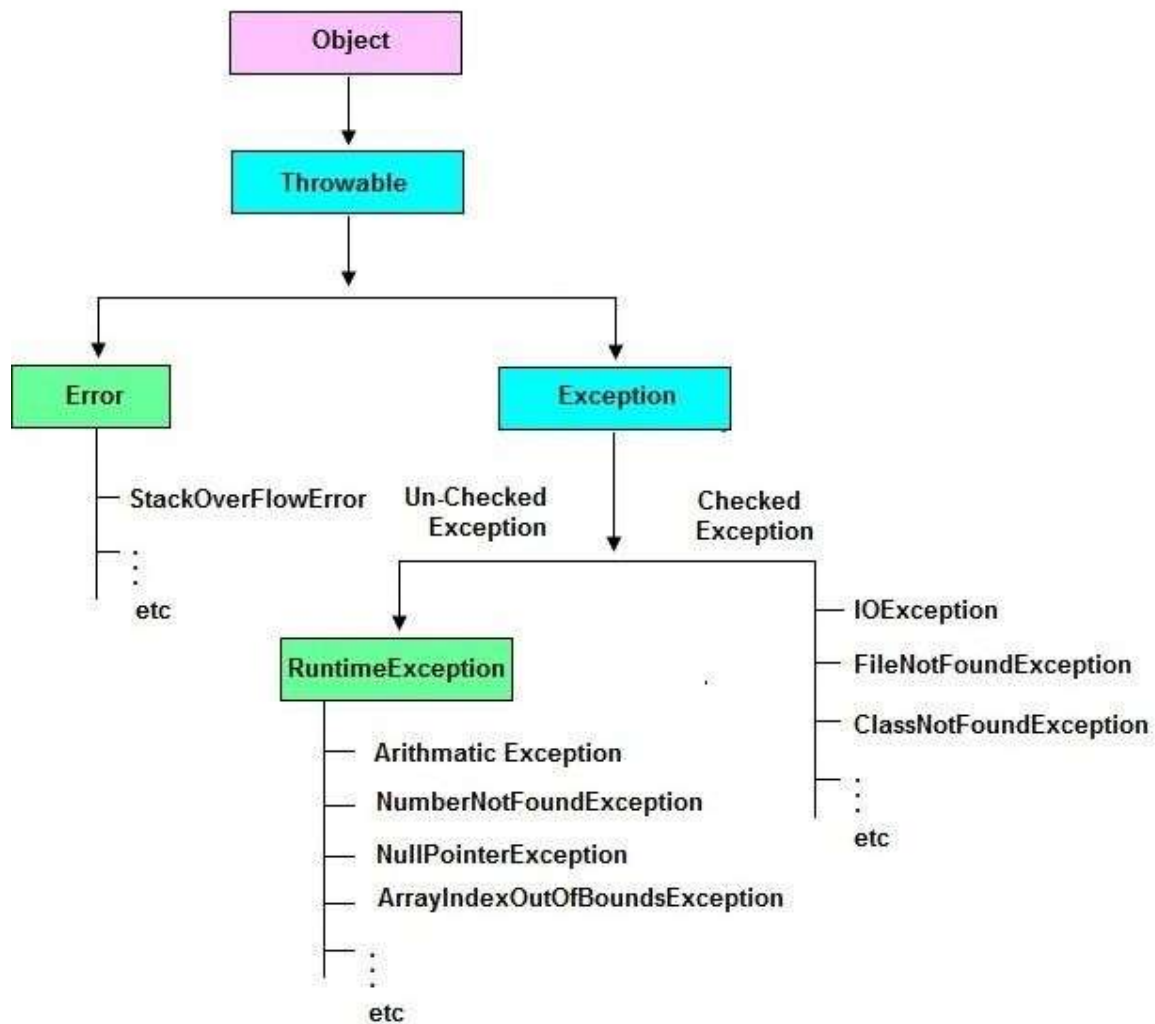


Hierarchy of Exception classes



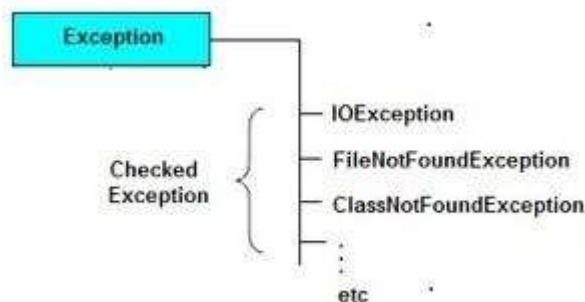
Types of Exceptions

- 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.



