C++ Week 8 Exceptions

Exception Handling

Exceptions provide a mechanism to react to exceptional circumstances in our program, by transferring control to special functions called *handlers*.

To catch exceptions we must place a portion of code under exception inspection. This is done by enclosing that code in a *try block*. When an exceptional circumstance arises within that block, an exception is thrown that transfers the control to the exception handler. If no exception is thrown, the code continues normally and all handlers are ignored.

An exception is thrown by using the throw keyword.

Exercise 1

Create a new C++ Console project named Exception and add the following class.

```
#include <iostream>
using namespace std;

class Weapon {
private:
	int rounds;
	bool jammed;
public:
	Weapon() {
		rounds = 0;
	}
	void Fire() {
		if (rounds < 1)
			throw (1);
		cout << "Weapon Fired\n" << endl;
		rounds--;
	}
	void Reload() {
		rounds += 10;
		if (rounds > 10)
			rounds = 10;
		cout << "Weapon Reloaded \n" << endl;
	}
};
```

Within the main function create a static instance of the Weapon and invoke its Fire member function. When you run the application the console will display a Debug Error message. This is because an exception was generated by Fire (throw(1)) and not handled within the code. Catch the exception within main and output the text **"error:"** followed by the error number (1- the number thrown).

The C++ Standard library provides a base class, from which we can derive new exception classes that reflect the specialized types of exceptions that may be generated in your programs. The base class is called exception and is defined in the <exception> header file under the namespace std. This class has the usual default constructors and destructors, plus an additional virtual member function called **what,** that returns a null-terminated character sequence (char *), which can be overwritten in derived classes to contain a description of the exception being thrown.

Exercise 2

Add the NoAmmoException class defined below to your application.

```
#include <exception>
using namespace std;

class NoAmmoException : public exception {
public:
        virtual const char* what() const throw()
        {
                return "Insufficient ammunition";
        }
};
```

Define the private data member below within Weapon.

```
NoAmmoException ENoAmmo;
```

Now change the code within fire to throw ENoAmmo instead of the error number 1. Modify the main function to catch the new exception type and display the error message.

Exercise 3

Define a new exception class MagazineFullException that is thrown when the Reload member function is invoked when the magazine already contains 10 rounds. Add appropriate code to catch the exception but ensure the code will still catch the NoAmmoException. Invoke reload twice within the try block to ensure the exception is thrown.

C++ Week 8 Exceptions

Exercise 4

Set jammed to true within the constructor and throw an appropriate exception within Fire if the weapon is jammed. Catch the exception within main using a general catch all handler and test it.

Exercise 5

Modify the JammedException class so that it records the number of rounds the weapon has when the exception is throw. Create a new exception handler for this exception that displays the rounds.

Exercise 6

Update the Weapon's member function declarations to show which exceptions they throw.

Exercise 7

Modify the Weapons constructor to accept the initial number of rounds and through an assert statement within the constructor ensure this is between 0 to 10 inclusive.