**Exception Handling – Part-07 – throws keyword**

* **throws keyword:**

In our program if there is a possibility of raising checked exception then compulsory we should handle that checked exception otherwise we will get compile time error saying

Unreported exception XXX; must be caught or declared to be thrown.

Example\_01:

import java.io.\*;

class Test{

public static void main(String[] args){

PrintWriter pw = new PrintWriter(“abc.txt”);

pw.println(“Hello”);

}

}

CE: unreported exception java.io.FileNotFoundException; must be caught or declared to be thrown.

Example\_02:

import java.io.\*;

class Test{

public static void main(String[] args){

Thread.sleep(10000);

}

}

CE: unreported exception java.lang.InterruptedException; must be caught or declared to be thrown.

Note:

We can handle this compile time error by using the following two ways:

1. By using try/catch.

class Test{

public static void main(String[] args){

try{

Thread.sleep(10000);

} catch(InterruptedException e){

}

}

}

1. By using throws keyword.

* **By using throws keyword:**

We can use throws keyword to delegate responsibility of exception of handling to the caller (it may be another method or JVM). Then caller method is responsible to handle that exception.

Example:

class Test{

public static void main(String[] args) throws InterruptedException{

Thread.sleep(10000);

}

}

Analogy:

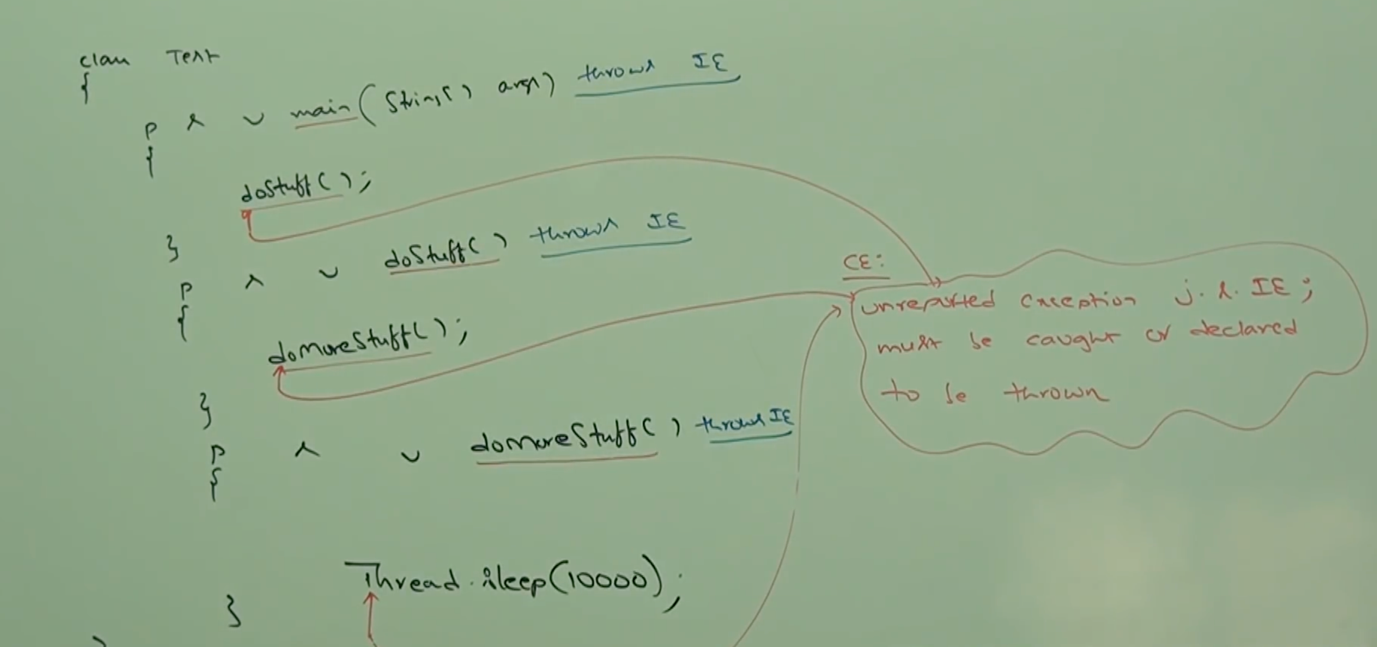
If someone assign a task to us, we can achieve it in two ways:

