Intro to CS: Objects, Events, and Graphics

Instructors

Prof. Andrea Danyluk
TCL 305
597-2178
andrea@cs.williams.edu
Office Hours TBA

Prof. Iris Howley
TCL 308
597-4633
iris@cs.williams.edu

TA Hours will be posted on the course webpage

Lectures MWF 9-9:50 or 10-10:50 in SSL 030A

Labs M 1pm–4pm, M 7pm–10pm, T 8:30am–11:20am in TCL 217a

Web Page http://www.cs.williams.edu/~cs134/

Texts

We will use the following text book, which is available at the bookstore:


Course Objectives

Computing is central to many aspects of our lives and the world. This course introduces fundamental ideas in computer science and builds the skills necessary to create computer programs in the Java programming language, with an emphasis on graphics and user interfaces. Students learn to design programs in a wide range of application areas, from games to spam filters and image editing to scientific simulations. Programming topics include object-oriented programming, control structures, arrays, recursion, and event-driven programming, as well as how to construct correct, understandable, and efficient programs. This course is appropriate for all students who want to create software and have little or no prior computing experience.

Course Work

There will be weekly lab programming assignments. All programs will be graded on design, documentation and style, correctness, and efficiency. Programs should be turned in electronically by the due date. We will go over how to submit work in lab.

Attendance in lab is mandatory. Unapproved absence will result in zero credit for that week’s lab.

To accommodate your busy schedules and unanticipated obstacles, you may use a maximum of three free late days during the course of the semester. A late day permits you to hand in a regular lab assignment up to 24 hours late, without penalty. Once those late days are exhausted, late labs will be penalized one letter grade per day. Programs will not be accepted more than four days late. When using a late day, please email Prof. Danyluk to tell us that you are doing so.

There will also be a midterm exam and a final exam, as well as two larger Programming Projects. The first Project will occur around Reading Period, and the second during the last couple weeks of the semester. Homework exercises (non-programming assignments) may be assigned and collected
in class periodically and there may be in-class quizzes. Note that late days may not be used for any assignments other than regular weekly labs.

Grades will be determined roughly as follows:

- Labs: 30%
- Projects: 10%-15% each
- Midterm: 15%
- Final exam: 20%
- Homework & other: 5-10%

In addition to the 6 hours we spend together during our class and lab time, you should expect to spend at least 10 hours per week on the academic and creative work related to class. Keep in mind that this is an average weekly work load and that the number of hours you spend will naturally vary from week to week. If you find that you are consistently spending considerably more (or less) time to engage with this course academically, please contact me so that we can determine the best course of action as you approach the materials.

---

**Honor Code**

Homework and lab assignments are to be the sole work of each student unless the assignment explicitly states otherwise. Students may discuss issues related to an assignment, provided that such discussions are cited in the material turned in. However, students may not collaborate on designing or writing code. Uncredited collaborations or use of resources outside of those provided on the course web site will be considered a violation of the honor code and will be handled appropriately. Restrictions on collaboration and use of external sources will be greater for programming projects and exams. For a full description of the Computer Science Honor Code, please see [https://csci.williams.edu/the-cs-honor-code-and-computer-usage-policy/](https://csci.williams.edu/the-cs-honor-code-and-computer-usage-policy/). If in doubt of what is appropriate, do not hesitate to ask us.
Tentative Schedule

This will undoubtedly change as we begin to explore these topics.

