MODULE - 4 PACKAGES & INTERFACES

INTRODUCTION

Java Source File Structure

- *A java Program can contain any no. Of classes but at most one class can be declared as public. "If there is a public class the name of the Program and name of the public class must be matched otherwise will get compile time err[illegible]"*
- *If there is no public class then any name can b[illegible]given f[illegible]java source file.*
- *Example:*

```
class A
{
}
public class B
{
}
public class C
{
}
```

Case 1:

If [illegible]ere is no public class the[illegible]se any name for java source file there are no [illegible]tions.
Exa[illegible]A.java B.java C.java Ash[illegible]a

C[illegible]e 2:

If class B [illegible] as public then the name of the Program should be B.java otherwise will [illegible]pile time error saying "class B is public, should be declared in a file named B.j[illegible]

Case 3:

If both B and C class[illegible] are declared as public and name of the file is B.java then will get compile time error saying "class C is public, should be declared in a file named C.java" It is highly recommended to take only one class for source file and name of the Program (file) must be same as class name. This approach improves readability and understandability of the code.
Example:

```
class A
{
        public static void main(String args[])
        {
                System.out.println("A class main method is executed");
        }
        }
        class B
        {
                public static void main(String args[])
                {
                        System.out.println("B class main method is executed");
                }
        }
        class C
        {
                public static void main(String args[])
                {
                        System.out.println("C class main method is executed");
                }
        }
        class D
        {
        }
```

Output:

```
D:\Java>java A
 A class main method is executed

D:\Java>java B
 B class main method is executed

 D:\Java>java C
 C class main method is executed

D:\Java>java D Exception in thread "main" java.lang.NoSuchMethodError:
main
```

$$\begin{tikzcd}
 & & \text{javac sample.java} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{A.java} & \text{B.java} & \text{C.java} & \text{D.java}
\end{tikzcd}$$

- *Java Program can be compiled but not java class in that Program for every class one dot class file will be created.*
- *We can run a java class but not java source file whenever we are trying to run a class the corresponding class main method will be executed.*
- *If the class won't contain main method then will get runtime exception saying "NoSuchMethodError: main".*

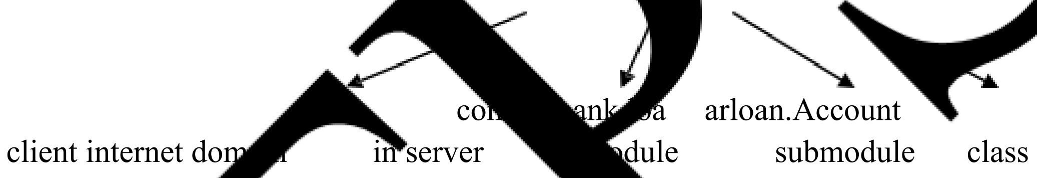
- *If we are trying to execute a java class and if the corresponding .class file is not available then will get runtime error saying "NoClassDefFoundError: Ashok".*

Packages

- A group of similar types of classes, interface, enumeration or sub-package.
- The main objectives of packages are:

  - ✔ To resolve name conflicts.
  - ✔ To improve modularity of the application.
  - ✔ To provide security.
  - ✔ There is one universally accepted naming co[illegible]ersion for packages that is to use internet domain name in reverse.
  - ✔ it becomes easier to locate the related classes and it also provides a good structure for projects with hundreds of classes and other files.
- Packages are categorized into

  - ✔ **Built-in Package:** Existing Java pack[illegible] java.lang java.util etc.
  - ✔ **User-defined-package:** J[illegible]ackage cr[illegible] user to catego[illegible] their project's classes and interface.
- Example


co[illegible]ank.loa[illegible]arloan.Account
client internet dom[illegible]
in server
[illegible]dule
submodule
class

How to com[illegible]ge Program:

Sy[illegible]x:

javac [illegible]ectory filename

```
Example:
package com.sbibank.
an.carloan; class Acount
{
        public static void ma[illegible]ring args[])
        {
                System.out.println("package demo");
        }
}
```

javac Acount.java generated class file will be placed in current working directory.

```
        Cwd
         |-      Account.class
Javac -d .|Acount.java
```

- ✔ -d means destination to place generated class files "." means current working directory.

