Objects, Classes, and UML

1-18-2004

Opening Discussion

- I hope everyone had a nice weekend. What did we talk about last class?
- Have you thought about possible project ideas? Keep in mind that the design for the assignment is due next Tuesday. You should read the project description on the web page under the links.
- Did anyone read any of “From C to Java”? I didn't get any typos and since I haven't proofed it even once I'm guessing that means it wasn't read much. Perhaps that will change after today.

What is Object-Orientation?

- This is actually a very difficult question to answer because it is hard to get people to agree. The one term that almost everyone will agree is part of object-orientation is encapsulation.
- Encapsulation is the binding together of data and functionality into entities called objects. This often also includes being able to hide some data and functionality.

Objects

- An object is basically a set of data (attributes) that has certain functions associated with it. The functions (called methods, behaviors, or operations) can act on the data for that object.
- In many ways, an object in an object oriented program is much like an object in the real world. It has certain things that it can do (methods) and data describing a state.

Objects Continued

- Keep in mind that an object represents a single entity and gives the data for that entity.
- For example, computer is not an object. Your computer is an object. This is a significant distinction to make. The object itself represents a single entity, not a class of entities.
- Calling a method of an object is often referred to as sending it a message.

Classes

- As the name implies, a class represents a class of objects. In some ways, a class is a blueprint for objects of a given type. Just as a blueprint for a car is not a car, a class is not an object. A class defines a type.
- What you will write in your code are classes. (Note that not all object oriented languages are class based.) You get to specify in some general way what types of object you have in your program.

Classes Continued

- In Java objects are instances of a class.
- When a method is called on an object it has access to all the attributes of that object.
- Java is not 100% object-oriented because it has primitive types that aren't objects and aren't instances of any class. This was done for efficiency.
- When talking about classes we often talk about their interface or "public interface". This is the set of methods and attributes that are used by other objects.

Classes vs. Structs

- Classes are similar to structs. They add a lot more though. A struct was a blueprint for a collection of data and you could make instances of it by declaring variables of that type or using malloc to get the memory.
- A class can have data just like a struct, but it can also have the member functions that manipulate that data.

Inheritance: Short Version

- Class based OOPLs typically also allow classes to "inherit" from one another.
- Inheritance implies two things. The name comes from the fact that the inheriting class (subclass) gets operations and attributes from the inherited class (superclass). It also implies a subtyping relationship. Example: Honda inherits from Car and Accord inherits from Honda.

Polymorphism: Short Version

- The term polymorphism technically means "many shapes". In programming, it implies that something works with many types. The subtyping aspect of inheritance plays a role here.
- A function that works on Cars should work with any instance of any subtype of Car as well. For example, you could give it my instance of Accord and it should work.

Basics of Classes in Java

- All functions in Java are methods of some class. There are no stand alone functions.
- Classes, attributes, and methods each have a visibility attached to them.
  - public - can be used by anything
  - package private - can be seen by all classes in this package (this is the default)
  - protected - can be used by subclasses
  - private - can be used only by methods of that

this and Using Members

- When you are writing a method of a class, it has direct access to the member data and methods of that class. You don’t have to use the ‘.’ notation.
- To be explicit, you can use the ‘this’ keyword which implies the object that the method was invoked on.

static

- The term static in the C-family languages implies something like “there is only one”. This is true in Java as well.
- A static member or method is associated with the class itself, not with an object/instance of that class.
- They can be reached or invoked without having an object of that class too.

main in Java

- Like C/C++, Java programs always begin in a special method named main. However, in Java main is a static method of a class (remember there are no stand alone functions). Every class can have its own main which can be very helpful for debugging.
- The signature of main is
  - public static void main(String[] args) { }

UML Class Diagrams

- UML stands for Unified Modeling Language. It is a formal graphic representation of software analysis and design. There are many types of UML diagrams, but we will mainly be looking at class diagrams.
- In this diagram classes are represented by boxes.
- Java also has things called interfaces that we will look at more a little later.

Inside a Class

- The box for each class is divided into three regions. The top one contains the class name and possibly some modifiers.
- The second region has attributes of the class. Typically these are specified with a visibility modifier followed by name:type.
- The third region holds operations (methods). They are displayed in a similar format but arguments can follow the name.

Modifier Symbols

- Any of the attributes of operations can be modified with a symbol showing visibility.
  - + for public
  - # for protected
  - - for private
- Attributes can also be preceded by a '/' in some design tools to show that a given attribute is read only. Note that attributes aren't always member data.

Class Relationships

- In the diagram a subclass points to the class that it inherits from with an arrow with a closed head (more on what this means in two classes).
- An open header arrow in UML also gives us a way to denote when two classes are associated with one another is some way. Typically this implies that one class has attributes whose type is the other class.
- Associations can be labeled with a name telling what type of association it is.
- Example: Screen has grid of Blocks

Documentation Comments

- In Java, comments that start with /** are documentation comments. These comments are used by javadoc to produce HTML documentation.
- These comments should go above all classes and methods. Inside the comment you start with a summary sentence then have a paragraph describing the class or method. After that can come certain “tags” that begin with @.

Minute Essay

- Do you have any questions about assignment #1 at this point? Remember, your design is due to me next Tuesday.
- Over the next 3 classes I will be running through Java basics as well as more details on inheritance and OOP. Were there any aspects of today's class that you felt were unclear and would like to hear again in more detail?