COMP 211
Principles of Program Design
Spring 2010

Prof. Robert “Corky” Cartwright
Rice University
Instructor Information

Corky Cartwright (cork@oplink.net, cork@rice.edu]
  • Duncan Hall (DH) 3104, 713-348-6042
  • www.cs.rice.edu/~cork
  • Office hours: MWF 11:00-12:00, and by appointment

  • Vivek Sarkar [vsarkar@rice.edu]
    • Duncan Hall (DH) 3103, 713-348-5718
    • www.cs.rice.edu/~vsarkar
    • Office hours: TBA

  • Zung Nguyen [dxnguyen@rice.edu]
    • Duncan Hall (DH) 3098, 713-348-3835
    • www.cs.rice.edu/~dxnguyen
    • Office hours: TBA

  • Teaching Assistants:
    • Shams Imam
    • Dragos Sbirlea
    • Alina Simion
    • Robert Brockman
    • Mina Yao
    • Kiran Nair
Course Materials

Course web page:
https://wiki.rice.edu/confluence/display/cswiki/211
(or search for the terms “confluence” “comp” “211”)

- This is a new wiki associated with the new CLE@R educational computing facility.
- If you forget the long URL given above, you can simply go to www.cs.rice.edu/~cork and follow the link to Comp 211.
- Course information like TAs, office hours, etc. are covered on the course web site. Some of that information is still TBA but will be resolved by the end of this week (15 Jan 2010).
Course Demands

- Prerequisite: some programming experience

- Workload: difficult, time-consuming course, requiring about 10-12 hours outside of lectures each week.

- Weekly homework assignments, all of which involve programming; submitted electronically

- Two “segmented” take-home exams.

- Grading: 50% homework  50% exams
Course Mechanics

Immediate Concerns

• Take Short Entrance Survey (on wiki web page)
• Contact Zung (by email or in person) if you have not already signed up for a lab section. Available times: M 2, M 3:30, Tu 2:30 (approx 90 minutes long)
• Labs begin today at 2pm. First lab is primarily course emigration.
• HW0 posted on the course wiki is due on Wednesday!
  • Sign up for a CLE@R account
  • Pick a homework partner, preferably someone in the same lab (pair-programming)
  • Download PLT Scheme 4.2.3
Course Policies

• No late assignments will be accepted--except assignments that draw on 7 slip days granted to each student for use during the term.

• Programs are graded according to a precise formula described on the course wiki. Whether a program “works” or not constitutes less than half the grade for that program. Program design (including design documentation), unit test cases, program coding style, code documentation all matter a great deal.

• We know from experience as professional software developers that good design, (the right form of) program documentation, appropriate tests, and clean program are critical in practice.
Why Focus on Program Design?

- *Program Design* is the core of Computer Science
  - Why not *Algorithms*?
  - Software is the dominant artifact of modern civilization
    - “Code is Law” [Lessig]
      http://harvardmagazine.com/2000/01/code-is-law.html
    - Code regulates many aspects of our lives
    - Code is emerging medium for expressing knowledge
      (HTML, XML, PDF)
    - Code is omnipresent in manufactured goods
      Airplanes, cars, blenders, phones, toys, greeting cards, ...
- *Program Design* is intellectually challenging
Why COMP 211?

- Repackaging of innovative curriculum for better marketing
  - Comp 210/212 developed at Rice, with major NSF funding
  - DrScheme, DrJava, *How to Design Programs, OO Design Notes* were developed to support this curriculum

- How it is different from other introductory courses?
  - Focus on principles of design, not on details of a particular language or software platform
    - Programming as mathematics
    - Lean, elegant linguistic frameworks (Core Scheme/Java)
    - **Data definitions (types)** drive the design process
  - Follow leading edge software engineering practices

- Program design is **not coding** (e.g., iterators not Java for loops) anymore than architecture is **drafting**.
Course Overview

- Functional program design in Scheme (6 wks)
  - Data-directed (functional) program design 2-10
  - Algorithm design 11-14
  - Applied functional programming and review 15-17

- Object-oriented (OO) design in Java (8 wks)
  - Rudiments of the OO programming model 18-19
  - Data-directed OO program design 20-25
  - Advanced Java constructs (inner classes, generics) 26-29
  - Fundamental data structures and algorithms 30-38
  - Event-driven programming, GUIs, concurrency 39-40
Design pattern = template for solving a computational problem

- union/composite/interpreter
- singleton
- command/strategy
- factory method
- visitor
- model-view-controller
- decorator?
- template method?
- adapter?
- factory?
Why Scheme?