- ✔ Generated class file will be placed into corresponding package structure.

```
Cwd
 |-   com
 |     |-  sbibank
             |----loan
                   |-    carloan
                           |
                           -    Account.class
```

If the specified package structure is not already available then this command itself will create the required package structure.

How to execute package Program:

D:\Java>java com.sbibank.loan.carloan.Account

At the time of execution compulsory should provi ully qualified name

Conclusion 1:

In any java Program there sh ost one package statement tha if there are more than one package sta ent will mpile time error.

Example:

```
package
pack1;
package
pack  class A
{

Ou   Compile time error.
 D:       vac A.java
 A.java     , interface, or enum ex    ted package pack2;
```

Conclusion 2:

In any java Program the n comment statement should be package statement [if it is available] otherwise will time error.

Example:

```
import java.util.*;
package pack1;
class A { }
Output: Compile time error.
D:\Java>javac A.java A.java:2: class, interface, or enum expected package pack1;
```

Import Statement

- import is a keyword used to import built-in and user-defined packages into your java source file which can be used to access classes and interface from a package.
- Import statement specified one time and can be accessed multiple times.

- All import statement should be specified after the package statement and before the class.

- There are 3 different ways to refer to any class that is present in a different package:

✔ **Using fully qualified name (But this is not a good practice.)**

❖ **Adv :** For this approach, there is no need to use the import statement.

❖ **Disadv :**

- In fully qualified name to import any class into the program, then only that particular class of the package will be accessible in the program, other classes in the same package will not be accessible.
- But you will have to use the fully qualified name every time to accessing the class or the interface, which increases the length of the code.

❖ This is generally used when two packages have classes with same names. For example: java.util java.sqld package contain Date class.

❖ **Example**

```
//save by

pack    pack
p   ic cla

    public void msg

            System.out.println("

    }

// sa    y B.

package m
ck calss B
{
    public static void    ain(String[] args)
    {
            pack.A obj = new pack.A();          //using fully qualified name
            obj.msg();
    }
}
```

Output :
Hello

- **To import only the class/classes you want to use**

    - If you import packagename.classname then only the class with the classname in the package with the packagename will be available for use.

❖ **Example :**

```
//save by A.java

package pack
public class A
{
    public void msg()
{
    System.out.println("Hello");
}
                  }

// save by B.java

package mypack
import pack.A
class B
{
    public static void main(String[] args)
    {
            A obj = new A();            //using fully qualified name
            obj.msg();
    }
}
Output :
    Hello
```

✔ **To import all the classes from a particular package**

❖ Usage of packagename.*, then all the classes and interfaces of this package will be accessible but the classes and interface inside the subpackages will not be available for use.

❖ The import keyword is used to make the classes and interface of another package accessible to the current package.

❖ **Example :**

```
//save by A.java

package pack
public class A
{
        public void msg()
        {
                System.out.println(“Hello”);
        }
```

```
    }

// save by B.java

package mypack
import pack.*;
class B
{
        public static void main(String[] args)
        {
                A obj = new A();                    //using fully qualified name
                obj.msg();
        }
    }
Output :
        Hello
```

Note: *class Test*

```
{
public static voi[illegible]in(String args[illegible]
{
        ArrayList l=[illegible]rrayList([illegible]
}
}
Output: C[illegible]npile time error.
D:\Ja[illegible] Test.java
Tes[illegible]va:3[illegible]not find symbol symbol : c[illegible]rray[illegible]ist location: class Test
[illegible]ayList l=n[illegible]rrayList();
```

- *We [illegible] can resolve t[illegible]lem by using fully [illegible]qualified name "java.util.ArrayList l=new java.util.ArrayList(); [illegible] problem with using fully qualified name every time is it increases length of the co[illegible]educes readability.*
- *We can resolve this problem [illegible]ing import statements.*
- *Example:*

```
import java.util.ArrayList;
class Test
{
        public static void main(String args[])
        {
                ArrayList l=new ArrayList();
        }
}
Output: D:\Java>javac Test.java
```

Hence whenever we are using import statement it is not require to use fully qualified names can use short names directly. This approach decreases length of the code and improves readability.

