C SC 397a, Spring 2010
Assignment 8
Due: Monday, April 19 at 22:00:00

Problem 1. (75 points) Books

In this problem you will address the fact that bookcases often hold items other than books. You are to generalize the Bookcase class from assignment 6 so that it also can accommodate fish bowls, fish tanks, and plants, along with books.

Instead of holding pointers to instances of Book, a Bookcase will hold pointers to instances of classes derived from Item, an abstract class. Three classes are directly derived from Item: Book, Plant, and Aquarium. Book and Plant are concrete classes. Aquarium is abstract; two concrete classes, FishBowl and FishTank, are derived from it.

Here is the interface for Item:

```
class Item {
   public:
      //
      // Return the width of the Item in its thinnest acceptable orientation
      //
      virtual int minWidth() const = 0;

      //
      // Return the maximum weight of the Item.
      //
      virtual int maxWeight() const = 0;

      //
      // Return a description of the Item
      //
      virtual String descrip() const = 0;

      //
      // Print the description of the Item on standard output, followed by a newline.
      //
      virtual void print() const;

      //
      // Enable (or disable) debug descriptions.  When enabled, the String produced
      // by descrip() includes the minimum width and maximum weight of the Item.
      // Initially, debugging is disabled.
      static enableDebug(bool);

      //
      // Item needs a virtual destructor.
      virtual ~Item() {}

   };
```

Item may have no other public member functions, but private and protected member functions may be added. Private and protected data members are acceptable, of course.

Here is the interface for Bookcase:

```
class Bookcase {
   public:
      Bookcase(int id);
      int id() const;
      bool addShelf(int width, int capacity);
      bool add(Item *) const;
      void print() const;
      };
```

The interface differs from the assignment 6 interface in one way: add(Item *) has replaced add(Book *). Methods should have the same behavior as in assignment 6, but interpret those specifications generically—in terms of items, not books.

The Book constructor has this signature:

```
Book(const String& title, int thickness, int weight );
```

As in assignment 6, the thickness is in millimeters and the weight is in dekagrams. Book::descrip() returns the title, enclosed in single quotes. Here is an example of construction and description:

```
Book b("C by Dissection", 35,  50);
cout << b.descrip() << endl;  // Output: 'C by Dissection'
```

Note that Book is not required to have the name(), thickness(), and weight() methods specified in assignment 6 but you may retain those methods if you desire.

The FishTank constructor has this signature:

```
FishTank(int base1, int base2, int height);
```

A FishTank is a rectangular solid (i.e., a box) with the specified dimensions in millimeters. The length of one side of the base is base1 millimeters; the length of the other side is base2 millimeters. A fish tank is always placed on a shelf so that its widest side is visible. That is, its width is the larger of base1 and base2.

The weight of a fish tank is considered to be the weight of the water it contains if it is completely full. Assume that a cubic millimeter of water weighs .0001 dekagrams. Note: Calculate the weight using mixed arithmetic and then cast it to an int (discarding any fractional part) like this:

```
int weight = (int)(base1 * base2 * height * .0001);
```

Example of construction and description:

```
FishTank ft1(200, 400, 300), ft2(400, 200, 300);
cout << ft1.descrip() << endl;  // Output: 400mm wide fish tank
cout << ft2.descrip() << endl;  // Output: 400mm wide fish tank
```

The FishBowl constructor has this signature:

```
FishBowl(int radius);
```

A fish bowl is considered to be a sphere with the given radius. Like the fish tank, the maximum weight of a fish bowl is the weight of the water it contains if completely full. The formula for the volume of a sphere with radius r is $4/3\pi r^3$. The width of a fish bowl is twice the radius.

Example of construction and description:

```
FishBowl fb(15);
cout << fb.descrip() << endl;  // Output: 30mm high fish bowl
```

It is required that FishTank and FishBowl be derived from Aquarium, which is turn is derived from Item. Aquarium must be an abstract class.

