**COSC 1437 Programming Fundamentals II**

L E S S O N S E T

**By KariAnn Harjo**

14 More about Classes

Exercise 1:

Write a full class definition for a class named Counter, and containing the following members:

* A data member counter of type int.
* A data member named limit of type int.
* A static int data member named nCounters which is initialized to 0.
* A constructor that takes two int arguments and assigns the first one to counter and the second one to limit. It also adds one to the static variable nCounters
* A member function called increment that accepts no parameters and returns no value. If the data member counter is less than limit, increment just adds one to the instance variable counter.
* A member function called decrement that accepts no parameters and returns no value. If counter is greater than zero, decrement subtracts one from the counter.
* A member function called getValue that accepts no parameters. It returns the value of the instance variable counter.
* A static function named getNCounters that accepts no parameters and return an int. getNCounters returns the value of the static variable nCounters.

class Counter {

int count; // Current count value

int limit; // Limit for the count

static int numCounters; // Number of Counter instances created

public:

Counter(int initialCount, int countLimit) {

count = initialCount;

limit = countLimit;

numCounters++;

}

void increment() {

if (count < limit) {

count++;

}

}

void decrement() {

if (count > 0) {

count--;

}

}

int getValue() {

return count;

}

static int getNumCounters() {

return numCounters;

}

};

int Counter::numCounters = 0;

Exercise 2:  
  
Define a class named Window with the following members:

* An int member variable named width
* An int member variable named height
* A constructor that accepts two int arguments. The first argument should be assigned to width and the second argument should be assigned to height.
* The Window class should declare the areSameSize function, described below, as a friend.

Outside of the Window class, write a function named areSameSize that accepts two Window objects as arguments, and returns a bool indicating whether the two Window objects are the same size. Two Window objects are the same size if their widths and heights match.

class Window {

int width; // Width of the window

int height; // Height of the window

public:

Window(int w, int h) {

width = w;

height = h;

}

friend bool areSameSize(const Window& win1, const Window& win2);

};

// Friend function to compare sizes of two Window objects

bool areSameSize(const Window& win1, const Window& win2) {

return (win1.width == win2.width) && (win1.height == win2.height);

}

Exercise 3:

Assume that a class named Window has been defined, and it has two int member variables named width and height.

Write a function that overloads the << operator for the Window class. The function should accept a reference to an ostream object and a constant reference to a Window object as arguments. The function should send a string in the following format to the ostream object:

"a (width x height) window"

Where width and height are replaced by the values of the Window object's width and height members. For example, if the Window object's width member is set to 80 and its height member is set to 20, the function would send the following string to the ostream object:

"a (80 x 20) window"

Don't forget to have the function return the proper value as well.

Assume that the function has already been declared as a friend in the Window class with the following statement:

friend ostream &operator << (ostream &, const Window &);

#include <iostream>

using namespace std;

class Window {

int width; // Width of the window

int height; // Height of the window

public:

Window(int w, int h) {

width = w;

height = h;

}

friend ostream& operator<<(ostream& os, const Window& win);

};

// Overloaded << operator to display Window object

ostream& operator<<(ostream& os, const Window& win) {

os << "a (" << win.width << " x " << win.height << ") window";

return os;

}

int main() {

Window myWindow(80, 20);

cout << myWindow << endl; // Output: "a (80 x 20) window"

return 0;

}