Syllabus

Intro to CS: Objects, Events, and Graphics

Instructors

Prof. Andrea Danyluk	Prof. Iris Howley
TCL 305	TCL 308
597-2178	597-4633
andrea@cs.williams.edu	iris@cs.williams.edu
Office Hours TBA	TBA

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:

Bruce, Danyluk and Murtagh, *Java: An Eventful Approach*, Prentice-Hall, 2006.

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/`. 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.

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

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

CSCI 134

Intro to Computer Science
Objects, Events, Graphics

Iris Howley
iris@cs.williams.edu
Assistant Professor of Computer Science

Williams

Welcome to CSCI 134!

How might we draw a Scribble?

(A collection of lines drawn when the user `onMouseDrags`)

Take a moment to write down your response, individually.

Discuss with a partner!

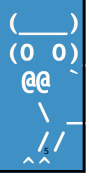
How might we draw a Scribble?

(A collection of lines drawn when the user `onMouseDrags`)

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

SpiralsScribblesRecursive

7

CONGRATS!

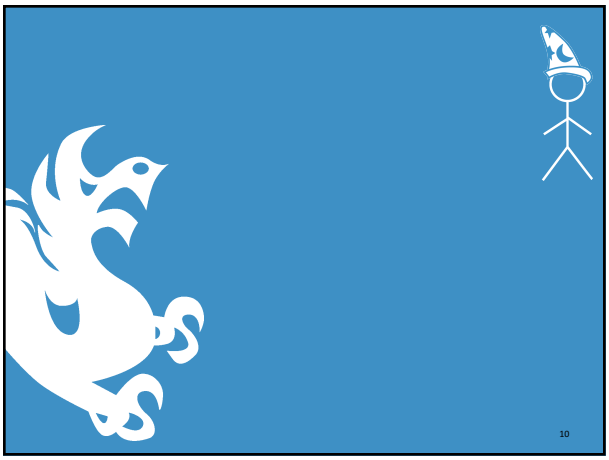
Halfway through the semester!!

