Today

- Course Overview
- C++ Classes and Inheritance

Assignments

- HW1 (out, due Tues)
Course Overview

Course communication

- Website: http://www.cs.gonzaga.edu/bowers/courses/cpsc223
- Piazza (you’ll receive an invite)
- GitHub classroom (more soon)

Please be sure to **carefully** read the syllabus

- **Office hours**: TR 11–12, W 1–2, by appointment
- **Grading**:
  - 45% programming homework
  - 5% reading
  - 10% quizzes
  - 20% Two mid-semester exams
  - 20% Final exam
- **Textbook**: ZyBooks online book (see syllabus)
  - Readings in HW and via “**Week #n:** -- ”
Exam, Quiz, and Homework basics

Exams and Quizzes are closed book, closed note, your own work
- you are responsible for understanding how to do the assignments
- you will be tested on material from programming assignments
- you must turn in homework individually

(Some) collaboration is encouraged ...
- discuss and talk about assignments
- work together (as long as you are doing your own work)
- plagiarism not allowed (e.g., do not split up work, share code)

Most CS jobs require you to demonstrate your ability to program
- and especially much of what we discuss in 223!
Warnings!

- Longer, more complex programming than you may have seen so far
- Class moves fairly quickly
- Requires time commitment (start assignments early!)

Class Sessions

- Mix of lecture, exercises, and short quizzes
- No electronic devices allowed during lecture and quizzes
  - Please see me for accommodations
  - Okay to use during exercises as needed
- Quizzes mostly at beginning of lectures (no makeup quizzes)
- Highly encourage questions during lecture
- Use piazza for questions outside of class
- Lecture notes posted after each class

Homework

- All assignments must have a cover sheet
- Homework assignments have a programming and an online reading/QA part
- Assignments more like intense “exercises” (versus “real” programs)
Course Topics

C++ Programming ... (first couple of weeks)

- Classes/OOP (e.g., operator overloading)
- Inheritance
- Practice with lists (vectors)
- More practice with pointers and recursion

Please Note: This is not a C++ syntax class!

- Mainly apply C++ you’ve learned to implement new data structures
- We do cover some C++ “loose ends”

Data Structures ... (majority of class)

- Review of linked lists
- Hash tables
- Binary trees
- Heaps
- More if time

Algorithms and (Complexity) Analysis ...

- Searching for data
- Sorting data (basic sorting algorithms)
- How to compare algorithmic performance
Quizzes and Project

At least one quiz a week
- usually at beginning of class
- usually take about 10 minutes
- no make-up quizzes!

Why are there so many quizzes?
- Give you feel for kind of questions I ask on exams
- Give you feedback on your understanding of material
- Give me feedback on your understanding
- Short quizzes shown to help improve final course grades / retention

Small group project during last third of semester
- more details later
- will involve a presentation

⇒ Quiz 0
Basic Terminology (Review)

⇒ Exercise Questions 1–3

Modularity

- Organizing functionality into separate, composable components ("modules")
- Helps make software easier to implement (decomposition) and maintain

Information Hiding

- Idea of restricting access to implementation details (hide information)
- Allows implementations to change without impacting other components

Encapsulation

- Use procedures (interface) to control access to implementation details

Design Goal: Can change implementation without changing non-module code
Classes are one way to create modules

- Classes should include both behavior (functions) and state (data)
- ... just behavior implies functions, just state implies simple variables or structs

A client of a class is code or another component that uses the class

- Creates a dependency ... the client depends on the class

Clients interact with classes via class objects

- a class is instantiated to obtain an object
- each object has its own state
Anatomy of a Class (Review)

⇒ Exercise Question 4

A simple BankAccount class definition (interface):

```cpp
class BankAccount
{
public:
    BankAccount(double min_bal);
    void deposit(double amount);
    double get_minimum_balance() const;
    double get_available_balance() const;
private:
    double min_balance;
    double cur_balance;
};
```

A class has member functions (“behavior”) and member variables (“state”)

- members can be more or less hidden from clients

Public access

- member is fully visible to clients (e.g., deposit)
- member is fully visible to other members

Private access

- member is hidden from clients (e.g., cur_balance)
- but is fully visible to the member functions of the class

We’ll look at 2 other access modifiers later: protected and friends