<table>
<thead>
<tr>
<th>Date</th>
<th>Mon</th>
<th>Wed</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 8</td>
<td>Graphics, Events</td>
<td>Variables, Numbers</td>
<td>Introduction</td>
</tr>
<tr>
<td></td>
<td><em>Chapter 1, 2</em></td>
<td><em>Chapter 3</em></td>
<td><em>Preface</em></td>
</tr>
<tr>
<td>Sep 11–Sep 15</td>
<td>Primitive Types</td>
<td>Classes</td>
<td>Conditionals</td>
</tr>
<tr>
<td></td>
<td><em>Chapter 5</em></td>
<td><em>Chapter 6</em></td>
<td><em>Chapter 4</em></td>
</tr>
<tr>
<td>Sep 18–Sep 22</td>
<td>More Classes, Loops</td>
<td>Loops, Active Objects</td>
<td>Declarations, Scope</td>
</tr>
<tr>
<td></td>
<td><em>Chapter 7</em></td>
<td><em>Chapter 9</em></td>
<td><em>Chapter 8</em></td>
</tr>
<tr>
<td>Sep 25–Sep 29</td>
<td>Images</td>
<td>Interfaces</td>
<td>Active Objects</td>
</tr>
<tr>
<td>Oct 2–Oct 6</td>
<td><em>Chapter 10</em></td>
<td>GUIs</td>
<td></td>
</tr>
<tr>
<td>Oct 9–Oct 13</td>
<td>Reading Period</td>
<td>GUIs</td>
<td></td>
</tr>
<tr>
<td>Oct 16–Oct 20</td>
<td>GUIs</td>
<td>Recursion</td>
<td>Recursion</td>
</tr>
<tr>
<td></td>
<td><em>Chapter 12</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oct 23–Oct 27</td>
<td>Recursion</td>
<td>For Loops</td>
<td>2D Arrays</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Chapter 13</em></td>
<td><em>Chapter 14, 15</em></td>
</tr>
<tr>
<td>Oct 30–Nov 3</td>
<td>Arrays</td>
<td>Collections</td>
<td>Inheritance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><em>Chapter 17</em></td>
</tr>
<tr>
<td>Nov 6–Nov 10</td>
<td>Strings</td>
<td>Strings</td>
<td>OO Design</td>
</tr>
<tr>
<td></td>
<td><em>Chapter 16</em></td>
<td></td>
<td><em>Chapter 21</em></td>
</tr>
<tr>
<td>Nov 13–Nov 17</td>
<td>Exceptions</td>
<td>Files, Streams</td>
<td>Networks</td>
</tr>
<tr>
<td></td>
<td><em>Chapter 18</em></td>
<td><em>Chapter 19</em></td>
<td></td>
</tr>
<tr>
<td>Nov 20–Nov 24</td>
<td>Networks</td>
<td>Thanksgiving Recess</td>
<td>Thanksgiving Recess</td>
</tr>
<tr>
<td>Nov 27–Dec 1</td>
<td>Searching</td>
<td>Sorting</td>
<td>Sorting</td>
</tr>
<tr>
<td></td>
<td><em>Chapter 20</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dec 4–Dec 8</td>
<td>Advanced Topics</td>
<td>Advanced Topics</td>
<td>Wrap Up</td>
</tr>
</tbody>
</table>

The midterm is scheduled for the evening of Tuesday, October 24, with a review session at 7:00pm on October 19.
Welcome to CSCI 134!

How might we draw a Scribble?

(A collection of lines drawn when the user onMouseDragged)

Take a moment to write down your response, individually.

Discuss with a partner!
How might we draw a Scribble?
(A collection of lines drawn when the user onMouseDragged)

Is there a more concise way to program this [repetitive!] algorithm?

- Know when to stop.
- Decide how to take one step.
- Break the journey down into that step plus a smaller journey.

RECURSION

DEMO

ColorScribbleController
DEMO
Spirals Scribbles Recursive

CONGRATS!
Halfway through the semester!!

Administrative Details
• Lab: Monday and Tuesday this week, as usual
• Midterm: 6-7:45p OR 8-9:45pm, 10/24 TPL 203
• Sample midterm up on course Lectures page
• How to study?
  – Sample midterm
  – Class demos
  – Understand comments on your labs
  – Practice hand-writing codes:
    • Textbook problem sets (answers on course website/Resources)
• We will give you Java Swing & ObjectDraw reference sheets

Dragon, I need to know if any of the numbers in this list are odd:
(3142, 5798, 6550, 8914)

Sorry, I can only tell you if the first number of the list is odd.
But I need to know if any number in the list is odd, not just the first!

I’ll only look at the first number, but I’ll look at as many lists as you like.

What should Sam do?

(3142, 5798, 6550, 8914)

The first number is not odd.

(3142, 5798, 6550, 8914)

The first number is not odd.
The first number is not odd.

That’s an empty list! It can’t be odd.

None of the numbers the Sorcerer gave me were odd, thank you!

How can you know that? I only told you about the first number!

The lists I gave you were:

- (3142, 5798, 6550, 8914)
- (5798, 6550, 8914)
- (6550, 8914)
- (8914)
- ()

Tricky. Looks like you’ve discovered recursion.

Why did this work?
The lists I gave you were:
(3142, 5798, 6550, 8914)
(5798, 6550, 8914)
(6550, 8914)
(8914)
()

Steps for Recursion
1. Know when to stop.
2. Decide how to take one step.
3. Break the journey down into that step plus a smaller journey.

Steps for Recursion
• When to stop?
  – When list is empty
• What is the one step?
  – Check the first list item
• How to break the journey down?
  – Progress through each of first list items

Pseudocode
array = {3142, 5798, 6550, 8914}
printFirstOdd(array)

function printFirstOdd(a[]) {
  if a[] not empty? {
    is a[0] odd? → Print a[0]
  }
  printFirstOdd(a[1,a.length])
}

Let’s implement this recursive algorithm in Java
Discuss with a partner
How might we draw a Scribble recursively?
(A collection of lines drawn when the user onMouseDragged)

When should we choose loops (iteration) over recursion?

Learning Goals
By the end of this class, students should be able to:
1. Describe the 3-step process for recursion
2. Explain why recursion is a useful approach for some problems
   We’ll be discussing recursion again!