# Lab 3

## Step One

First, create an empty directory for lab3. There is no starter code for this lab.

You will be throwing and catching exceptions in this exercise.

Create a file called RuntimeException.h and put the following code in it.

```
#include <string>

class RuntimeException {
private:
    string errorMsg;
public:
    RuntimeException(const string& err) { errorMsg = err; }
    string getMessage() const { return errorMsg; }
}
```

## Step Two

In a new .cpp file in your directory, write a main function that will call another function test1. Test1 should return **void** and take no arguments.

In the test1 function, put a try/catch statement. Inside the try block, first throw a RuntimeException, giving it some error message of your choice. You will need to #include "RuntimeException.h" at the top of the file in order to do this.
After the throw statement, print the string "failed to throw in test1" and an endline to *cout*. In the what-to-catch part of the catch clause, put *RuntimeException& exception*. In the catch body, print out "passed. " and the error message from the runtime exception.

For this time, I will give you what the code should look like, but in the rest of the exercise I won't. Part of the exercise is interpreting my words into code.

```
void test1() {
    try {
        throw RuntimeException("Replace this error message");
        cout << "failed to throw in test1" << endl;
    }
    catch (RuntimeException& exception) {
        cout << "passed. " << exception.getMessage() << endl;
    }
}
```

Compile and run your program to make sure it passes the test.

## Step Three

Now create another void function called test2. test2 should have a for loop that takes i from 1 to 10. Inside the for loop, put a try/catch statement. Make the try part throw a RuntimeException if i is equal to 4. Again catch a RuntimeException. When you catch the exception (in the catch body) print out "passed at i=" followed by the value of i, followed by a space and then the exception's error message. After printing this, **return** from the function test2. Finally, after the for loop, print "failed to throw in test2".

Now add a call to test2 after the call to test1 in the function main. Compile and run your program and make sure it passes both tests. Also make sure it prints "passed at i=4 …" for the second test.

## Step Four

Create a subclass of RuntimeException called LaboratoryException. Be sure that LaboratoryException's constructor accepts a string argument that it then passes to RuntimeException's constructor as part of its initializers.

Now include that subclass's .h file in your main file. Create a function test3 that is a copy of test2, except that it throws a LaboratoryException and reports a failure to throw in test3. It should still catch a RuntimeException. Add test3 to the tests called from main. Compile and run and verify that all three tests are passed.

## Step Five

Create a new class called Laboratory. Laboratory should have one member function, which returns void, called execute(). It takes no arguments and may throw a LaboratoryException. Be sure to declare the fact that it can throw this exception. execute should be coded so that it always throws a LaboratoryException, with some appropriate error message. After the throw statement it should print a "failed" message.

Now go back to your main file and create a function called test4. test4 should have a try/catch statement where the try body creates a Laboratory object (preferably as a local variable, not with the **new** operator), calls its execute() member function, and prints out a failure message.
The catch block should catch a LaboratoryException this time, and print out a "passed" message with the exception's error message.

Add test4 to the tests called by your main routine. Compile and run to verify that all four tests pass. Then call the TA to demonstrate your code and get your mark.

Updated Tue June 04 2019, 22:08 by shermer.