Example, assuming ft1 and fb from above:

```
Aquarium *a[ ] = { &ft1, &fb };
a[0]->print();
a[1]->print();
```

Output:

```
400mm wide fish tank
30mm high fish bowl
```

The Plant constructor has this signature:

```
Plant(const String& name, int spread, int weight);
```

Each plant has a name. The spread is the width of the plant's foliage, in millimeters. The weight is specified in dekagrams. Here is an example of construction and description:

```
Plant p("Coltrane", 300, 200);
cout << p.descrip() << endl;  // Output: A plant named Coltrane
```

Classes derived from Item may have additional member functions, if you desire. For example, Aquarium might have a method that performs a computation that's common to both FishTank and FishBowl.

Be sure that Item and Aquarium are abstract. For example, definitions like Item i; and Aquarium a; should produce compilation errors indicating (in some way) that the class is abstract.

You may use your assignment 6 solution, or my solution, in $FILES/a8/Books.{h,cc}, as a starting point. The String class is our String class. The latest version is in $FILES/a8/String.{h,cc}. A new static member function is String String::toString(int). It returns an ASCII representation of its integer argument. It uses the ostringstream class, a subclass of ostream that treats insertions as concatenations onto a string. We'll see it in lecture soon. Also, overloaded == and != operators have been added.

For π, include the <cmath> header and use M_PI.

All source code must be contained in two files: Books.h and Books.cc. You may distribute code between

the two files as you see fit.

Here is an example:

```
// - - - b6.cc - - -
#include <iostream>
#include "Books.h"

using namespace std;

int main()
{
   Book bk("C by Dissection", 35, 50);
   FishTank ft1(100,200,80), ft2(50,50,200);
   FishBowl fb(30);
   Plant p("Coltrane", 300, 200);

   Bookcase b(10);
   b.addShelf(400,100);
   b.addShelf(400,200);
   b.addShelf(400,400);

   Item *stuff[] = {&bk, &ft1, &ft2, &fb, &p, 0};

   for (int i = 0; stuff[i]; i++)
      b.add(stuff[i]);
   b.print();

   Item::enableDebug(true);
   cout << endl << "Debug enabled..." << endl;
   b.print();

}
```

Execution:

```
$ g++ -Wall -o b6 b6.cc Books.cc String.cc
$ ./b6
Bookcase #10
--- Shelf (400 mm, 100 dg) ---
 1: 'C by Dissection'
 2: 60mm high fish bowl
--- Shelf (400 mm, 200 dg) ---
 1: 200mm wide fish tank
--- Shelf (400 mm, 400 dg) ---
 1: 50mm wide fish tank
 2: A plant named Coltrane

Debug enabled...
Bookcase #10
--- Shelf (400 mm, 100 dg) ---
 1: 'C by Dissection' (35mm, 50dg)
```

```
 2: 60mm high fish bowl (60mm, 11dg)
--- Shelf (400 mm, 200 dg) ---
 1: 200mm wide fish tank (200mm, 160dg)
--- Shelf (400 mm, 400 dg) ---
 1: 50mm wide fish tank (50mm, 50dg)
 2: A plant named Coltrane (300mm, 200dg)
```

Note that the second b.print(), preceded by Item::enableDebug(true), causes the minimum width and maximum weight to be appended to the description.

For grading, an additional class derived from Item will be introduced. If the code in your Bookcase class and any supporting classes, like Shelf, is written in terms of Items and Item methods, your code should have no trouble handling the new class. As a practical check, if the text "Aquarium", "Fish" or "Plant" appears in any of your code for Bookcase, you're headed for trouble.

Here's one approach to this problem: (1) Overhaul the code in your Bookcase and Shelf classes so that it is written in terms of Item, not Book. That's a little more than globally changing "Book" to "Item", but not a lot more. (See me if you don't see that.) (2) Derive Book from Item. (3) Do Plant. (4) Do Aquarium, FishTank, and FishBowl. Get the code to compile for each step before moving on to the next step.

Problem 2. (10 points) IntList

In this problem you are to write an iostream inserter for the IntList class from assignment 7. You may build on your IntList code or use my code, in $FILES/a8/IntList.{h,cc}.