Case 1:
Example:

```
 import java.util.*;
import java.sql.*;
class Test
{
          public static void main(String args[])
          {
                    Date d=new Date();
          }
}
```

Output: Compile time error.

```
 D:\Java>javac Test.java
 Test.java:7: reference to Date is ambiguous, [illegible] java.sql.D[illegible] java.sql and
class java.util.Date in java.uti[illegible]

Date d=new Date();
```

Even in the List case also [illegible] get the sa[illegible] ambiguity prob[illegible] ecause it is available in both util and [illegible] packages.

Case 2:
While [illegible] ing class names compiler [illegible] always gives the importance in the following ord[illegible]
1. [illegible] xplicit c[illegible] port
2. Classes prese[illegible] current working director[illegible]
3. [illegible] plicit class i[illegible]

Example:

```
 import java.util.Date;
import java.sql.*;
class Test
 {
          public static void main(String args[])
          {
                    Date d=new Date();
          }
}
```

The code compiles fine and in this case util package Date will be considered.

Case 3:

Whenever we are importing a package all classes and interfaces present in that package are by default available but not sub package classes.

Case 4:
In any java Program the following 2 packages are not require to import because these are available by default to every java Program.
1. java.lang package
2. default package(current working directory)

Case 5:
"Import statement is totally compile time concept" if more no of imports are there then more will be the compile time but there is "no chang[illegible] execution time".

Difference between C language #include an[illegible]ava language import ?
At runtime JVM will execute the correspon[illegible]g standard library and use it's result in current program.
Ex : <jsp:@ file=""> Ex : <jsp:include >

#include	import
It can be used in C & C+[illegible]	It can be used in Jav[illegible]
At compile time only co[illegible]ler copy the code from standard [illegible]d placed in current program	At runtime JVM wil[illegible] execute the corresponding [illegible]rary and use it's result in current [illegible]gram.
It is static inc[illegible]	It is dynamic inclusion
wastage o[illegible]emory	No wastage of memory

- *[illegible]ase of C language #inclu[illegible] the header files will be loaded at the time of in[illegible]statement hence it follow[illegible]ing.*

- *But in ja[illegible]port statement no ".c[illegible]ass" will be loaded at the time of import statements in [illegible]lines of the code whenever we are using a particular class then only correspon[illegible]".class" file will be loaded. Hence it follows "dynamic loading" or "load-[illegible]nd" or "load-on-fly".*

Exception Handling

What is the difference between Error and Exception?

Error	**Exception**
Error is a problem at runtime, forwhich we are unable to providesolutions programmatically.	Exception is a problem, for which, weare able to provide solution programmatically.
Ex: JVM internal Problem StackOverFlowError InSufficientMainMemory	Ex:ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException

Definition

- *Dictionary meaning of the exception is abno*
- **An exception is an event th uring execution of the prog hat disturbs normal flow of the prog n instruc**
- If the application co xception the e program rminated ab rmally then the rest of the application is uted. To me above limi rder to execute the rest of the appl n & get n termi tion of the a cation st handle the exception.

Reasons for exce ons

- *ope g -existing file.*
- *N vork co on problems.*
- *lues are out e values .*
- *End user input m …….etc*

Exception Handling

- **The main objective of exce andling is to get normal termination of the application in order to execu rest of the application code.**
- Exception handling means just we are providing alternate code to continue the execution of remaining code & to get normal termination of the application.
- There are two ways to handle the exceptions in java.
  1) By using try-catch block.
  2) By using throws keyword.

Types of Exceptions

As per the sun micro systems standards The Exceptions are divided into three types

1) Checked Exception 2) Unchecked Exception 3) Error

Checked Exception

- The Exceptions which are checked by the compiler at the time of compilation are called Checked Exceptions.
- Examples:- IOException,SQLException,InterruptedException,ClassNotFoundException… etc
- If the application contains checked Exception the compiler is able to check it and it will give intimation to developer regarding Exception in the form of compilation error.
- Handle the checked Exception in two ways
  - ✔ using try-catch block.
  - ✔ using throws keyword.

