**ASSIGNMENT NO.-G-29**

**Title:**

Write a program on template and exception handling in Java: in this assignment multiple templates are to be designed as a pattern and these patterns to be used to take decisions.

**Objectives:**

1. To understand concept of Exception.

2. To know various ways of handling exceptions.

**Facilities:** java, 64 bit Fedora, eclipse IDE

**Theory:**

The process of converting system error messages into user friendly error message is known as **Exception handling**. This is one of the powerful feature of Java to handle run time error and maintain normal flow of java application.

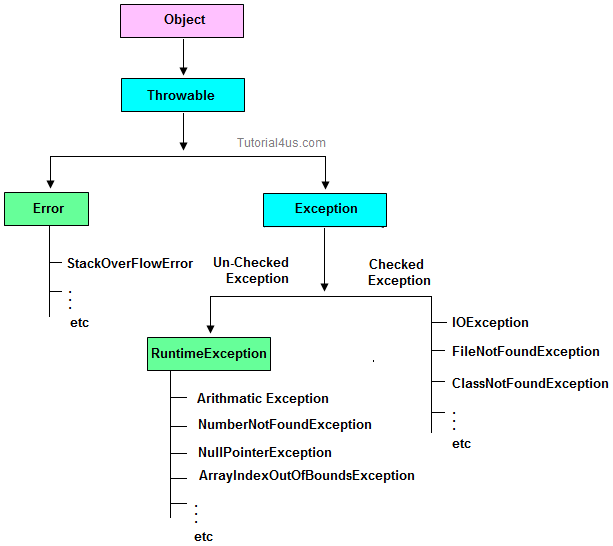
**Exception**

An **Exception** is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's Instructions.

**Why use Exception Handling**

Handling the exception is nothing but converting system error generated message into user friendly error message. Whenever an exception occurs in the java application, JVM will create an object of appropriate exception of sub class and generates system error message, these system generated messages are not understandable by user so need to convert it into user friendly error message. You can convert system error message into user friendly error message by using exception handling feature of java.  
For Example: when you divide any number by zero then system generate **/ by zero** so this is not understandable by user so you can convert this message into user friendly error message like **Don't enter zero for denominator.**

**Hierarchy of Exception classes**

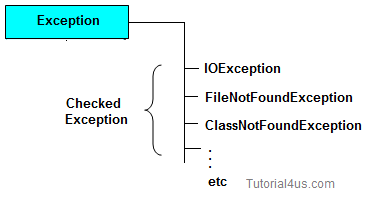
**Type of Exception**

* Checked Exception
* Un-Checked Exception

**Checked Exception**

**Checked Exception** are the exception which checked at compile-time. These exception are directly sub-class of java.lang.Exception class.

**Only for remember:**Checked means checked by compiler so checked exception are checked at compile-time.

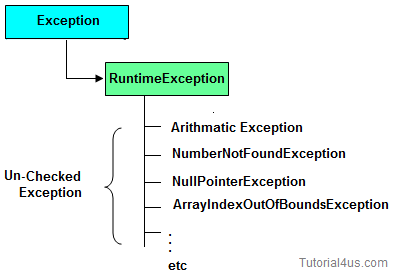


**Un-Checked Exception**

**Un-Checked Exception** are the exception both identifies or raised at run time. These exception are directly sub-class of java.lang.RuntimeException class.

**Note:**In real time application mostly we can handle un-checked exception.

**Only for remember:**Un-checked means not checked by compiler so un-checked exception are checked at run-time not compile time.



**Difference between checked Exception and un-checked Exception**

|  |  |  |
| --- | --- | --- |
|  | **Checked Exception** | **Un-Checked Exception** |
| 1 | checked Exception are checked at compile time | un-checked Exception are checked at run time |
| 3 | e.g.  FileNotFoundException, NumberNotFoundException etc. | e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. |

**Difference between Error and Exception**

|  |  |  |
| --- | --- | --- |
|  | **Error** | **Exception** |
| 1 | Can't be handle. | Can be handle. |
| 2 | Example: NoSuchMethodError OutOfMemoryError | Example: ClassNotFoundException NumberFormateException |

**Handling the Exception**

Handling the exception is nothing but converting system error generated message into user friendly error message in others word whenever an exception occurs in the java application, JVM will create an object of appropriate exception of sub class and generates system error message, these system generated messages are not understandable by user so need to convert it into user-friendly error message. You can convert system error message into user-friendly error message by using exception handling feature of java.

**Use Five keywords for Handling the Exception**

* try
* catch
* finally
* throws
* throw

Syntax for handling the exception

**Syntax**

**try**

{

// statements causes problem at run time

}

**catch**(type of exception-1 **object**-1)

{

// statements provides user friendly error message

}

**catch**(type of exception-2 **object**-2)

{

// statements provides user friendly error message

}

**finally**

{

// statements which will execute compulsory

}

**Example without Exception Handling**

**Syntax**

**class** ExceptionDemo

{

**public** **static** **void** main(String[] args)

{

**int** a=10, ans=0;

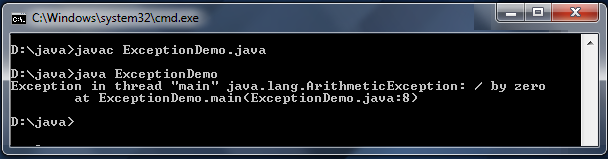
ans=a/0;

System.**out**.println("Denominator not be zero");

}

}

Abnormally terminate program and give a message like below, this error message is not understandable by user so we convert this error message into user friendly error message, like "denominator not be zero".



**Example of Exception Handling**

**Example**

**class** ExceptionDemo

{

**public** **static** **void** main(String[] args)

{

**int** a=10, ans=0;

**try**

{

ans=a/0;

}

**catch** (Exception e)

{

System.**out**.println("Denominator not be zero");

}

}

}

**Output**

Denominator not be zero

**Conclusion:** This program gives us the knowledge about exceptions & exception handling mechanisms.

**Questions asked in university exam.**

1. What is Exception?
2. Explain user defined exception with example.
3. Explain Exception handling mechanisms.