**throws Keyword**

Any method that is capable of causing exceptions must list all the exceptions possible during its execution, so that anyone calling that method gets a prior knowledge about which exceptions are to be handled. A method can do so by using the throws keyword. **Exception Handling is mainly used to handle the checked exceptions.**

class Test

{

static void check() throws ArithmeticException

{

System.out.println("Inside check function");

throw new ArithmeticException("demo");

}

public static void main(String args[])

{

try

{

check();

}

catch(ArithmeticException e)

{

System.out.println("caught" + e);

}

}

}

=====================================================================

**when we handle the exception**

import java.io.\*;

class M{

void method()throws IOException

{

throw new IOException("device error");

}

}

public class Throws2{

public static void main(String args[]){

try{

M m=new M();

m.method();

}

catch(Exception e)

{

System.out.println("exception handled");

}

System.out.println("normal flow...");

}

}

==================================================================

**Program if exception does not occur**

import java.io.\*;

class M{

void method()throws IOException{

System.out.println("device operation performed");

}

}

class Throws3{

public static void main(String args[])throws IOException

{ **//declare exception**

M m=new M();

m.method();

System.out.println("normal flow...");

}

}

=====================================================================

**Program if exception occurs**

import java.io.\*;

class M{

void method()throws IOException{

throw new IOException("device error");

}

}

class Throws4{

public static void main(String args[])throws IOException{ //declare exception

M m=new M();

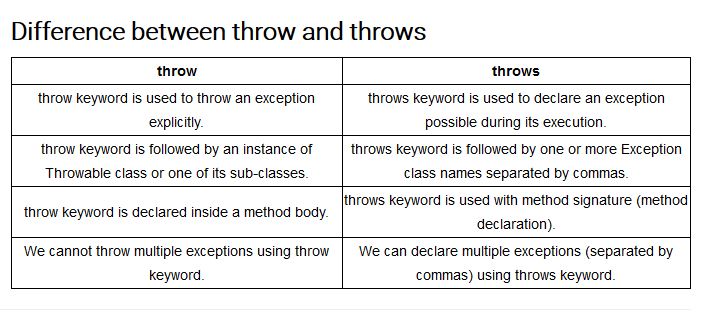
m.method();

System.out.println("normal flow...");

}

} // RTE

=====================================================================================

****

===================================================================

**Rule 1:** **If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception.**

import java.io.\*;

class Parent{

void msg()

{

System.out.println("parent");

}

}

class ExceptionChild1 extends Parent

{

void msg()throws IOException

{

System.out.println("TestExceptionChild");

}

public static void main(String args[]){

Parent p=new ExceptionChild1();

p.msg();

}

} //CTE

=====================================================================

**Rule 2**: **If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception but can declare unchecked exception.**

import java.io.\*;

class Parent{

void msg()

{

System.out.println("parent");

}

}

class ExceptionChild2 extends Parent{

void msg()throws ArithmeticException

{

System.out.println("child");

}

public static void main(String args[]){

Parent p=new ExceptionChild2();

p.msg();

}

}

====================================================================

**Rule 3: If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.**

import java.io.\*;

class Parent{

void msg()throws ArithmeticException

{

System.out.println("parent");

}

}

class ExceptionChild3 extends Parent{

void msg()throws Exception

{

System.out.println("child");

}

public static void main(String args[]){

Parent p=new ExceptionChild3();

try{

p.msg();

}catch(Exception e){}

}

}

//CTE

====================================================================

**Rule4:** **subclass overridden method declares same exception**

import java.io.\*;

class Parent{

void msg()throws Exception{

System.out.println("parent");

}

}

class ExceptionChild4 extends Parent{

void msg()throws Exception{

System.out.println("child");

}

public static void main(String args[]){

Parent p=new ExceptionChild4();

try{

p.msg();

}catch(Exception e){}

}

}

====================================================================