8

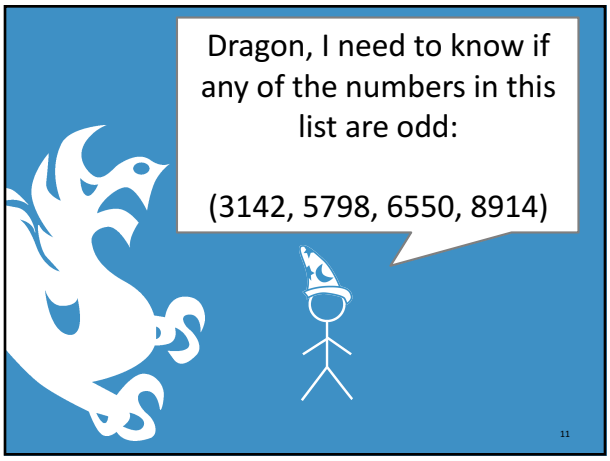
Administrative Details

- 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

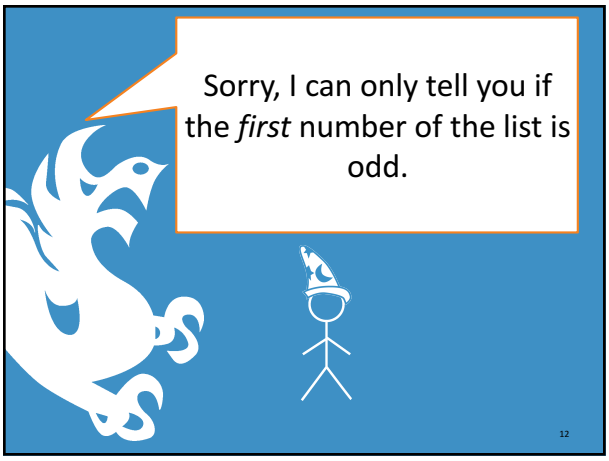
9



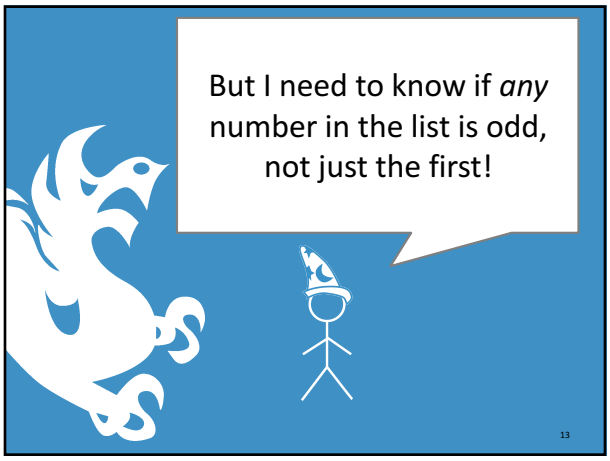
10



11

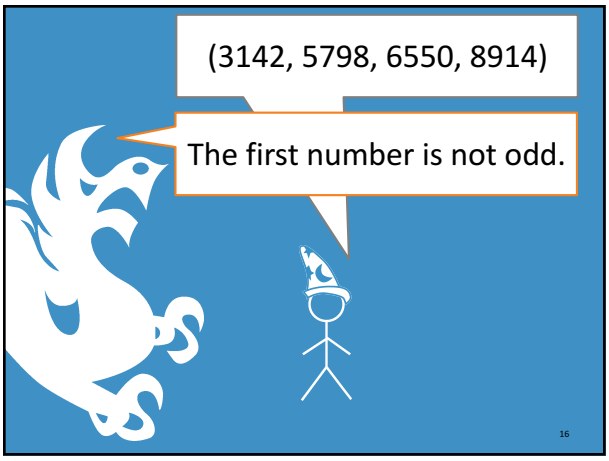


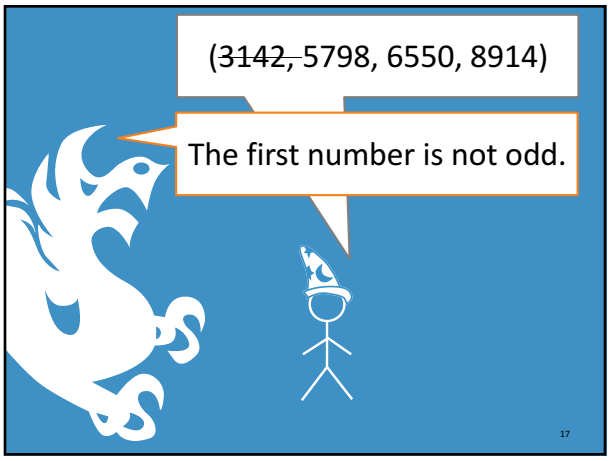
12

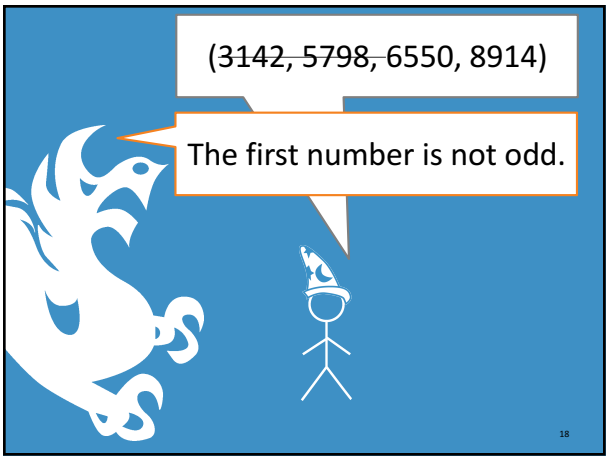

But I need to know if *any* number in the list is odd, not just the first!
13

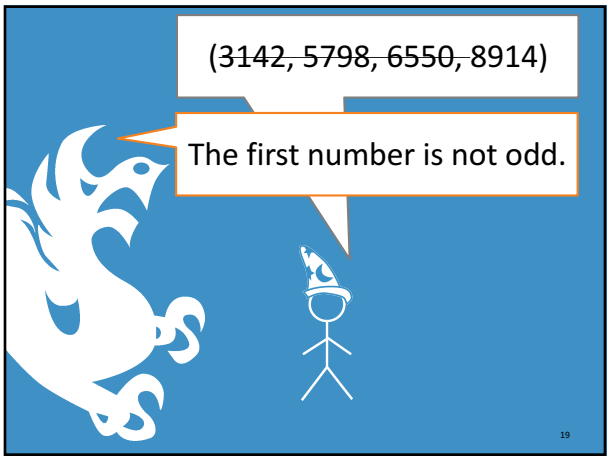

I'll only look at the first number, but I'll look at as many lists as you like.
14

