COMP 211: Principles of Program Design

Instructors:  
Prof. Robert “Corky” Cartwright  
Dr. Stephen Wong  

Staff:  
Alina Sbirlea  
Kamal Sharma  
Nicholas Coltharp  

Lectures:  
Duncan Hall (DH) 1075  
Time: MWF 10:00-10:50am

Lab:  
Ryon 102  
Time: Tuesday 10:50am-12:05pm

Office Hours:  
- Dr. Cartwright: MWF 9:00 AM - 10:00 AM  
- Dr. Wong: See home page  
- Alina: MWF 11:00 AM - 12:00 PM  
- Kamal: MF 3:00 PM - 5:00 PM  
- Nicholas: TR 7:00 PM - 8:30 PM

e-mail the entire class: comp211s11 at owlspace-ccm.rice.edu

e-mail just the staff: comp211 at rice.edu

Introduction

This course is an introduction to the fundamental principles of programming. The focus is on systematic methods for developing robust solutions to computational problems. Students are expected to have experience writing interesting programs in some credible programming language (e.g., Python, Java, Scheme, C#, C++, Visual Basic .NET, PRL, Scheme, Lisp, etc.) but no specific programming expertise is assumed. The course is targeted at potential Computer Science majors but mathematically sophisticated non-majors are welcome. We expect students to be comfortable with high-school mathematics (primarily algebra, mathematical proofs, and induction) and the mathematical rigor and vocabulary of freshman calculus. Success in the course requires a deep interest in the foundations of computer science and software engineering, self-discipline, and a willingness to work with other people on programming projects. Topics covered include functional programming, algebraic data definitions, design recipes for writing functions, procedural abstraction, reduction rules, program refactoring and optimization, object-oriented programming emphasizing dynamic dispatch, OO design patterns, fundamental data structures and algorithms from an OO perspective, simple Graphical User Interfaces (GUIs), and an exposure to the challenges of concurrent computation.

Students will learn the practical skills required to write, test, maintain, and modify programs. Labs and assignments use the Scheme and Java programming languages.

Text For Scheme: **How to Design Programs** by Felleisen et al. QA76.6 .H697 2001 (Available online; no purchase is necessary.)

**DrScheme/DrRacket**: Please download and use the DrScheme/DrRacket system available from the [Racket download site](http://www.racket-lang.org). To avoid compatibility problems, please make sure you use a Version numbered 5.0.2

- [DrRacket Tips and Traps](http://docs.racket-lang.org/tips/)

Notes for Java: **Object-oriented Design**

References for Java

- [Principles of Object-Oriented Programming by Zung Nguyen and Stephen Wong](http://www.amazon.com/)  
  An online self-contained introduction to OOP in Java roughly corresponding to the former Comp 212 course. It is reasonably complete, but still under construction.

- [Design Patterns Lens](http://www.javadoc.com/designpatterns/)

- [Index to online Java Tutorials by Sun Microsystems](http://java.sun.com/docs/books/tutorial/)  
  The Sun tutorials refer to the full Java language not the Elementary and Intermediate language levels supported by DrJava. Nevertheless, they cover many important language details in depth, such as the complete collection of primitive operators on primitive data types.

- [Java Basics by Fred Swartz](http://www.javabasics.com/)  
  A clearly written traditional introduction to Java that focuses on Java mechanics rather than OO Design. It can be helpful in learning the mechanics of writing full Java code. Please ignore what he says about program design.

- [Java Notes by Fred Swartz](http://www.javabasics.com/JavaNotes.html)  
  A reasonably comprehensive Java reference that is a good supplement to the official Sun documents.

**DrJava**: Please download and use the DrJava pedagogic programming environment available from [drjava.org](http://drjava.org). You must install either the Java 5 or Java 6 JDK on your machine for DrJava to work.  
(Addendum: DrJava releases starting in 2011 can also run using a Java 7 JDK.) If you machine is running some flavor of Windows or Linux, go to the [Sun Download Site for the Java SE 6.0](http://java.sun.com/javase/downloads/index.jsp). Make sure that you download the JDK not the JRE. If you have a Mac, a Java JDK is available from Apple. In fact, it is part of the standard Mac OS X software package.

DrJava will run on either the Apple Java 5.0 or Java 6.0 JDK.
Entrance Survey

Please fill out this survey.

Confirm that You Can Attend Lab

The lab is an essential component of the course. Every student should attend lab on Tuesday 10:50-12:05 in Ryon 102. If you have a conflict, you probably should drop the course. If you have a conflict, but still want to take the course, please come see Prof. Cartwright or Prof. Wong to evaluate your options.

Computing Environment

All Comp 211 programming assignments will be run and graded on the CLEAR (Linux) educational computing facility. The Ryon 102 lab is equipped with machines connected to the CLEAR network. For instructions on how to use CLEAR, see the CLEAR web page.

