- e.g. If file is open in try block, then file should be closed in finally block.
- e.g. If database connection is open in try block, then Connection should be closed inside finally block.

***

```
public class Test
{
  p.s.v. main(String[] args)
  {
    try {
        Sysout.("try--start");
        int x = 10/0;
        Sysout.("try--end");
    }
    finally
    {
        Sysout.("finally--block");
    }
    Sysout.("main--end");
  }
}
```

Output:-
try--start
finally--block
Exception ArithmeticException by zero

***
If there is Exception in try, then finally will execute (if there's no catch), After finally executing Exception will be thrown.

```
public class Test
{
    p.s.v.m.(String[] args)
    {
        try {
            System.out.println("try--start");
            int x = 10/0;
            Sysout.("try--end");
        }
        Catch (ArithmeticException e)
        {
            Sysout.("catch--block");
        }
        finally
        {
            Sysout.("finally--block");
        }
        Sysout.("main--end");
    }
}
```

Output :-

```
try--start
catch--block
finally--block
main--end
```

ArithmeticException e

e.getMessage()

```
try
{
    return x;
}
finally
{
    return ...;
}
```

① this will execute

② then this will execute

① then this

- If try with finally block is there, and try has return statement, then before control returns back to caller, first finally block code will be executed.

```
public class Test1
{   public int m1()
    {   int x = 10;
        try{
            Sysout.("try--");
            return x;
        }
        finally {
            Sysout.("finally");
        } }
    p.s.v. main (String[] args)
    {   Test1 t = new Test1();
        int a = t.m1();
        System.out.println(a);
    }
}
```

Output:-

```
try--
finally
10
```

if you change the value of x here, even then 10 will be returned from try.

- If finally block changes return statement's value (if return is in try) ~~(of try)~~ then it won't be changed for caller; it will only change for finally.
- If try and finally both have return statements, then finally block's return statement will be executed.

```
public class Test1
{   public int m1()
    {   int x=10;
        try{
            Sysout.("try--");
            return x;
        }
        finally
        {   x = 30;
            Sysout.("finally--"+x);
            return x;
        } }
```

```
p.s.v.main (String[] args)
{
    Test1 t = new Test1();
    int a = t.m1();
    Sysout.(a);
```

Output:-

```
try--
finally--30
30
```

~~public class Test~~
~~{~~
~~public~~

Output:-
try -- block
x in catch : 55
finally -- block
x in finally : 45
in main : 55

```
public class Test
{   public int m1()
    {   int x = 10;
        try
        {   Sysout.("try -- block");
            int m = 10/0;
            return x; ← This will never be executed
        }
        catch (ArithmeticException e)
        {   x=55;
            Sysout.("x in catch : " + x);
        }   return x;    // 55 will be returned

        finally {
            System.out.println("finally -- block");
            x = 45;
            Sysout.("x in finally:" + x);
        }
    }   // Error- return x; Unreachable code

P.S.V. main(String [] args)
{   Test obj = new Test();
    int num = obj.m1();
    Sysout.("in main: " + num);
}
}
```

Page No.:
Date: 23rd July 2023 YOUVA

23/07/23

chance to caller method to handle exception

throws - checked exception

throws - ① It is used for propagating exception

② By using throws keyword we can give a chance to caller method to handle the exception.

③ throws keyword comes with method syntax or constructor syntax.

④ throws keyword is followed by Exception class. It can be one or multiple.

throws keyword is used with checked Exception class.

```
public void m1() throws IOException
{
}
public A() throws IOException, classNotFoundException
{
}
```

Unchecked exception

```
public class A
{
  public void m1()
  {
    System.out.println("m1--");
    try{
        B b = new B();
        b.m2();
    }
    catch(ArithmeticException e)
    {
      Sysout.("catch-block");
      Sysout.("m1--end");
    }
  }
```

try-catch in caller func

```
public class B
{
  public void m2()
  { Sysout.("m2--B");
    int x = 10/0;
    Sysout.("m2--end");
  }
}
```

```
public static void main(String[] args){
    Sysout.("main--start");
    A a = new A();
    a.m1();
    Sysout("main--end");
}}
```

* Compiler forces some statements to write try-catch or throws
for eg. - FileReader fr = new FileReader("abc.txt");

Page No.: Date: YOUVA

Q What is checked Exception?

