ECE 2574: Data Structures and Algorithms - Array-based implementations

C. L. Wyatt

Today we will see how to formalize the notion of a public interface using classes, continue our fixed-size array implementation of the Bag ADT, and introduce the test-code-test cycle.

- Warmup
- Using an interface defined as an Abstract class
- Implementing the LimitedSizeBag
- Testing

Interfaces

We have seen that the first step to designing a class is to define it's interface.
The interface for a class are the public type definitions, constants, and member functions. Any associated functions exported via the header are included in the interface.

Review of the C++ class

A type is a concrete representation of a concept.
For example, the type float approximates a real number
A class is a user-defined type that extends the built-in types.
For example, the class Bag models the concept of the ADT Bag.

Review of the C++ class

Be sure you understand the difference between a type and an instance of the type.
An instance of a type is also called a variable or an object.
See code example.

Classes provide many advantages for implementing ADT's.

- They can hide implementation details via private.
- They provide a means of forcing the ADT interface to be used.
- They enable type-checking on complex concepts.
- They assist with assertion checking and maintaining constraints.

All of this helps to keep the type in a **well-defined state**.

The first decision is whether to make a function a member

A rule of thumb is only make functions members if they:

- Have to be (constructors, etc)
- Must access internal data for efficiency reasons

Example: size() versus isEmpty
isEmpty can just test if size() == 0 and so can be an non-member function.
Note the C++ standard library seemingly violates this rule of thumb regularly, but there is always a performance reason.

Lets review the Bag interface

See `abstract_bag.hpp` in the starter code for today.

Warmup #1

How should you determine what methods of a class are public and which are private? (check all that apply)

- all methods should be private 9%
- all methods should be public 7%
- methods that correspond to the API should be private 16%
- methods that correspond to the API should be public 81% CORRECT

Warmup #2

What does it mean to make a method (member function) constant? (check all that apply)

- all arguments are constant 21%
- the method can be called on a instance that is constant 21%
- the returned value is constant 26%
- the method can only call other methods in the same class that are constant 38% CORRECT
- the method cannot modify any member data 93% CORRECT

Is there ever an advantage to using public variables?

One case where it makes sense is if you can make the variable constant at object construction. This saves writing an accessor/getter.

Warmup #3

When should you write code that tests the code you are working on?

- Never 0%
- Before writing any other code 22%
- While writing the code 67% CORRECT
- After you are done 10%

Testing

Testing is like exercise and eating healthy. You know its good for you but most people don't like it.
But testing prevents *technical debt*, keeping your code healthy, and can actually improve your productivity (like exercise and a good diet).
The mantra is: "Test Early, Test Often, Test Automatically"

Testing should be integrated into your workflow.

Almost everyone does this at some level (or should). Never just dump a bunch of code into a file without testing.

"code a little, test a little"

But rather than manually testing each compile cycle, take the time to write tests.

What to test: Unit tests

Unit tests exercise each module, generally a class and associated functions.

Treat the public interface as a contract. Your test code checks the contract.

Since this course is about ADT's unit tests are our focus. There are other kinds of testing though (see ECE 3574).

Exercise: LimitedSizeBag

- Define a Limited Size Bag that conforms to the AbstractBag interface
- Write some tests for the LimitedSizeBag methods
- Implementing the LimitedSizeBag methods

Next Actions and Reminders

- Read CH 117-132 on memory allocation (this should be a review)
- Take warmup before noon on Fri 9/15.
- Project 1 has been released, due 9/27 by 11:55 pm (Wyatt section)