Checked Exception scenarios

✔ ***java.lang.InterruptedException***

- ❖ *Thread.sleep(2000); is [illegible]ecuted, [illegible] enters into sleeping mode th[illegible] other threads are able to interru[illegible] program is [illegible] abnormally & rest of the [illegible]plication is not executed.*
- ❖ *To overcome [illegible]bove pro[illegible]ile time [illegible]mpiler is c cking [illegible] exception & displaying ception in[illegible] in the for[illegible]pilation error.*
- ❖ *Based [illegible] compiler generated [illegible]ssage will handle the exception using the try-cat[illegible]cks or throws , if runti[illegible]y exception raised the try-catch or throws k [illegible]ord[illegible]d program is terminat no[illegible]ly.*

✔ ***Ja[illegible].io.FileNotF[illegible]Exception***

- ❖ *Reading the file [illegible] disk but at runtime if the file is not available program is terminated abnorma[illegible]f the application is not executed.*
- ❖ *To overcome above pr[illegible]em compile time compiler is checking that exception & displaying exception [illegible]formation in the form of compilation error.*
- ❖ *Based on compiler generated error message will handle the exception using the try-catch blocks or throws , if runtime any exception raised the try-catch or throws keyword executed program is terminated normally.*

✔ ***Java.sql.SQLException***

- ❖ *Connecting to data base but at runtime data base is not available program is terminated abnormally rest of the application is not executed. Note: In above scenarios compile time compiler is display just exception information but* *exception raised at runtime.*

Unchecked Exception

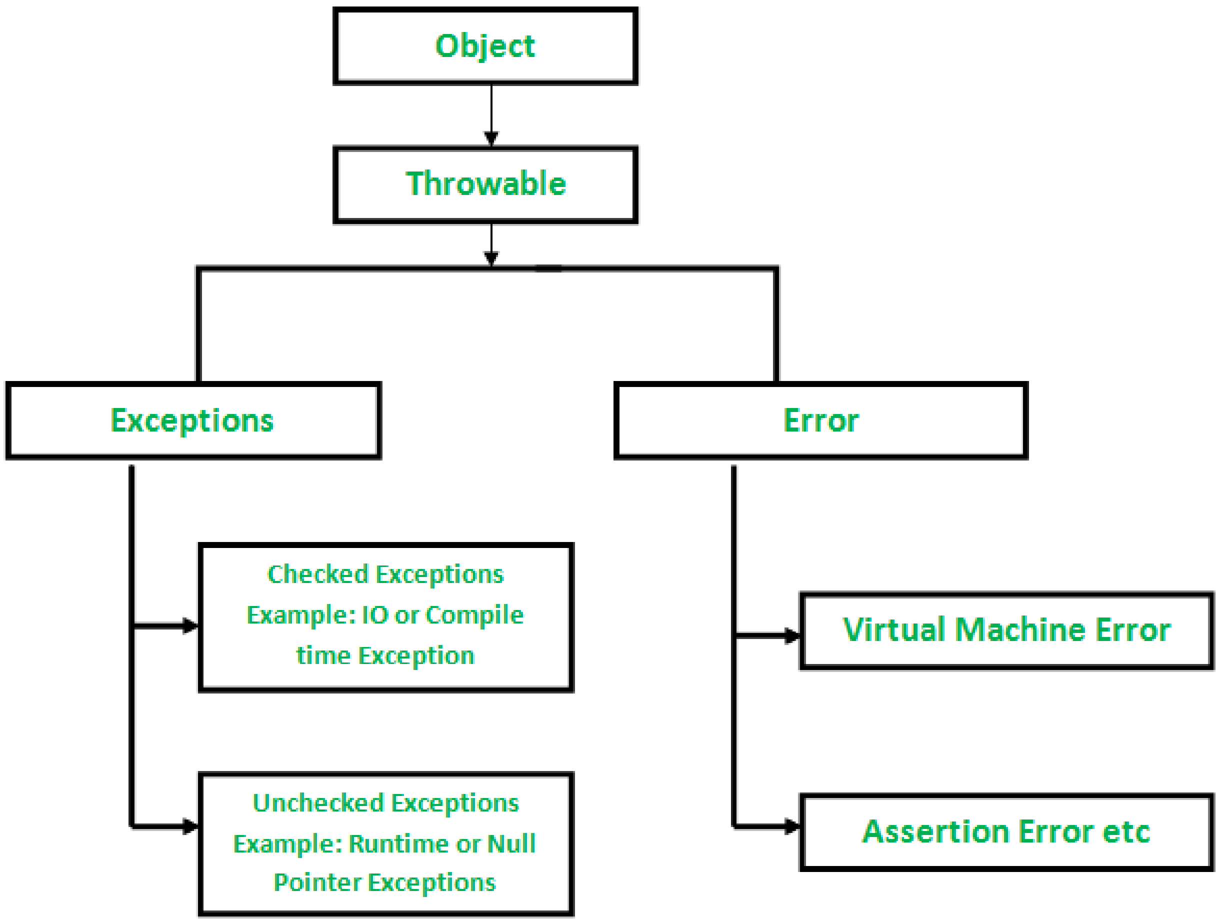
- The exceptions which are not checked by the compiler at the time of compilation are called unchecked Exception .
- Example:
  ArithmeticException,
  ArrayIndexOutOfBoundsException,
  NumberFormatException….etc
- If the application contains un-checked Exception code is compiled but at runtime JVM (Default Exception handler) display exce[illegible]ge then program terminated abnormally.
- To overcome runtime problem must handle the exception in two ways.
  - ❖ using try-catch blocks.
  - ❖ using throws keyword.

```
class Test
{
        public static void [illegible]String[] arg[illegible]
        {
                [illegible]={10,20,3[illegible]
                System.out.println([illegible]
        }
}
```

```
D:\>ja[illegible]est Excepti[illegible]read "main" java.lang.A[illegible]xOutOfBoundsException: 6
```

