Interlude 1: C++ Classes

Vocabulary

Write out a description of each vocabulary term for reference later.

Term	Description
Abstract class	
Accessor (“Getter”) function	
Base class (Derived class, Child class)	
Constructor function, Default constructor, Parameterized constructor	
Destructor function	
Header file	
Inheritance	
Initializer	
Mutator (“Getter”) function	
Namespace	
Overriding methods	
Pass-by-reference, pass-by-const-reference	
Pass-by-value	
Preprocessor directive	
Private members	
Pure virtual method	
Source file	
Super class (Parent class)	
Templates	
Virtual method	

Concepts

File separation, Page 32

What kind of code belongs in header (.hpp or .h) files?

In general, libraries that are written for C++ but NOT C the .hpp extension. You can use C libraries in C++, so often these use the .h extension.

What kind of code belongs in source (.cpp) files?

Member accessibility, page 33

What is the design purpose of making member variables of a class private?

What does public, protected, and private accessibility mean?

Constructors and Destructors, Page 34

When is a Constructor method run?

When is a Destructor method run?

What is a default constructor?

What is a parameterized constructor?

Accessors and Mutators, Page 34

What is the purpose of an Accessor (aka “Getter”) function?

- What kind of parameter(s) does an Accessor function take, if any?
- What kind of return type does an Accessor function have?

What is the purpose of a Mutator (aka “Setter”) function?

- What kind of parameter(s) does an Mutator function take, if any?
- What kind of return type does an Mutator function have?

Parameters, Page 35

By default, non-array variables are passed to functions in what manner? (Pass-by-…)

By default, arrays are passed to functions in what manner? (Pass-by-…)

What is the purpose of pass-by-reference?

What is the design purpose of pass-by-const-reference?

Preproccessor, Page 36

The #ifndef … #define … #endif preprocessor directives should go at the beginning and end of what kind of C++ files? (.cpp? .hpp/.h?)

Why should #pragma once not be used?

What is the purpose of the #ifndef … #define … #endif preprocessor directives?

Namespaces, Page 37

Namespaces are an optional feature of C++ but they can be useful when creating a library of code. By using namespaces, you can avoid naming conflicts and ambiguity.

Example: let's say that you're writing a program that uses a UI library and a Game library; both libraries have Draw() functions, but you need to be able to explicitly call a UI object's Draw vs. a Game object's Draw. If they're implemented within namespaces, you could do something like `UILib::Draw(…)` and `GameLib::Draw(…)`.

How do you use namespaces within code?

Initializers, Page 37

What type of member method are initializers used in?

What is the point of using initializers?

Templates, Page 38

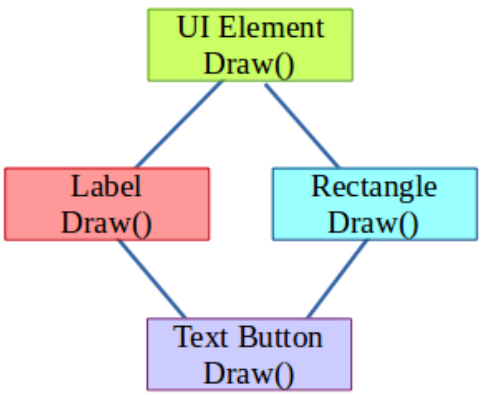
What is the point of templates?

Give some example code of a templated function.

Give some example code of a templated class/struct.

Class relationships, Page 40

You can use inheritance to create more specialized versions of a class. By inheriting from a parent class (aka base class), the child (aka subclass) inherits the public and protected members of its parent. This can be useful when wanting to re-use specific types of



functionality. In C++, a class can inherit from multiple parents, but this is not true of all OOP languages.

Example: Give an example of a family of classes and their relationships, using items like: ANIMAL, FELINE, CAT, or VEHICLE, BOAT, CAR. Try to come up with some behaviors (functions) and attributes (variables) that might be shared between them, and that might be unique to each subclass.

Overriding Methods

Sometimes while inheriting from a parent class, you will want to override an existing function; for example, a button's Draw() function might work differently than a text label's Draw() function. You can override functions while still being able to access ancestors' versions of those functions.

Polymorphism, Page 44

In C++, you can utilize polymorphism in a family of classes. In general, the idea is that you treat some variable or variables as the base (parent) class, and the program will figure out which functions to call (the parents' version or the actual subclass' version) at run-time. To implement this, you must use virtual functions.

How do you make a member function virtual?

How do you make a member function *pure virtual*?

What is an abstract class?