Writing Classes

Kate Gregory
www.gregcons.com/kateblog

pluralsight
hardcore developer training

A Class of Your Own

- **A class pulls together related data**
  - A customer's first name, last name, address, and phone number
  - An account's number, balance, and list of transactions
- **It also adds functions that (generally) operate on the data**
  - Get a customer's full name
  - Post a transaction to an account – including updating the balance
  - Includes operator overloads if those make sense
- **Keeping these together makes the code easier to change and use**

Simple Class System - Design

- **Account**
  - Keeps track of a balance
  - Holds a vector of Transaction objects
  - Deposit and Withdraw member functions
  - Report function – collection of strings that calling code can print
- **Transaction**
  - Should have a date, but we'll ignore that for simplicity
  - Holds an amount, and a transaction type (string for now)
  - Report function – string describing amount and type

- **Deposit will:**
  - create a Transaction
  - Add it to the vector
  - Update the balance
- **Withdraw is the same**
  - Except you can't take out more than you have

Translating Design Into Code

- **Generally, member variables are private**
  - *Encapsulation*
- **Functions you think of early are usually public**
  - Services the class offers
- **Some classes need constructors**
  - Initialize variables – it won't be done for you
    - Use special initializing syntax to initialize member variables
  - Name of the constructor function is name of the class
  - Constructors have no return type
    - Not the same as returning void
  - Constructor takes parameters if it doesn't make sense to have an instance with default values

Structuring the Code

- **Can define it all in one file, but more typically:**
  - One header file per class just explains what is in the class
  - One .cpp file per class implements all the functions
- **Any code that uses the class includes the header**
  - So does the .cpp file that implements the class

- **Keywords to know**
  - `class { …many lines … };`
  - `private:`
  - `public:`
  - **Scope resolution operator** `::`

Inline Functions

- **Some functions are really obvious**
  - Added mostly to keep the data private
  - Makes sense to show the code right with the declaration of the function
  - Often called “inline”
- **Technically inline is a slightly different thing**
  - Compiler chooses
  - Speeds up your application
  - Vast majority of functions written in the declaration are inlined, and others might be too

Encapsulation

- **A well written class is changeable**
- **Make all your member variables private**
  - Code outside the class can't count on their names or types
  - You can change name, type without breaking code outside the class
  - Code outside the class doesn't need to know the rules or remember them
    - You can change the business rules later
- **You can add public member functions as gatekeepers**
  - Eg `GetBalance()` to find out an account's balance
    - Never assume one `GetSomething()` for every member variable
  - Don't always need a `SetSomething()`
- **Add as few public member functions as you can**
  - Use private functions if you just want to keep from repeating code
- **The more that is encapsulated, the better**
  - Changes in one part of the code don't affect other places
  - Easier for the developer and less likely to cause bugs elsewhere

Creating Instances

- **A constructor that takes no arguments is called a *default constructor***
- **Declare objects with default constructors the same as built in types:**
  - `Account acct;`
- **Declare objects with parameter-taking constructors using ()**
  - `Transaction t(amount, type);`

- **Don't use = when declaring an object and initializing it**
  - There are exceptions, but this is a good general rule

- **This code doesn't do what you think it does:**
  - `Account acct();`
  - It actually declares a function!

There's more … later

- **Some classes work with real-world things**
  - Open a file … and need to close it
  - Open a database connection … and need to close it
- **When an object is out of scope, you can't ask it to close or clean up the things it held**
- **C++ has great mechanisms to manage this**
  - Class can have a destructor that is called automatically
  - RAII makes life simple
  - You need to deal with copying
    - Two classes have copy of handle and one thinks it's done and closes the file?
- **Simple classes that just have local variables in them don't need to worry about lifetime management**

Summary

- **Writing a class starts with design**
- **A well designed class can be used like a built in type**
- **A well designed class hides its implementation details**
  - Leaves the developer free to change them without breaking other code
  - Saves those who use the class from having to remember to do things
- **Using one .h and one .cpp file per class is a good practice**