Note-1:- *Whether it is a c[illegible]eption or unchecked exception exceptions are raised at runtime but not compile time.*
Note 2:- *In java whether it is [illegible]ked Exception or unchecked Exception must handle the Exception by using try-catch blocks [illegible]s keyword to get normal termination of application & to execute rest of the application.*

Exception Handling Tree Structure


Object
Throwable
Exceptions
Error
Checked Exceptions Example: IO or Compile time Exception
Unchecked Exceptions Example: Runtime or Null Pointer Exceptions
Virtual Machine Error
Assertion Error etc

Exception h[illegible]key words

1) [illegible]
[illegible]catch
3) [illegible]inally
4) throw
5) throws

There are two ways to handle the [illegible]tions in java.

1) By using try-catch block.
2) By using throws keyword.

Exception handling by using Try –catch blocks

❖ Syntax:-

```
try
{
        exceptional code;
}
catch (ExceptionName reference_variable)
{
        Code to run if an exception is raised (alternate code);
}
```

❖ Example-1 :- Application without try-catch blocks class Test

```
{
        public static void main(String[] args)
        {
                System.out.println("start");
                System.out.println(10/0);
                System.out.println("rest of the application");
        }
}
```

```
E:\>java Test
        start
        Exception in Thread "main" java.lang.ArithmeticException: / by zero
        {Handled by JVM}  {   type of the Exception   }  {description}
```

In above example exception raised progra [illegible] & rest of the application is not executed

Whenever the exception r[illegible]ed the defa[illegible]ception handler is responsib[illegible] to handle the exception & it is com[illegible] of the JVM.

Application with tr[illegible]locks

Wheneve[illegible]e exception is raised [illegible] try block JVM won't terminate the program imm[illegible] will search corresponding [illegible] block.

a. [illegible] catch block is matched th[illegible] block will be executed & rest of the ap[illegible]lication execu[illegible] program is terminated [illegible]ormally.

b. If the catch [illegible] is not matched program is terminated abnormally.

```
class Test
{
        public static void main(String[] args)
        {
                System.out.println("start");
                try
                {
                        System.out.println(10/0);
                }
                catch (ArithmeticException ae)
                {
                        System.out.println("cannot divide by zero");
```

```
                    }
                    System.out.println("rest of the application");
                }
            }
```

```
E:\>java Test
 start
cannot divide by zero
rest of the application
```

In above example we are handling exception by usi[illegible] try-catch block hence the program is terminated normally & rest of the application [illegible]ed.