- Functional programming (FP) is the key to understanding programming as mathematics.
- FP underlies many aspects of good programming design—particularly in the context of concurrent programming.
- Good notation for FP (provided by a functional language derived from mathematics) facilitates FP solutions.
- Scheme is the simplest functional language and we will use only the core constructs:
  - Function and constant definition
  - Function application
  - Conditionals
  - Structure definitions
  - Local definitions (blocks) and single assignment (binding)
- Simple formal semantics: rewrite program source text. Scheme is an extension of conventional arithmetic.
- Very good pedagogic IDE: DrScheme.
Why Java?

- Object-oriented (OO) programming is a powerful generalization of functional programming that decomposes programs into a collection of code units called classes.
- Widely used in mainstream software development.
- Classes support incremental test-driven development.
- Java/C# now dominate application programming
  - Only Java is almost completely platform independent: “write once; run anywhere.”
  - A good (but not great) OO language.
  - Efficiently implemented except for VM startup and memory footprint.
- Very good pedagogic design environment (IDE): DrJava
Future Options

• F# (supported as a language in Microsoft .NET) is a modern OO language with a rich functional subset akin to Ocaml. Potential problems: complex type system, only runs on Windows platform, no pedagogic IDE, not Java.

• Scala is a modern OO language with a rich functional subset that runs on the Java Virtual Machine. Potential problems: primarily an academic research project, no pedagogic IDE, not Java.
COMP 211 Prepares You to...

- Design well-engineered programs without focusing on language features.
- Work as a junior Java software developer.
- Learn deeper concepts of computing:
  - Programming languages (design and implementation)
  - Formal methods (program semantics, verification, formal logic)
  - Algorithms (including ideas central to artificial intelligence, data-mining, bioinformatics)
  - Systems (networks, operating systems, compilers)
  - Software engineering (application architecture, test-driven development, unit testing, refactoring)
More on Grading

- **Homeworks (50%)**
  - Usually once a week, Friday-to-Friday timeframe is most common.
  - Work jointly in teams of two. Do **not** divide work up.
  - No late homework will be accepted, except for 7 *slip* days to be used during the term. A fraction of a day counts as a full day. Advice: hoard your slip days until late in the course.

- **Exams (50%)**
  - Sample exams will be available online.
  - Take home, pledged, closed book.
  - First exam during week 7 counts 20%
  - Second exam during last week 15 counts 30%
How to Succeed

• Do the reading before class
  • This will help you understand our lectures.
• Attend class and mandatory labs
  • Reading, lectures, and hands-on instruction complement one another
  • Exam questions will be drawn from all three.
• Take homework assignments seriously; follow our examples
  • A program that simply “works” may get a failing grade.
• Use office hours
  • Asking questions is a sign of intelligent life.
• Ask questions in class
  • No dumb questions; only inappropriate ones
For Next Lecture

• Fill out entrance survey
• Contact Zung if you have not already requested a lab section
• Attend your assigned lab today or tomorrow
• Make sure you have done Homework 0
  • Already posted online on web-page
  • Due next class (Wednesday)
  • Individual help is available in lab!
• Next class will cover:
  • More details on how to create and submit programming assignments.
  • The building blocks of functional programming
Copyright

- COMP 211 lecture notes are copyrighted by Profs. R. Cartwright, V. Sarkar, W. Taha, and Z. Nguyen, Rice University.
- COMP 211 students can copy and modify these materials freely as long as this slide is preserved
- Commercial use requires written permission.
- Publicly available materials for the course are licensed under the Creative Commons (CC) license
  - Seecreativecommons.org basic idea, and then course Twiki for details
About Your Instructors

- Our research programs are concerned with
  - Improving programming technology including
    - Language design
    - Developing optimizing compilers
    - Programming tools: IDEs, “soft” typers, testing frameworks
    - Programming pedagogy
  - Improving programmer productivity, using
    - Automatic program generation,
    - Lightweight formal verification (type systems),
    - Higher-order typed languages (ML, Haskell, Java+, Fortress)
  - Improving productivity of people building:
    - Real-time and embedded systems,
    - Hardware (microprocessors or “chips”),