Errata

This wiki for Comp 211 is a revision of last year's and may contain a significant number of broken links and typos. If you notice an error in the wiki, please send email to the Comp 211 staff.

Course Schedule

Note that future date schedules are only guidelines. Future homeworks and slides may contain materials from previous Comp 210 and Comp 212 classes. New material will be provided before the corresponding class. There will only be two exams in the course: one given on functional programming during week 7 and one on object-oriented programming given during in the last week of the course. Both are take-home exams. There is no final examination.

<table>
<thead>
<tr>
<th>Day</th>
<th>Date (2011)</th>
<th>Topic</th>
<th>Reading</th>
<th>Lectures</th>
<th>Problems</th>
<th>Due (2011)</th>
<th>Lab</th>
<th>Supplements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mon Jan 10</td>
<td>Introduction &amp; Scheme Primitives</td>
<td>L 1</td>
<td>HW 0</td>
<td>W Jan 12</td>
<td>Lab 0</td>
<td>Pair Programming</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Wed Jan 12</td>
<td>Function definitions and conditionals</td>
<td>Read Chs. 1-10</td>
<td>L 2</td>
<td>HW 1</td>
<td>F Jan 21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Fri Jan 14</td>
<td>Data Definitions &amp; The Design Recipe</td>
<td>Read Chs. 11-13</td>
<td>L 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Wed Jan 19</td>
<td>Data-directed design</td>
<td>Read Chs. 14-15</td>
<td>L 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Fri Jan 21</td>
<td>Data-directed design: trees</td>
<td></td>
<td>L 5</td>
<td>HW 2</td>
<td>Mon Jan 31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Mon Jan 24</td>
<td>Mutually Referential Data Definitions</td>
<td>Read Chs. 16-17</td>
<td>L 6</td>
<td></td>
<td>Lab 2</td>
<td>Class Demo</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Wed Jan 26</td>
<td></td>
<td>Read Ch. 18</td>
<td>L 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Fri Jan 28</td>
<td>Local Definitions and Lexical Scope</td>
<td>Read Chs. 19-20</td>
<td>L 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Mon Jan 31</td>
<td>Functional Abstraction and Polymorphism</td>
<td>Read Chs. 21-22</td>
<td>L 8</td>
<td>HW 3</td>
<td>Mon Feb 7</td>
<td>Lab 3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Wed Feb 02</td>
<td>Functions as Values</td>
<td>Read Ch. 24</td>
<td>L 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Fri Feb 04</td>
<td>Lambda the Ultimate</td>
<td>Read Chs. 25-28</td>
<td>L 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Mon Feb 07</td>
<td>Generative Recursion</td>
<td>Study Chs. 25-28</td>
<td>L 11</td>
<td>HW 4</td>
<td>Mon Feb 14</td>
<td>Lab 4</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Wed Feb 09</td>
<td>Complexity and Accumulators</td>
<td>Read Chs. 29.1-2</td>
<td>L 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Fri Feb 11</td>
<td>Accumulators and Tail Calls</td>
<td>Read Chs. 30-32</td>
<td>L 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Mon Feb 14</td>
<td>Clever Programming With Functions</td>
<td>Review prior readings</td>
<td>L 15</td>
<td>HW 5</td>
<td>Mon Feb 21</td>
<td>Lab 5</td>
<td>210 Exam 1 210 Exam 2 Solution to 12.4.2</td>
</tr>
<tr>
<td>16</td>
<td>Wed Feb 16</td>
<td>Exam Review</td>
<td>Review prior readings</td>
<td>L 16, L 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Fri Feb 18</td>
<td>On to Java</td>
<td>OO Design Notes Ch 1.1-1.4</td>
<td>L 18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Mon Feb 21</td>
<td></td>
<td>OO Design Notes Ch 1.1-1.4</td>
<td>L 18</td>
<td>HW 6 (Optional) Exam 1</td>
<td>Mon Mar 07</td>
<td>Lab 6</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Wed Feb 23</td>
<td>Java Design Recipe</td>
<td>OO Design Notes Ch 1.1-1.4</td>
<td>L 19</td>
<td></td>
<td>IntList IntListTest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Day</td>
<td>Event</td>
<td>Chapters/Notes</td>
