CS 2334: Programming Structures and Abstractions

Andrew H. Fagg
Symbiotic Computing Laboratory
School of Computer Science
University of Oklahoma

Teaching Assistants:
Joshua Southerland and Abed Karim

This software stuff is hard … Why?

This software stuff is hard … Why?

Complexity due to:

- Many different types of data
- Many different cases
- Code base gets large
- Multiple programmers

This software stuff is hard … Why?

Complexity:

Coordinate many activities at once

UMass Torso

Why Should We Care?

Does it matter that we get it right?

Why Should We Care?

Does it matter that we get it right?

- Correct and efficient implementation is important to our customers & employers
- Resources are often precious: e.g., data, people, and cpu
- Lives can be at stake

  (literally)

This software stuff is hard …

How do we get a handle on it?

Abstraction

- Abstraction: the process of simplifying the representation or description of some entity
  - Keep the key pieces
  - Throw away the extraneous details
- In software development: we use abstraction to temporarily hide details so that we can “get our mind” around the “big picture”

Abstraction

Not just one level of abstraction possible: we can imagine multiple levels of abstraction, depending on what we are working on and what we need to communicate

Course Coverage

- Abstraction!
- Software development
  - Design
  - Implementation
  - Testing
  - Debugging
- Ethics in computer science

Design

Design: the process of assessing the requirements of a software system and planning a solution

- What are the inputs and outputs?
- What happens in between and how?
- How do we know when our implementation is correct?

- Abstraction is key for many of these steps

Implementation

- Connecting our design and our implementation
- Maintaining a separation of the logic of our solution from the implementation
- Tools that help us to manage our abstractions

Testing and Debugging

- Testing procedures are designed (often ahead of time)
- Testing procedures for different pieces of the code base
- Tools that allow us to analyze what our code is doing and what it is “thinking”
- Isolation of “buggy” code

Ethics in Computer Science

- Processes for detecting and analyzing ethical questions that can arise
- Privacy
- Intellectual property

My Assumptions About You

- At least one introductory course in programming
- Experience with java, including:
  - Control structures: if-then-else, while, for, case
  - Basic data types: integers, floats, chars, strings
  - Exposure to java objects

My Assumptions About You

- Laptop for lab and project work

Course Goals

By the end of this course, you should be able to:

- Analyze simple computing problems and define the requirements that are appropriate to their solution.
- Apply design and development principles to the implementation of a solution to the computing problems.

Course Goals

Continued:

- Evaluate and analyze the performance of your implementations, and use this information to make further implementation changes.
- Use an integrated development and debugging environment.
- Evaluate and analyze the ethical, professional, legal and social issues that are faced by computer scientists.

Sources of Information

- Textbooks:
  - Introduction to Java Programming: Comprehensive Version, Eighth Edition, Y. Daniel Liang, 2008, Pearson/Prentice-Hall. (Seventh edition is possible)
  - Ethics & Technology: Ethical Issues in an Age of Information and Communication Technology, Second Edition, Herman T. Tavani, 2007, Wiley.
- Desire to Learn
- Class web page:

  www.cs.ou.edu/~fagg/classes/cs2334

In the engineering library…

Class Schedule

www.cs.ou.edu/~fagg/classes/cs2334/schedule.html

- Lecture plans
- Required reading
- Assignment posting and due dates

As changes are made, they will be posted here

Channels of Communication

- Lecture
- Class email list: time-critical messages to the class
- Desire2Learn announcements
- Desire2Learn discussion group: you may post questions (and answers)
- Private email or office hours for non-public questions/discussions

Grading

- Components of your grade:
  - Midterm exams 1 and 2: 10% each
  - Final exam: 20%
  - Five projects: 41%
  - Ten labs: 15%
  - In-class participation: 4%
- Grades will be posted on the Desire2Learn
- Final grade boundaries will be selected based on the overall class distribution

Exams

- Assigned seating
- No electronic devices
- Grading questions must be addressed before the returned exams leave the classroom

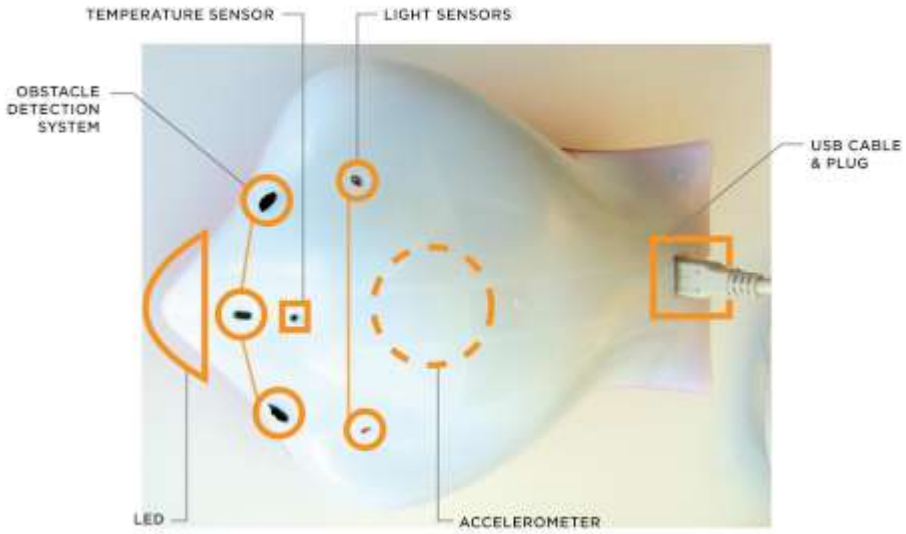
Labs and Projects

- A few: individual work
- Most: work in pairs
- Hand-in:
  - Digital components: D2L dropbox
  - Hardcopy components: to me or the TAs
- Grading questions must be addressed within one week of being returned

Platform for Labs and Projects

CMU Finch

- Variety of sensors
- Produce light, sound and movement
- Nice Java API



Group Grading

Group grades are a function of:

- Design
- Code correctness and readability
- Documentation

Individual grades:

- Group grade scaled by your personal contribution
- The scaling factor is determined in part by your fellow group member

Late Policies

- Labs must be handed in at the designated date/time
- Projects have some leeway:
  - 0-24 hrs: 20% penalty
  - 24-48 hrs: 40% penalty
  - 48+ hrs: 100% penalty

Classroom Conduct

- Ask plenty of questions
- Contribute to the discussions

- No: cell phone use (including texting)
- No: laptop use (except for classroom exercises)

Academic Conduct/Misconduct

Individual assignments:

- All work must be your own: no looking at or copying solutions from other students or from the net
- General discussion is OK (e.g., the fundamental skills that we are learning in class)

- When in doubt: ask me or one of the TAs

Academic Conduct/Misconduct

Projects:

- All work must be that of your group: no looking at or copying solutions from other groups or from the net
- General discussion is (again) OK

Secure your data

Next Time(s)

- Wednesday: Abstraction and Modularization (chapter 8)
- Thursday lab: JDK, compiling, Javadoc