Exceptions

if you have one or more catch blocks, they must

immediately follow the try block. It is a requirement

1. try {

2. // This is the first line of the "guarded region"

3. // that is governed by the try keyword.

4. // Put code here that might cause some kind of exception.

5. // We may have many code lines here or just one.

6. }

7. catch(MyFirstException) {

8. // Put code here that handles this exception.

9. // This is the next line of the exception handler.

10. // This is the last line of the exception handler.

11. }

12. catch(MySecondException) {

13. // Put code here that handles this exception

14. }

15.

16. // Some other unguarded (normal, non-risky) code begins here

Execution of the guarded region starts at line 2. If the program executes all the

way past line 5 with no exceptions being thrown, execution will transfer to line 15

and continue downward. However, if at any time in lines 2 through 5 (the try

block) an exception of type MyFirstException is thrown, execution will

immediately transfer to line 7. Lines 8 through 10 will then be executed so that the

entire catch block runs, and then execution will transfer to line 15 and continue.

Note that if an exception occurred on, say, line 3 of the try block, the rest of the

lines in the try block (4 and 5) would never be executed. Once control jumps to

the catch block, it never returns to complete the balance of the try block.

***Finally***

Finally block used as cleanup code such as closing a network socket.

A finally block encloses code that is always executed at some point after the

try block, whether an exception was thrown or not. Even if there is a return

statement in the try block, the finally block executes right after the return

statement is encountered and before the return executes!

If an exception is thrown,

finally runs.

If an exception is not thrown, finally runs. If the exception is caught, finally runs. If the exception is not caught, finally runs.

If there is a try block, it *must* be followed with a catch or finally.

***A try clause by itself will result in a compiler error.***

Legal Code:

try {

// do stuff

} catch (SomeException ex) {

// do exception handling

} finally {

// clean up

}

Illegal Code:

try {

// do stuff

}

// need a catch or finally here

System.out.println("out of try block");

***You can keep throwing an exception down through the methods on the***

***stack. But what happens when you get to the main() method at the bottom? You can***

***throw the exception out of main() as well. This results in the JVM halting, and the stack***

***trace will be printed to the output. The following code throws an exception:***