Various cases in Exception Handling

Case 1:

Example 1

If there is no exception in [illegible] block the[illegible]sponding catch blocks are no[illegible]ecked.

```
class Test
{
        public[illegible]oid main([illegible][][illegible]gs)
        {
                try
                [illegible]
                        System.out.println("star[illegible]);
                        [illegible]m.out.println(10/5);
                }
                catch(Arthimet[illegible]ception e)
                {
                        System.out.println("arthimetic Exception");
                }
                 System.out.println("rest of the app");
        }
 }
```

Output:

```
s
t
a
r
t
5
rest of the app
```

Case 2

Example 2

In Exception handling independent try blocks declaration are not allowed must declare try-catch or try- finally or try-catch-finally.

```
class Test
{
        public static void main(String[] args)
        {
                try
                {
                        System.o
                        ut.println
                        ("star
                }
                        System.o
                        ut.println
                        ("hello
                        world");
                         System.out.prin    ("rest of       app");
        }
  }
Output:
```

```
javac Test   v
Test.java: try without 'catch'      a
```

Case 3

- ✔ In between try-ca blocks it is not possible to declare any statements, if we are declaring statements compiler will generate error message.
- ✔ In exception handling must lare try with immediate catch block.
- ✔ Example 3

```
class Test
{
        public static void main(String[] args)
        {
                try
                {                                   System.out.println(10/0);

                }

                 System.out.println("Hello world");
                catch(ArithmeticException e)
```

```
                    {
                              System.out.println(10/2);
                     }
                     System.out.println("rest of the app");
          }
 }
```

Case 4

- ❖ If the exception raised in try block jvm will search corresponding catch block but if the exception raised other than try-catch blocks it is al[illegible]bnormal termination.
- ❖ In below example exception raised in catch [illegible] hence program is terminated abnormally.
- ❖ Example 4

  class Test

```
 {
          public st[illegible]c void ma[illegible]ing[]
           {
          try
           [illegible]
                    S stem.out.printl  "start"[illegible]
          [illegible]h(ArithmeticExc      on e)[illegible]
                    {
                    [illegible]tem.out.println(10/0);

                    S stem.ou[illegible]ln("re[illegible] of the pp");
          }
 }
```

Case 5

- ❖ If the exception raised in try block the remaining code of try block is not executed.
- ❖ Once the control is out of the try block the control never entered into try block once again.
- ❖ Don't take normal code inside try block because no guarantee all statements in try-block

will be executed or not.

- Example 5

```
class Test
 {
        public static void main(String[] args)
         {
             try
             {
                        System.out.println(10/0);
                        System.out.println("srtart");
                        System.out.println("stop");
             }
             catch(ArithmeticException e)
             {
                   System.out.println("Arthimetic Exception");
              }
              System.out.println("rest of the app");
        }
    }
```

Output :java Test
Arthimetic Exception
start
 rest of the app

Case 6 : Multiple catch block

- ❖ The way of handling the [illegible]ion is vari[illegible]om exception to [illegible]ception hence it is recommende[illegible] to rovide try with [illegible]e number of catch blocks.
- ❖ Examp[illegible]

  public class MultipleCatchBlock1

```
    {

        public static void main(String[] args)
        {
            try
            {                                     int
                                                  a[]=
                                                  new
            }                                     int[
                                                  5];
                                                  a[5]
                                                  =30
                                                  /0;
            catch(ArithmeticException e)
            {
                System.out.println("Arithmetic Exception occurs");
            }
            catch(ArrayIndexOutOfBoundsException e)
            {
              System.out.println("ArrayIndexOutOfBounds Exception occurs");
            }
            catch(Exception e)
            {
                System.out.println("Parent Exception occurs");
        }
                                                System.out.println("rest of the code");
                }
        }
```

Output:

```
Arithmetic Exception occurs
rest of the code
```

Example 2

```
public class MultipleCatchBlock2
{
        public static void main(String[] args)
        {
                try
                {
                                int a[]=new int[5];
                                System.out.prin
                }
tl                    1
n                     0
(                     ]
a                     );
[
            catch(ArithmeticException e)
            {
                System.out.println("Arithmetic Exception occurs");
            }
            catch(ArrayIndexOutOfBoundsException e)
            {
              System.out.println("ArrayIndexOutOfBounds Exception occurs");
             }
            catch(Exception e)
            {
                System.out.println("Parent Exception occurs");
            }
            System.out.println("rest of the code");
        }
    }
```