→ There're certain statements in our program which have more priority to create problem at runtime. So, compiler forces those statements to be written in try-catch block or throws Exception (checked Exception)

```
public class Test {
    public void m2()
    {
        // Error - FileReader fr = new FileReader("abc.txt");
                because this line has possibility to create Exception
                (abc.txt may or may not exist)
    }

    public void m1()
    {
        try {
            FileReader fr = new FileReader("abc.txt");
        }
        catch(FileNotfoundException e)
        {
        }
    }

    p.s.v. main (String[] args) throws FileNotfoundException
    {
        FileReader fr = new FileReader("abc.txt");
    }
```

IOException (parent class)

child class → FileNotFoundException

throws keyword with overridding

throws keyword

① If parent class's method does not "throws" Exception, then at the time of overridding, we cannot write throws keyword.

② If parent class's method "throws" exception, then at the time of overridding, there's no need to write "throws" keyword. If we want to write "throws" keyword, then we can write in same exception class or its child exception class. But we cannot write their parent exception class

```
A                                   B extends A
{                                   {
  public void m1()                    public void m1() throws Exception
  {                                   {
  }                                   }
}
```

Error

throws keyword with superclass constructor and subclass constructor

```
public class A                      public class B extends A
{                                   {
  public A() throws IOException       public B() throws
  {                                     FileNotFoundException
  }                                     IOException
}                                   {
                                    }
```

① child class's constructor can or cannot write throws without or with writing it in parent class's constructor

② If parent class's constructor has throws exception, then child class's constructor should write same exception class or its parent exception class

throw keyword

1. Used to create custom exception
2. written inside method body
3. throw keyword is followed by Exception class's object.
4. We can throw only one Exception class's object at a time with throw keyword
5. We can create both custom checked & unchecked exception

to create checked exception ⇒ extends Exception class

to create unchecked exception ⇒ extends RuntimeException class

```
public class A
{
    public void m1()
    {
        ArithmeticException e = new ArithmeticException();
        throw e;
    }
}
```

Steps to create custom Exception

1. create Exception class with meaningful class name.
2. create a constructor with String parameter.
3. extends the Exception class
   to create checked exception ⇒ extends Exception class
   to create unchecked excepn ⇒ extends RuntimeException
4. call super constructor & send string parameter
5. create the Exception class's object
6. throw the Exception class's object with throw keyword.

Page No.:
Date:
YOUVA

① class A
{
public A() throws FileNotFoundException {
}
}

// Error ~~Unchecked Exception~~
// class B extends A {
// Default constructor cannot handle exception thrown by Implicit super constructor
// }

② class B extends A
{
public B() throws IOException{
}
}

class A
{
}

③ class A
{
public A() throws FileNotFoundException {
}
}

class B extends A
{
public B() throws IOException{
}
}

child class should write same exception class or that exception class's parent class

④ class A
{
public P() {
}
}

class B extends A
{
public B() throws IOException
{
}
}

Code from internet :-

```
class UserDefinedException extends Exception
{
    public UserDefinedException(String str)
    {
        super(str);
    }
}

public class Test
{
    p.s.v. main(String[] args)
    {   try{
            throw new UserDefinedException("user-defined exception");
        }
        catch(UserDefinedException ude)
        {
            Sysout: ("caught the exception");
            Sysout.(ude.getMessage());
        }
}   }
```

~~Code from internet~~ :- My code:-

```
public class Test
{
    public static void validate(int age)
    {   if (age <= 18)
        {   throw new ArithmeticException("Invalid age");
        }
    }

    public static void main(String[] args)
    {
        validate(16);
    }
}
```

Output :-

Exception in thread main java.lang.ArithmeticException: Invalid age

My code :-

Date: YOUVA

```
public class Test
{
    public static void validate(int age)
    {
        if(age<=18)
        {
            try{
                throw new ArithmeticException("Invalid age");
            }
            catch(ArithmeticException e)
            {
                System.out.println(e.getMessage());
            }
        System.out.println("outside try--catch");
        }
        else
        {
            System.out.println("You can vote");
        }
    }

    public static void main(String[] args)
    {
        validate(14);
    }
}
```

Output :-

```
Invalid age
Outside try--catch
```