Summer Semester 2020, Final Project

Seneca Valet Parking Application

Milestone 3 – ReadWritable Abstract Bass Class
V1.0

Milestone 3:

To continue with the project, we need to have an abstract base class to make our future Car and Motorcycle classes read/writable. We call this class ReadWritable

The ReadWritable module:

Create a class called ReadWritable. ReadWritable class should force Read and Write capability to all its derived classes in two different ways; Comma Separated Values or Screen format.

The ReadWritable class implementation:

Properties: (member variables)

ReadWritable has only one member variable:
Comma Separated Values flag.
Add a bool flag as an attribute for the class to hold the object in Comma Separated mode or not.

Other properties:
Add other properties if, or when needed.

Constructor implementation:

- ReadWritable Has only a no-argument constructor that sets the Comma Separated Values flag to false.

Destructor implementation:

- Create an empty virtual destructor for ReadWritable Class.

Public Member function implementations:

- ```bool isCsv()const;```

  This query returns the Comma Separated Values flag;

- ```void setCsv(bool value);```
  This function set the Comma Separated Values flag to the incoming bool value.

- **Read and Write**
  Create two pure virtual functions to read and write this object in future derived class implementations.
  The read and write function signatures have been described in milestone 1.

Helper insertion and extraction operator overloads for istream and ostream:

- Implement the **operator<<** and **operator>>** to make any **ReadWritable** class Writable or Readable using **ostream** and **istream**.

Other member functions:

- Add other member functions to the **ReadWritable** class if needed.

Milestone 3 Duedate:

This milestone is due by Monday July 20; 23:59;
~profname.proflastname/submit 244/MS3/RW -due<ENTER>

Milestone 3 submission:

To test and demonstrate execution of your program use the data provided in the execution sample below.

If not on matrix already, upload **ReadWritable.cpp**, **ReadWriable.h and ms3_RWTester.cpp** programs to your matrix account.
Compile and run your code and make sure that everything works properly.

Then, run the following command from your account (use your professor's Seneca userid to replace profname.proflastname, and your section ID to replace NXX, i.e., NAA, NBB, etc.):

~profname.proflastname/submit 244/MS3/RW <ENTER>

and follow the instructions generated by the command.

Important: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

Milestone one tester program:

```
/* ----------------------------------------------------
Final Project Milestone 3
Module: ReadWritable
Filename: ms3_RWTester.cpp
Version 1.0
Author Fardad Soleimanloo
Revision History
-----------------------------------------------------------
Date      Reason
2020/7/4  Preliminary release
```

```
----------------------------------------------------------*/

#include <iostream>
#include <fstream>
using namespace std;
#include "ReadWritable.h"
using namespace sdds;
class Box :public ReadWritable {
   int m_width;
   int m_height;
public:
   Box(int width = 0, int height = 0) {
      m_width = width;
      m_height = height;
   }
   std::istream& read(std::istream& istr = std::cin) {
      if (isCsv()) {
         istr >> m_width;
         istr.ignore();
         istr >> m_height;
      }
      else {
         cout << "Width: ";
         istr >> m_width;
         cout << "Height: ";
         istr >> m_height;
      }
      return istr;
   }
   std::ostream& write(std::ostream& ostr = std::cout)const {
      if (isCsv()) {
         ostr << m_width << "," << m_height;
      }
      else {
         int i;
         int j;
         for (cout << "*", i = 0; i < m_width - 1; cout << " *", i++);
         ostr << endl;
         for (j = 0; j < m_height - 2; j++) {
            for (cout << "* ",i = 0; i < m_width - 2; i++, cout << "  ");
            cout << "*" << endl;
         }
         for (cout << "*", i = 0; i < m_width - 1; cout << " *", i++);
      }
      return ostr;
   }
};

void pause();
void test1();
void test2();
void test3();
void test4();
int main() {
   test1();
   pause();
   test2();
   pause();
```

```
   test3();
   pause();
   test4();
   return 0;
}

void pause() {
   cout << "Press Enter to continue.";
   cin.ignore(1000, '\n');
}
void test1() {
   int i;
   Box B[2] = { {3,3},{30,3} };
   cout << "Testing operator<< overload (pass 1): " << endl;
   cout << "These two outputs should match" << endl;
   cout << "Output 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1" << endl;
   for (i = 0; i < 2; i++) {
      cout << B[i] << endl;
      B[i].setCsv(true);
      cout << B[i] << endl;
   }
   cout << "Output 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2" << endl;
   cout
      << "* * *" << endl
      << "*   *" << endl
      << "* * *" << endl
      << "3,3" << endl
      << "* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *" << endl
      << "*                                                           *" << endl
      << "* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *" << endl
      << "30,3" << endl;
   cout << "************************************************************" << endl;
}
void test2() {
   int i;
   ofstream bxout("boxes.csv");
   Box B[2] = { {3,3},{30,3} };
   for (i = 0; i < 2; i++) {
      B[i].setCsv(true);
      bxout << B[i] << endl;
   }
   bxout.close();
   ifstream bxin("boxes.csv");
   cout << "Testing operator<< overload (pass 2): " << endl;
   cout << "These two outputs should match" << endl;
   cout << "Output 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1" << endl;
   while (bxin) {
      cout << char(bxin.get());
   }
   cout << "\rOutput 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2" << endl;
   cout << "3,3\n30,3\n";
   cout << "************************************************************" << endl;
}
void test3() {
   Box B;
   cout << "Testing operator>> overload (pass 1): " << endl;
   cout << "Enter the following: " << endl;
   cout << "4 <ENTER>" << endl;
```

```
   cout << "5 <ENTER>" << endl;
   cin >> B;
   cout << B << endl;
}
void test4() {
   int i;
   ifstream bxin("boxes.csv");
   Box B[2];
   for (i = 0; i < 2; i++) {
      B[i].setCsv(true);
      bxin >> B[i];
   }
   cout << "Testing operator>> overload (pass 2): " << endl;
   cout << "These two outputs should match" << endl;
   cout << "Output 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1" << endl;
   for (i = 0; i < 2; i++) {
      cout << B[i] << endl;
   }
   cout << "\rOutput 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2" << endl;
   cout << "3,3\n30,3\n";
   cout << "************************************************************" << endl;
}

/*
Testing operator<< overload (pass 1):
These two outputs should match
Output 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
* * *
*   *
* * *
3,3
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
30,3
Output 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
* * *
*   *
* * *
3,3
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
30,3
************************************************************
Press Enter to continue.
Testing operator<< overload (pass 2):
These two outputs should match
Output 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3,3
30,3
Output 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3,3
30,3
************************************************************
Press Enter to continue.
Testing operator>> overload (pass 1):
Enter the following:
```

```
4 <ENTER>
5 <ENTER>
Width: 4
Height: 5
* * * *
*     *
*     *
*     *
* * * *
Press Enter to continue.Testing operator>> overload (pass 2):
These two outputs should match
Output 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3,3
30,3
Output 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3,3
30,3
************************************************************
*/
```