<td>Lecture</td>
<td>HW</td>
<td>Lab</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------</td>
<td>---------</td>
<td>-----</td>
<td>------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Fri</td>
<td>Defining Inductive Data in Java</td>
<td>OO Design Notes Ch 1.5</td>
<td>L 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M-F</td>
<td>Spring Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Mon</td>
<td>Static Class Members and the Singleton Pattern</td>
<td>OO Design Notes Ch 1.6</td>
<td>L 21</td>
<td>HW 7</td>
<td>Lab 7</td>
<td>ObjectList ObjectListTest</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Wed</td>
<td>Polymorphism and Interfaces</td>
<td>OO Design Notes Ch 1.8</td>
<td>L 22</td>
<td></td>
<td></td>
<td>ComparableList ComparableListTest</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Fri</td>
<td>Handling Exceptions and Errors</td>
<td>OO Design Notes Ch 1.9-1.10, 1.12</td>
<td>L 23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Mon</td>
<td>The Strategy and Visitor Patterns</td>
<td>OO Design Notes Ch 1.9, 1.11</td>
<td>L 24</td>
<td>HW 8</td>
<td>Lab 8</td>
<td>IntList IntListVisitor IntListTest</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Wed</td>
<td>Visitors, Visitors, Visitors ...</td>
<td>OO Design Notes Ch 1.11</td>
<td>L 25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Fri</td>
<td>Accepting Reality: Full Java</td>
<td>OO Design Notes Ch 1.13</td>
<td>L 26</td>
<td></td>
<td></td>
<td>IntList IntListTest</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Mon</td>
<td></td>
<td>OO Design Notes Ch. 1.10, 1.13</td>
<td>HW 9</td>
<td>Fri Apr 4</td>
<td>Lab 9</td>
<td>List ListTest</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Wed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>School Holiday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Mon</td>
<td>Simple Generics in Java</td>
<td>OO Design Notes Ch 1.13.4 (contains advanced material as well)</td>
<td>L 29</td>
<td></td>
<td>Lab 10</td>
<td>lec29_code.zip (incl. genList)</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Wed</td>
<td>Mutation and Bi-Directional Linked Lists</td>
<td>OO Design Notes Ch. 1.13</td>
<td>L 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Fri</td>
<td>BLLists Continued</td>
<td>OO Design Notes Ch 2.1</td>
<td>L 30</td>
<td></td>
<td></td>
<td>BigBiList BigBiListTest</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Mon</td>
<td>Mutable Trees</td>
<td>OO Design Notes Ch 2.1</td>
<td>L 32</td>
<td>HW 10</td>
<td>Wed Apr 13</td>
<td>TreeMap TreeMapTest OOTreeMapTest</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Wed</td>
<td>Mutable Trees</td>
<td></td>
<td>L 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Fri</td>
<td>Mutable Trees</td>
<td></td>
<td>L 32</td>
<td></td>
<td></td>
<td>FunctionalQuicksort</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Mon</td>
<td>Mutable Trees and OO Data Structure Review</td>
<td>OO Design Notes</td>
<td>L 32</td>
<td></td>
<td>Lab 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Wed</td>
<td>OO Sorting Algorithms</td>
<td>CNX Module on Sorting (incl. insertion and selection sort animations)</td>
<td>L 36</td>
<td>HW 11</td>
<td>Milestone 1: Mon. Apr 18 Milestone 2: Fri. Apr 22 Design Patterns Slides Design Patterns Paper Sorter Demo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Fri</td>
<td>Graphical User Interfaces</td>
<td></td>
<td>L 37, 2010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Mon</td>
<td>Fast Searching and Memoization</td>
<td></td>
<td>L 38</td>
<td></td>
<td>Lab 13</td>
<td>MyHashMap.java MyHashMapTest.java BetterMath.java</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Wed</td>
<td>Fast Sorting Methods</td>
<td></td>
<td>L 39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Fri</td>
<td>Review</td>
<td></td>
<td>L 40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Grading, Honor Code Policy, Processes and Procedures

