Exception Handling Self notes

Advanced

\*In exception handling only the method in which the exception is occurred creates the object of exception

\*and this object will have

2.Description:> message

Part

3.StackTrace:> Location where

Exception happening

\*This object will be passed to default exception handler (if there is no catch means no exception handler)

**JVM**

JVM will ask the method that is it handling the

Exception or not,

If it is not handling the exception

Then it will look to calling method

Which is calling this method

Then if the calling method is being called from somewhere it will look for that calling method **in our case** calling method is **main** method which is directly being called by **JVM** so again control return to JVM

Then **jvm** understands that Exception is not handling then **jvm** pass created object to **default exception handler** and this (default exception handler) will print that object. This will terminate main method abnormally

\*Note that where ever you handle exception the StackTrace will remain same (all path to call the method where the exception is occurred)

\***throw** keyword:> throw keyword pass the programmer made exception to **JVM** unlike **try** and **catch** blocks here is no role of any method to create the exception **object.**

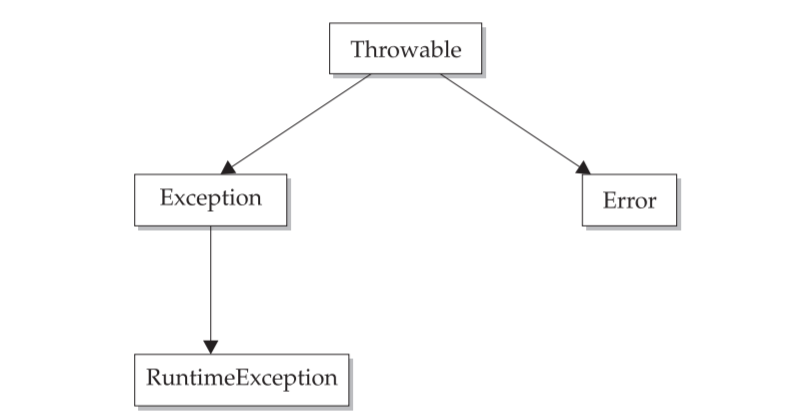
Also **throw** keyword is not recommended for **predefined exceptions** cuz this work can be done by the method already so in this context **throw** becomes useless.

**Throw**  is used to for **custom exceptions /user define exceptions**

***Sysntax:***

**throw new** exception\_name

**Types of exceptions**

****

**Checked:** class class\_name **extends** Exception

If we want to create checked exception

**Unchecked:** class class\_name **extends** RuntimeException

If we want to create unchecked Exception

**\*Note:** Remember that when we create **exception** through **throw** keyword it is always better to create **unchecked** exceptions

\***Note:** if you creates a custom Exception class then its your responsibility to create the object(through **throw**) of that class **no method or JVM will create it**

import java.util.Scanner;

class **NotAligible extends RuntimeException**

{

**NotAligible(String s)**

**{**

**super(s);** //Passing to default handler if nobody wil catch this exception

**}**

}

public class Practice

{

public static void main(String arg[])

{

Scanner sc = new Scanner(System.in);

int i;

System.out.println("Enter Here: ");

i= sc.nextInt();

if(i<15)

{

**throw new NotAligible("You're under age");**  //its like passing to Constructor of exception class

}

else

{

System.out.println("You're Aligible");

}

}

}

**Finally Block**

Finally block is used as cleanup

**Try**

{

//Risky code

}

**Catch**

{

//Handling Code

}

**Finally**

{

// Clean up code

}

\***finally** block will execute even if exception is occurred or not even when there is no **catch** block fond and exception goes to **default handler**

public class Practice

{

public static void main(String arg[])

{

int i= 4;

int b=0;

int arr[]=new int[2];

try

{

arr[5]=9;

i = i/b;

}

catch(ArithmeticException E) // But the exception will be

ArrayIndexOutOfBounds

{

System.out.println("this is exception");

//System.out.println(E);

}

finally { //but still finally block get executed

System.out.println("this should get print");

}

System.out.println("This print");

}

}