Case 7

- When we declare multiple catch blocks then the catch block order must be child-parent but if we are declaring parent to child compiler will generate error message.
- No compilation error (catch block order child to parent type)
- Example 7

  public class MultipleCatchBlock4

```
{
        public static void main(String[] args)
        {
                try
                {                                       String s=null;
                                                        System.out.println
                                                        (s.length());
                }

                catch(ArithmeticException e)
                {
                        System.out.println("Arithmetic Exception occurs");
                }
                catch(ArrayIndexOutOfBoundsException e)
                {
                     System.out.println("ArrayIndexOutOfBounds Exception occurs");
                }
                catch(Exception e)
                {
                        System.out.println("Parent Exception occurs");
                }
                System.out.println("rest of the code");
        }
}
```

Case 8

- There are three methods to print Exception information
  - toString(): used to display name and description of exception.
  - getMessage(): used to display description of the exception.
  - printStackTrace used to display name, description and stack trace(location). In other words, it is used to display complete information of exception.
- Example 8

```
class Test
{
        void m1()
        {
                m2();
        }
        void m2()
```

```
{
```

```
          m3();
     }
     void m3()
     {
          try
          {
                              System.out.println(10/0);

          }

          catch(ArithmeticException a[illegible]
          {
               System.out.println(ae.[illegible]ing())
               System.ou[illegible]intln(ae.g[illegible]
               ae.pr[illegible]S[illegible]
          }
     }
     public [illegible]void ma[illegible]g[] a[illegible]
     {
          Test1 t = new Test1();
          [illegible]1();
     }
};
java.lang.ArithmeticEx[illegible]: / by zero       //toString() method output
/ by zero                                          //getMessage() method output
 java.lang.ArithmeticExceptio[illegible]: by zero  //printStackTrace() method
```

Note : internally JVM used printStackTrace() to print exception information.

Finally Block

- It is used to execute important code such as closing connection, stream etc.
- Java finally block is always executed whether exception is handled or not.

- Java finally block follows try or catch block.

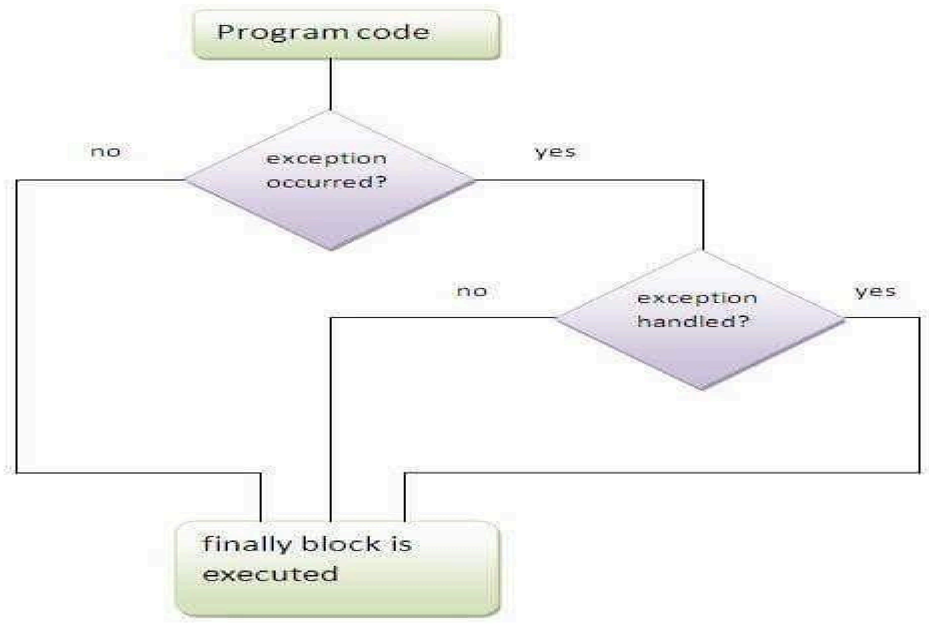
- **Syntax :**

```
try
 {
        critical code;
 }
catch (Exception obj)
 {
        code to be run if the exception raised (handling code);
}
 finally
{
        Clean-up code;(database connection closing , streams closing……etc)
}
```

- Example :

```
class TestFinallyBlock1
{
        public static void main(String ar      )
        {
                try
                {
                        int data=25/0
                        System.out.printl data);
                }
                catch(NullPointerException e)
                {
                        ystem.out.println(e);
                }
                finally
                {
                        System.out.println("finally block is always executed");
                }

                System.out.println("rest of the code...");
        }
```


Program code
exception occurred?
no
yes
exception handled?
no
yes
finally block is executed

}