Checked Exception Classes

- FileNotFoundException
- ClassNotFoundException
- IOException

- InterruptedException

FileNotFoundException

If the given filename is not available in a specific location (in file handling concept) then FileNotFoundException will be raised. This exception will be thrown by the FileInputStream, FileOutputStream, and RandomAccessFile constructors.

ClassNotFoundException

If the given class name is not existing at the time of compilation or running of program then ClassNotFoundException will be raised. In other words this exception is occurred when an application tries to load a class but no definition for the specified class name could be found.

IOException

This exception is raised whenever problem occurred while writing and reading the data in the file. This exception is occurred due to following reason;

- When try to transfer more data but less data are present.
- When try to read data which is corrupted.
- When try to write on file but file is read only.

InterruptedException

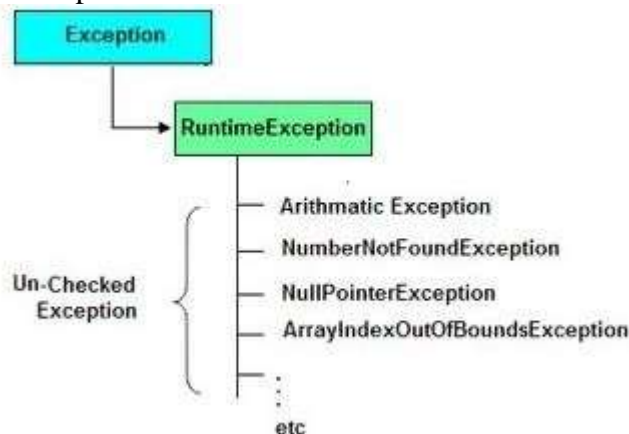
This exception is raised whenever one thread is disturb the other thread. In other words this exception is thrown when a thread is waiting, sleeping, or otherwise occupied, and the thread is interrupted, either before or during the activity.

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.



Un-Checked Exception Classes

- ArithmeticException
- ArrayIndexOutOfBoundsException
- StringIndexOutOfBoundsException
- NumberFormatException
- NullPointerException
- NoSuchMethodException
- NoSuchFieldException

ArithmeticException

This exception is raised because of problem in arithmetic operation like divide by zero. In other words this exception is thrown when an exceptional arithmetic condition has occurred. For example, an integer "divide by zero".

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");
}
}
}
```

ArrayIndexOutOfBoundsException

This exception will be raised whenever given index value of an array is out of range. The index is either negative or greater than or equal to the size of the array.

Example

```
int a[]=new int[5];
a[10]=100; //ArrayIndexOutOfBoundsException
StringIndexOutOfBoundsException
```

This exception will be raised whenever given index value of string is out of range. The index is either negative or greater than or equal to the size of the array.

Example

```
String s="Hello";
s.charAt(3);
s.charAt(10); // Exception raised
charAt() is a predefined method of string class used to get the individual characters based on index value.
```

NumberFormatException

This exception will be raised whenever you trying to store any input value in the unauthorized datatype.

Example: Storing string value into int datatype.

Example

```
int a;
a="Hello";
Example
String s="hello";
int i=Integer.parseInt(s); //NumberFormatException
```

NoSuchMethodException

This exception will be raised whenever calling method is not existing in the program.

NullPointerException

A NullPointerException is thrown when an application is trying to use or access an object whose reference equals to null.

Example

```
String s=null;
```

```
System.out.println(s.length());//NullPointerException
```

StackOverflowException

This exception throw when full the stack because the recursion method are stored in stack area.

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, FileNotFoundException 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 NumberFormatException

Custom Exception in Java

If any exception is design by the user known as user defined or Custom Exception. Custom Exception is created by user.

Rules to design user defined Exception

1. Create a package with valid user defined name.
2. Create any user defined class.
3. Make that user defined class as derived class of Exception or RuntimeException class.
4. Declare parametrized constructor with string variable.
5. call super class constructor by passing string variable within the derived class constructor.
6. Save the program with public class name.java