1. Complete the work by yourself. (catch block)
2. Delegate it to someone. (throws keyword)

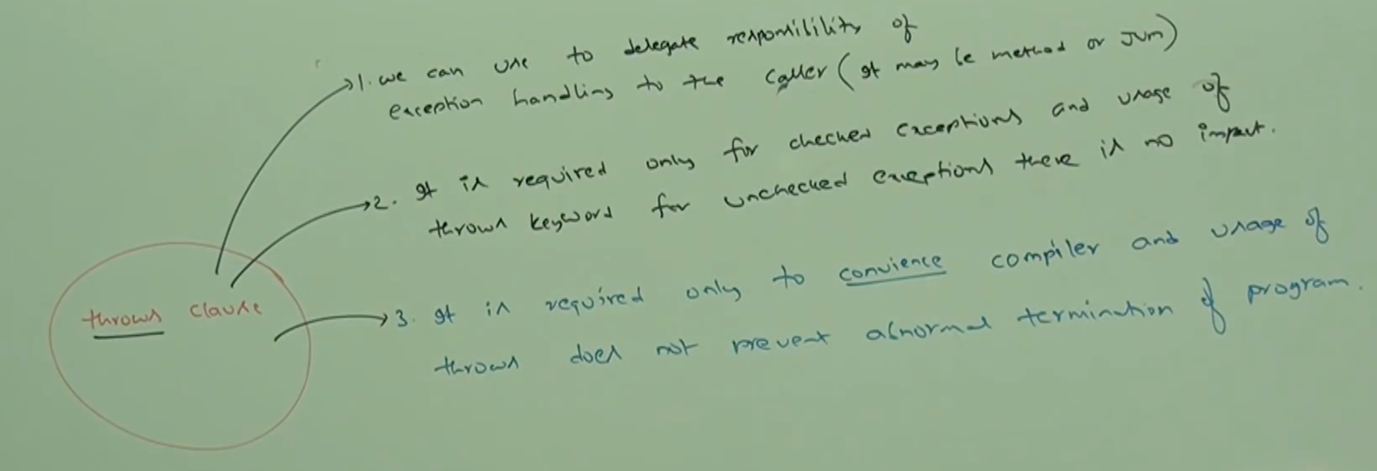
Conclusions:

throws keyword required only for checked exceptions and usage of throws keyword for unchecked exceptions there is no use or impact.

throws keyword required only to convince compiler and usage of throws keyword doesn’t prevent abnormal termination of the program.



In the above program, if we remove at least one throws statement then the code won’t compile.



Note:

It is recommended to use try/catch over throws keyword.

* **Case\_01:**

We can use throws keyword for methods and constructors but not for classes.

Example:

class Test throws Exception{

Test() throws Exception{

}

public void m1() throws Exception{

}

}

* **Case\_02:**

We can use throws keyword only for Throwable types, if we are trying to use for normal Java classes then we will get compile time error saying: incompatible types

Example:

class Test{

public void m1() throws Test{

}

}

CE: incompatible types

found:Test

required: throwable

class Test extends RuntimeException{

public void m1()throws Test{

}

}

* **Case\_03:**

class Test{

public static void main(String[] args){

throw new Exception();

}

}

CE: unreported exception

Must be caught or declared to be thrown.

class Test{

public static void main(String[] args){

throw new Error();

}

}

RE: Exception in thread “main”: java.lang.Error

at Test.main();

* **Case\_04:**

Within the try block if there is no chance of raising an exception, then we can’t write catch block for that exception otherwise we will get compile time error saying

Exception XXX is never thrown in body of corresponding try statement.

But this rule is applicable only for fully checked exceptions.

Example:

class Test{

public static void main(String[] args){

try{

System.out.println(“Hello”);

} catch(ArithmeticException ae){

}

}

}

Output: Hello

class Test{

public static void main(String[] args){

try{

System.out.println(“Hello”);

} catch(Exception e){

}

}

}

Output: Hello

class Test{

public static void main(String[] args){

try{

System.out.println(“Hello”);

} catch(IOException ie){

}

}

}

CE: exception java.io.IOException is never thrown in the body of corresponding try statement.

class Test{

public static void main(String[] args){

try{

System.out.println(“Hello”);

} catch(InterruptedException ie){

}

}

}

CE: exception java.lang.InterruptedException is never thrown in the body of corresponding try statement.

class Test{

public static void main(String[] args){

try{

System.out.println(“Hello”);

} catch(Error e){

}

}

}