Grading will be based on your performance on homeworks (worth 50%) and exams (20% for first exam, and 30% for the second exam).

Take-home exams, which are pledged under the honor code, test your individual understanding and knowledge of the material. Collaboration on exams is strictly forbidden.

We will normally use the OwlSpace Comp 211 Announcements and General Discussion forum to post important announcements and discussions related to the class, particularly Q&A on class assignments. The Owlspace e-mail alias to the entire class (see above) will also be used.

### Other Mailing Lists:
- **cs-events-l@mailman.rice.edu:**
  - Announcements relating to talks and other interesting events hosted by the CS departments.
  - Subscription to this list is optional but highly recommended
Questions

If you have a question about homework – you’re not sure what is expected for a given problem, you haven't received feedback from a previous assignment, or you don't understand or agree with the assessment of your work, for example – you can raise the question with a TA in lab or on the class mailing list. It is preferable to send a question out for the whole class to discuss so that everyone can benefit from each other's contributions as well the responses from the staff. Please restrict messages sent to only the staff to more sensitive matters that do not warrant viewing by the entire class. If you still don't feel that your concerns have been addressed, you may wish to contact Prof. Cartwright or Prof. Wong directly.

Homeworks:

Homeworks help you verify your understanding of the material and prepare you for the exams. You are encouraged to discuss the homework problems with the instructors and staff. Help from other students, including Comp 211 graduates, is also encouraged (but should be cited), although that does not include giving or receiving complete answers. All homework partners are responsible for knowing all the submitted material. If you fail to understand the homework solutions, you won't succeed on the exams.

Homeworks will generally be handed out on Fridays, and will be due before class the following Friday.

You are expected to work in groups of two.

You may change partners during the semester.

Partners should work together on all aspects of the homework -- all students are expected to contribute equally. You and your partner should hand in exactly one solution.

Late homework will not be accepted with one exception. Every student is allotted 7 slip days. Each whole day or fraction of a day that an assignment is late counts as a slip day. Each student in the pair submitting a late assignment must spend the requisite number of slip days. Since assignments get progressively harder during the semester, we strongly encourage you to hoard your slip days for use near the end of the term.

We recommend that you review the homework guide as you develop your solutions. Review the submission checklist when you turn in your homework.

Your work will be graded as documented on the grading page.

Reading: For each lecture, there is associated reading. Students are required to complete the reading before the class associated with this reading.

Other Resources

- Practical matters:
  - DrScheme Programming Environment
- Special interest groups:
  - CSters
  - Computer Science Club
- ... (please send in suggestions!)

Additional References

Here is a nice article about the basic approach taken in this course.

More on CS

<table>
<thead>
<tr>
<th>Title</th>
<th>QA76 .D448 1993</th>
</tr>
</thead>
<tbody>
<tr>
<td>The New Turing Omnibus, A. K. Dewdney</td>
<td></td>
</tr>
<tr>
<td>Algorithmics: The Spirit of Computing, David Harel</td>
<td>QA76.9 .A43 H37 2004</td>
</tr>
<tr>
<td>Computers Ltd.: What They Really Can't Do, David Harel</td>
<td>QA76.5 .H3575 2000</td>
</tr>
<tr>
<td>Great Ideas in Computer Science, Alan W. Biermann</td>
<td>QA76 .B495 1997</td>
</tr>
<tr>
<td>Computer Science: An Overview, J. Glenn Brookshear</td>
<td>QA76 .B743 1997</td>
</tr>
<tr>
<td>Metamagical Themas, Douglas Hofstadter</td>
<td>Q335 .H63 1985</td>
</tr>
</tbody>
</table>

More on Functional Programming

<table>
<thead>
<tr>
<th>Title</th>
<th>QA76.73 .S34 F75 1996</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google's MapReduce</td>
<td>(Online)</td>
</tr>
<tr>
<td>The Little Schemer, Friedman &amp; Felleisen</td>
<td>QA76.73 .S34 F77 1996</td>
</tr>
<tr>
<td>The Seasoned Schemer, Friedman &amp; Felleisen</td>
<td>QA76.73 .S34 F77 1996</td>
</tr>
<tr>
<td>Developing Applications with Objective Caml, Emmanuella Chailloux, Pascal Manoury, and Bruno Pagano</td>
<td></td>
</tr>
<tr>
<td>The Haskell School of Expression: Learning Functional Programming Through Multimedia, Paul Hudak</td>
<td>QA76.62 H83 2000</td>
</tr>
</tbody>
</table>

More on Object-Oriented Design in Java
<table>
<thead>
<tr>
<th>Book Title</th>
<th>Author(s)</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principles of Object-Oriented Programming</td>
<td>The Connexions Curriculum for the former Comp 212 course</td>
<td></td>
</tr>
<tr>
<td>The Java Programming Language, Arnold &amp; Gosling</td>
<td>Gosling was the principal designer of Java 1.0</td>
<td></td>
</tr>
<tr>
<td>Thinking In Java, Bruce Eckel</td>
<td>The 3rd edition is available free.</td>
<td></td>
</tr>
<tr>
<td>Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson &amp; Vlissides</td>
<td>The “Bible” of OO software design.</td>
<td></td>
</tr>
<tr>
<td>Head First Java, Bert Bates &amp; Kathy Sierra</td>
<td>A fun read while you get introduced to Java and learn how to think like a Java Programmer.</td>
<td></td>
</tr>
<tr>
<td>Head First Design Patterns, Freeman &amp; Freeman</td>
<td>An accessible introduction to Design Patterns.</td>
<td></td>
</tr>
</tbody>
</table>

**More on Data Structures and Algorithms**


**Accommodations for Students with Special Needs**

Students with disabilities are encouraged to contact me during the first two weeks of class regarding any special needs. Students with disabilities should also contact Disabled Student Services in the Ley Student Center and the Rice Disability Support Services.