***Exception handling:***

1. An unwanted, unexpected event which disturbs a normal flow of the program is called Exception.
2. For example you have a database connection and you want to read the data from it through sql and an exception arises in query and hence it stops their only and will not close the connection with the database. Suppose if multiple users are going to access it then multiple connection will not be closed and may be the limit of database connection crosses ,then this will lead to total shutdown of an application.
3. The main objective of exception handling is the normal termination of the program.
4. Having an alternative way to continue any work is the concept of exception handling.

***Runtime stack mechanism:***

1. For every thread JVM creates one runtime stack.
2. Main method is executed by main thread.

class Test

{

Psvm(String args[])// main thread and first entry in the stack

{

DoStuff();

}

Psv doStuff()//Second entry in the stack

{

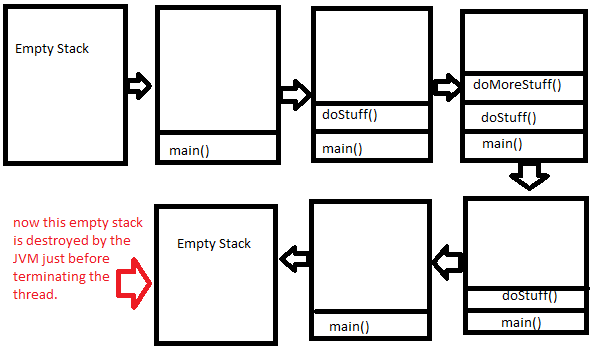
doMoreStuff();

}

Psv doMoreStuff()//Third entry in the stack

{  
syso(“Hello”)

}



In the above there is a normal flow of a program. Each Entry in the stack is called activation record of stack frame.

class Test

{

Psvm(String args[])// main thread and first entry in the stack

{

DoStuff();

}

Psv doStuff()//Second entry in the stack

{

doMoreStuff();

}

Psv doMoreStuff()//Third entry in the stack

{  
syso(5/0);

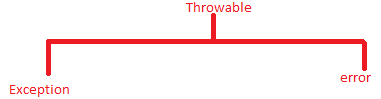
}

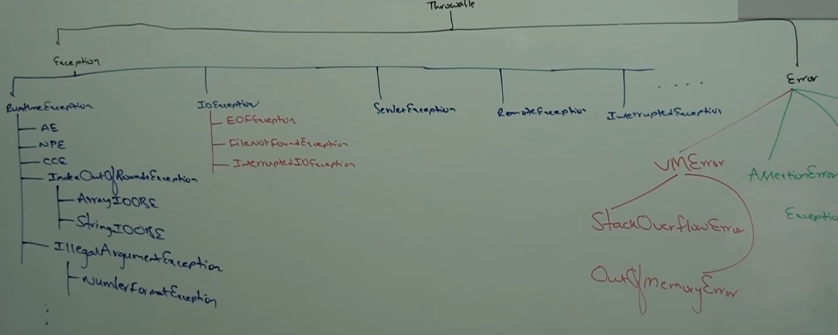
In the above the arithmetic exception arises and is terminated by the JVM abnormally and default exception handler is called because it has not been handled.

Default Exception handler just print the information of the exception and terminates the program abnormally.

As the exception arises in doMoreStuff() and hence it is responsible for creating an Exception Object and hand it over to JVM.

The flow in this will go back from doMoreStuff() to doStuff() and finally to main() which is called by the JVM. And if main() method also does not contain any handling JVM also terminates abnormally and remove its corresponding entry in stack and pass it to default Exception handler which is a part of JVM. Hence the information is like this: Exception in thread main: java.lang.ArithmeticException





***THROWS KEYWORD:***

1. It passes the responsibility of exception handling to a caller. The caller may by JVM or may be any method.
2. Throws keyword is required for checked Exception. For unchecked it is meaningless (If u want to use then u can use).
3. Even though you use throws keyword the execution is abnormal. It means that it can’t prevent abnormal termination.

So try catch is recommended than to throws.

class A

{

psvm(String a[])throws InterruptedException

{

Thread.sleep(1000);

}

In the above the caller is JVM.

class Test

{

Psvm(String args[])throws InterrptedException

{

DoStuff();

}

Psv doStuff()throws InterrptedException

{

doMoreStuff();

}

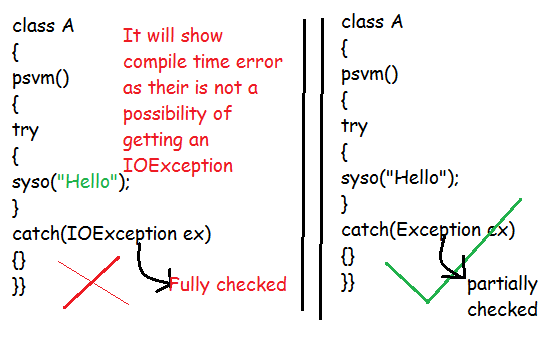
Psv doMoreStuff()throws InterrptedException

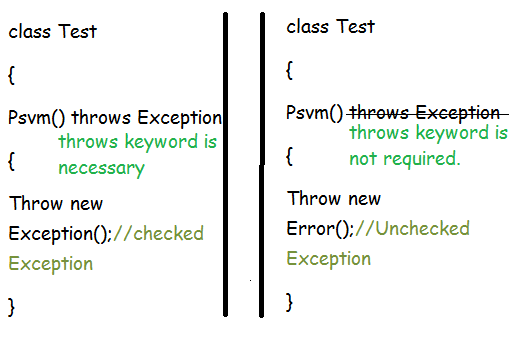
{  
Thread.sleep(1000); //Checked Exception and hence must be handled

}

As in the above there is a checked Exception, it must be handled either by try/catch or by throws keyword.

In the above doMoreStuff() tells compiler that why you are asking me about it ,please ask to my caller about the exception . The caller of doMoreStuff() is doStuff() and hence compiler asks doStuff() about the Exception and it also tells the same to the compiler .Now the caller of the doStuff() method is main() method and main() method pass this Exception to the JVM which further pass it to the default Exception handler.

1. Exception is a partially checked Exception.



CUSTOMIZED