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
- I hate them
Who am I?

• Steve!
  – [http://www.youtube.com/watch?v=ZOOICh1K1og](http://www.youtube.com/watch?v=ZOOICh1K1og)
  – 20-odd years teaching CS and math
  – 8 years in industry
  – MS in CS and 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

  3.                          2.                          1.
#1: What is programming?

• Well, it isn’t computer science
• Programming is like *craftsmanship* in building something
  – The end product is called *software*
    • *Applications and systems*
    • *Informally: “code”*
• That’s one reason I like it, I have always liked to build stuff!
Analogy

• In construction, architects *design* houses, craftsmen *build* them
  – Architects usually do not have the skills to build the things they design!

• In software, it never seems to work that way
  – There are software designers who don’t actually write code
  – ...but they all start out *learning* to write code
The way it is

• Virtually every university program in CS or SE or CPRE starts out with a couple of semesters of programming
• So, here we are 😊
You didn’t answer the question

• What is programming?
  – It just means writing out a sequence of instructions for a machine to carry out.
  – Instructions are usually very basic, e.g., “add these two numbers together”

• Any sequence of instructions can be called a “computer program”

• But, what does it mean to “design” software?
Designing software

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

• Applications like Word or Firefox may involve a 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 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 is technically an “introductory” course
• In practice, it can be difficult for someone who has had no exposure to programming!
  • Partly because we are introducing *programming* and *objects* at the same time
• Be sure you have the math background
  • Placement into Math 142 or calculus
• Consider taking Com S 104 first
  • Programming at a slower pace, without “objects”
Course organization

• There are 400 students
• 4 lecture sections like this one
  – MWF at 8:00, 9:00, 10:00 and 1:10
• 20 hands-on lab sections
  – Two hours each week in groups of ~20
  – 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 much by watching someone else write code

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

• See: http://www.cs.iastate.edu/~cs227/syllabus.html