Here are some examples of the required output format:

```
IntList L;

cout << "L = " << L << endl;   // Output: L = [ ]

L.add(10);
cout << "L = " << L << endl;   // Output: L = [10]

L.add(2);
L.add(-4);
cout << "L = " << L << endl;   // Output: L = [10, 2, -4]
```

It can be seen that the list contents are enclosed in square brackets. If a list has more than one value, a comma and a blank appear between values. No other whitespace appears.

Along with including the <iostream> header in IntList.h, BE SURE to specify 'using namespace std;' If you don't, you'll get some syntax errors that aren't very helpful. (Or, use std:: as needed.)

You can (1) define the inserter as an inline in IntList.h and leave IntList.cc as-is or (2) declare the inserter in IntList.h and define it in IntList.cc, but turn in both IntList.h and IntList.cc regardless of your approach.

Problem 3. (10 points) sleeper.txt *[Puzzle Problem]*

Consider the following inserter for the Range class. It has passed the developer's tests but a critical error lies undetected. What is the error and what would its manifestation be? Submit your answer as a text file, sleeper.txt.

```
ostream& operator<<(ostream& o, const Range& r)
{
   int low, high;
   r.get_bounds(low, high);
   cout << '[' << low << ".." << high << ']';
   return o;
}
```

Problem 4. (10 points) overhead.cc

Write a simple program that demonstrates that the presence of even a single virtual member function in a class causes instances of the class to occupy more memory. Submit it as overhead.cc.

Problem 5. (5 points) error.txt *[Puzzle Problem]*

The following source file ($FILES/a8/error.cc) generates compilation errors on the last two statements.

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

int main()
{
   cout << "2 + 4 = " << 2 + 4 << endl;
   cout << "2 * 4 = " << 2 * 4 << endl;
   cout << "2 | 4 = " << 2 | 4 << endl;
   cout << "2 & 4 = " << 2 & 4 << endl;
}
```

What is the cause of the errors? What should the developer do to remedy the problem? Submit your answers as a text file, error.txt.

Problem 6. (Extra Credit) extra.txt

Submit a plain text file named extra.txt with the following.

(a) (1 point extra credit) Estimate how long it took you to complete this assignment. Other comments about the assignment are welcome, too. I appreciate all feedback, favorable or not.

(b) (1-3 points extra credit) Cite an interesting course-related observation (or observations) that you made while working on the assignment. The observation should have at least a little bit of depth. Think of me saying "Good!" as one point, "Interesting!" as two points, and "Wow!" as three points. I'm looking for quality, not quantity.

Miscellaneous

You must make good use of C++, including but not limited to:

- Using bool, const, and member initializers when appropriate
- Using new and delete rather than malloc and free
- No public data members
- No memory leaks

On this assignment and those to follow, printf, sprintf, et al. are forbidden—you'll need to use the iostream library from now on for ALL input and output.

Like assignment 6, Books will be tested with a combination of b?.cc files and bx.? files that are input for bx.cc, in $FILES/a8. There is one test program for IntList: $FILES/a8/il0.cc.

Each of the test programs and bx.? files has a comment designating the point value. Each is all or nothing for the specified point value; if there's a diff, even one byte, it'll be zero points for that one. (Use diff to be sure that the output of your solutions exactly matches the output of the reference versions!) The set of test programs, reference versions, and any associated data files will freeze exactly one week (168 hours) prior to the due date/time.

$FILES/a8/{testbooks,testintlist} are simple test scripts.

Feel free to use comments to document your source code as you see fit, but note that no comments are required.

You may distribute code between .h and .cc files as you see fit.

You should be able to complete this assignment using the material on slides 1-244.

Don't hesitate to ask me for hints and/or help if you have trouble with a problem.

Deliverables

Use `turnin` with the tag 397a_8 to submit your solutions for grading. The deliverables for this assignment are Books.h, Books.cc, IntList.h, IntList.cc, sleeper.txt, overhead.cc, error.txt, and, if you choose to submit it, extra.txt.