```
// AgeException.java  
package pack;  
  
public class AgeException extends Exception  
{  
    public AgeException(String s)  
    {  
        super(s);  
    }  
}
```

```
graph LR
    L1[// AgeException.java] --> C6((6))
    L2[package pack;] --> C1((1))
    L3[public class AgeException extends Exception] --> C2((2))
    L3 --> C3((3))
    L4[public AgeException(String s)] --> C4((4))
    L5[super(s);] --> C5((5))
    L6[}] --> C6
```

Example

```
// save by AgeException.java
package nage;
```

```
public class AgeException extends Exception
{
    public AgeException(String s)
    {
        super(s);
    }
}
```

Compile: javac -d . AgeException.java

Example

```
// save by CheckAge.java
package nage;
```

```
public class CheckAge
{
    public void verify(int age)throws AgeException
    {
        if (age>0)
        {
            System.err.print("valid age");
        }
        else
        {
            AgeException ae=new AgeException("Invalid age");
            throw(ae);
        }
    }
}
```

Compile: javac -d . CheckAge.java

Example

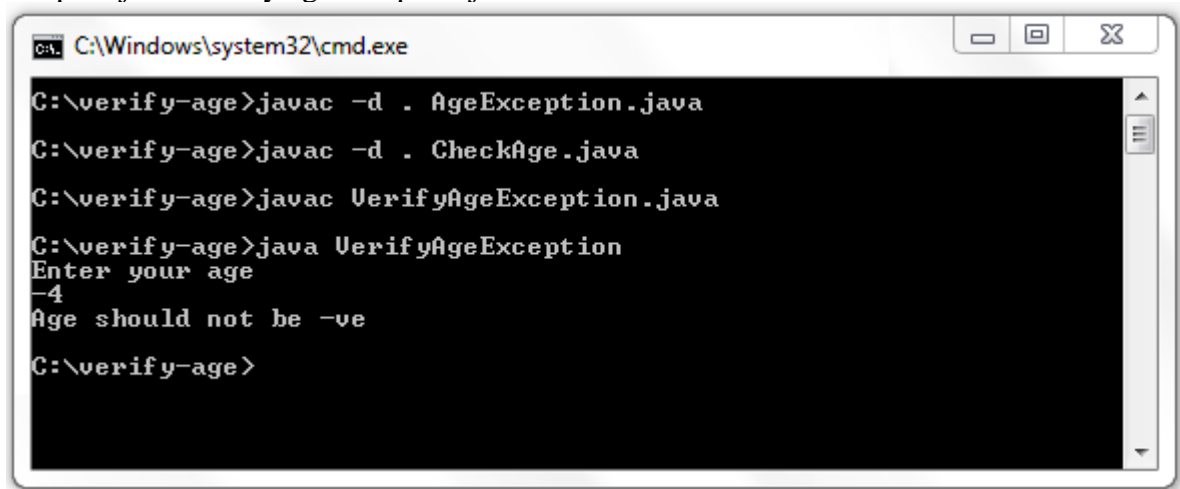
```
// save by VerifyAgeException
```

```
import nage.AgeException;
import nage.CheckAge;
import java.util.*;
```

```
public class VerifyAgeException
{
    public static void main(String args[])
    {
        int a;
        System.out.println("Enter your age");
        Scanner s=new Scanner(System.in);
        a=s.nextInt();
        try
        {
            CheckAge ca=new CheckAge();
```

```
ca.verify(a);
}
catch(AgeException ae)
{
System.err.println("Age should not be -ve");
}
catch(Exception e)
{
System.err.println(e);
}
}
}
```

Compile: javac VerifyAgeException.java



```
C:\Windows\system32\cmd.exe

C:\verify-age>javac -d . AgeException.java
C:\verify-age>javac -d . CheckAge.java
C:\verify-age>javac VerifyAgeException.java
C:\verify-age>java VerifyAgeException
Enter your age
-4
Age should not be -ve
C:\verify-age>
```