Welcome to Com S 227

Please turn off your cell phone!
Welcome (or welcome back)

• Sorry!
  – You look familiar, but I can’t remember your name!
    • It could not possibly be because of old age

• Well, stop by my office sometime and introduce yourself
This Is Not A Powerpoint Slide

• I never use powerpoint slides
Who am I?

• Steve?

• Steve! [http://www.youtube.com/watch?v=ZOOlCh1K1og](http://www.youtube.com/watch?v=ZOOlCh1K1og)
But seriously

- 20-odd years teaching CS and math
- 8 years in industry, rejoined ISU in 2008
- MS in CS, PhD in Math
  - But you didn’t finish high school? Don’t tell them that!
What are we doing here

- Introduction to Object-Oriented Programming
#1: What is programming?

• “writing out a sequence of instructions for a machine to carry out?”

• Hmmm…
  – That’s a little like saying that creating a novel is accomplished by putting words together into sentences

• What about the *thinking* where you decide what the instructions should *do*?

• What does it mean to “design” software?
Designing software

• Suppose you have a few hundred lines of instructions
  – Tic-tac-toe game, print a loan table, sort a list of names...
  – Well, this is probably just a “program”

• Applications like Word or Firefox may involve several *million* lines of code
  – Too complex for one person to understand...
  – ...unless very carefully designed!
Object-oriented design

• This is where the “OO” comes in (item #2)
• Modern applications are too complex to be written as a simple sequence of instructions
• OO is a natural way of breaking down a complex system into components
  – Each component is simpler than the whole
  – You specify
    • What does each component do?
    • How do the components interact?
Analogy

• A typical car consists of approximately 30,000 parts
Analogy

• But it makes a lot more sense as a system of interacting components
• (Engine, Chassis, Steering...)
Objects

• In OO design and programming, the components are called “objects”
  – Within each component there are operations, containing instructions to execute...
  – But we understand and build an application as a system of interacting objects
Com S 227

• Emphasis of 227: designing, implementing, using objects effectively to make stuff
• It will still involve details of programming:
  – Variables and expressions for arithmetic and text
  – If-then control structures
  – Instructions for repetition, or “loops”
• Will also cover OO concepts such as inheritance and polymorphism
Which brings us to #3

• This course does have the word “introduction” in the title
  – But it’s “Introduction to Object-Oriented Programming”, not “Introduction to Programming”

• From the catalog: Prereq: Placement into MATH 143, 165, or higher; recommended: a previous high school or college course in programming or equivalent experience.
• In practice, 227 can be difficult for someone who has had no exposure to programming!
• Be sure you have the math background
  • Placement into Math 143 or calculus
• Consider taking Com S 127 first
  • Programming at a slower pace, without “objects”
Should I be taking this course?

• Many people with no previous programming experience have done just fine
  – Good study and time management skills?
  – A’s in Calculus, good at logical thinking?
  – National merit scholar?
• ...but many people, even with some programming experience, have a lot of trouble!
  – Last spring: 500 students, 120 drop/fail
Why do so many people end up taking the course twice?

• First time: learn not to procrastinate
• Second time: learn about object-oriented programming
• Being self-motivated is exceptionally important
  – Look around the room...
Is the course too hard?

• Probably not
• Standards are not hypothetical, but based on real-world expectations of skills of graduates
• “Actually, those courses [227 and 228] should have been a lot harder.”
  – Nicole Bruck, former student and 227 TA, currently a project manager at Microsoft
I’ve seen all this stuff before, can I skip Com S 227?

• Maybe. Not so fast.
  – Be careful what you wish for
  – 227 provides experience with much more complex projects than in your community college course, rigor in design and testing to a specification

• Take a look at the archive from last semester
  – Can you do homeworks 3 and 4 easily? How about the final exam review problems?
I don’t want to be a programmer

• What does programming have to do with computer science?
• Is programming too hard? Too easy? Too boring?
• If I major in computer science, do I have to be a programmer?
  – No!
Course organization

• There are about 540 students this spring
• 3 lecture sections like this one
  – MWF at 1:10 and 3:10
• 15 hands-on lab sections
  – Two hours each week in groups of 40
  – Opportunity to try things out where there are TAs and people around to answer questions!
Learn me to program!

• How to develop programming skills?  
  http://www.youtube.com/watch?v=6AOpomu9V6Q

• You don’t actually learn that much by watching someone else write code...
  – So I don’t need to come to class? Cool!

• *Take initiative, try examples, do them in different ways, make up new problems, get stuck, ask questions*...
Read the syllabus

• What’s a syllabus??
• See: http://www.cs.iastate.edu/~cs227/syllabus.html