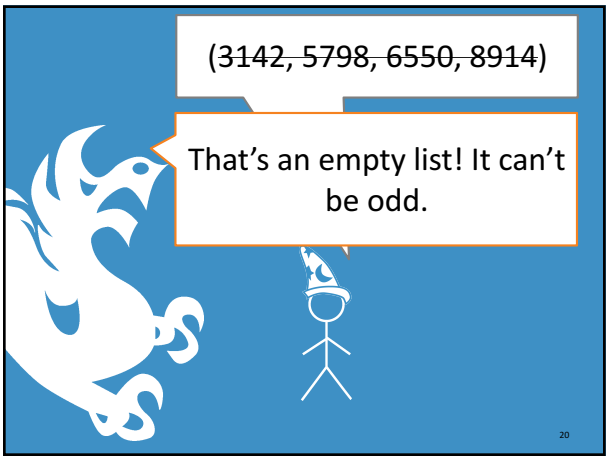

?
What should Sam do?
15

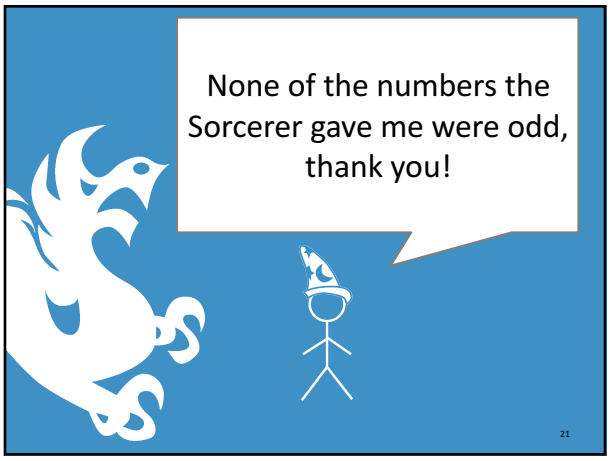

(3142, 5798, 6550, 8914)
The first number is not odd.
16

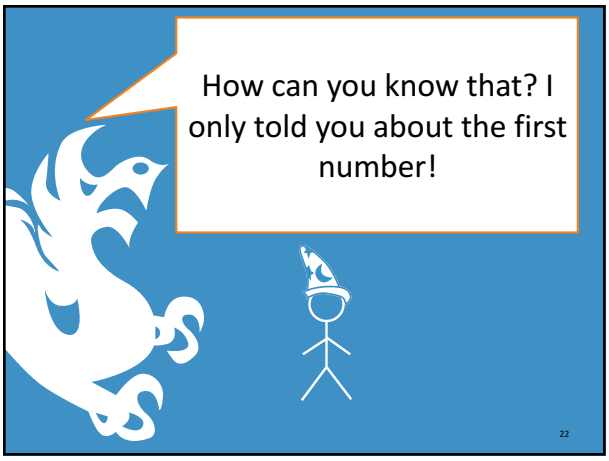

(~~3142,~~ 5798, 6550, 8914)
The first number is not odd.
17

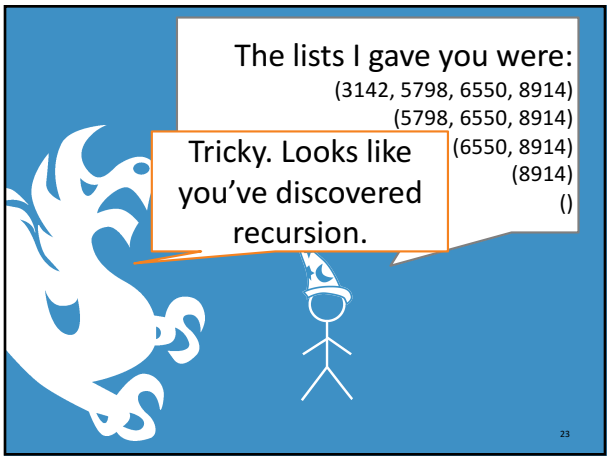

(~~3142, 5798,~~ 6550, 8914)
The first number is not odd.
18

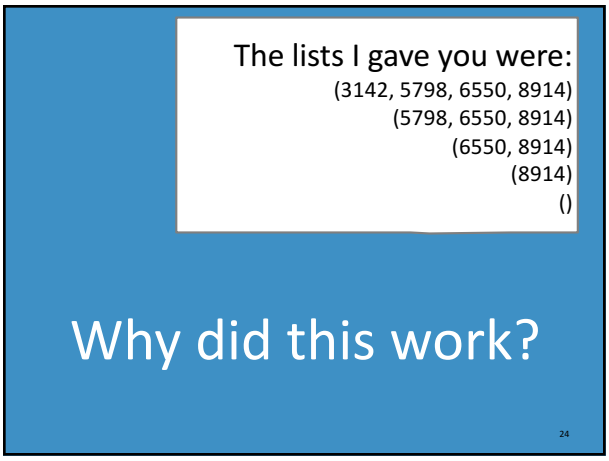

(3142, 5798, 6550, 8914)
The first number is not odd.
19

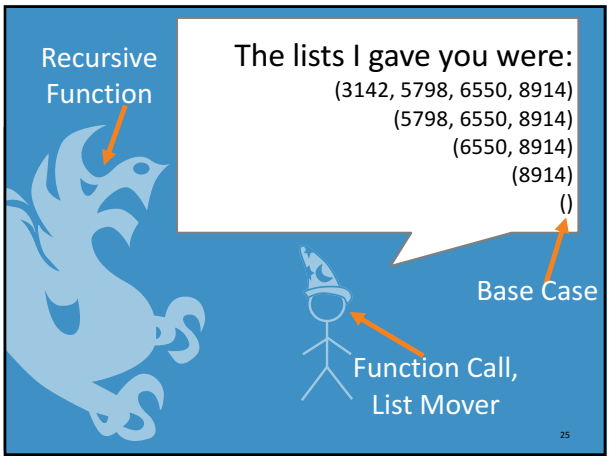

(3142, 5798, 6550, 8914)
That's an empty list! It can't be odd.
20


None of the numbers the Sorcerer gave me were odd, thank you!
21


How can you know that? I only told you about the first number!
22


The lists I gave you were:
(3142, 5798, 6550, 8914)
(5798, 6550, 8914)
(6550, 8914)
(8914)
()
Tricky. Looks like you've discovered recursion.
23


The lists I gave you were:
(3142, 5798, 6550, 8914)
(5798, 6550, 8914)
(6550, 8914)
(8914)
()
Why did this work?
24



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.

26

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

27

Pseudocode

```
Sam: The list is {3142, 5798, 6550, 8914}
Dragon: Is list empty?
Dragon: No? Is first number odd?
Dragon: Yes? Print.
Sam: Drop first num from list!
Dragon: Is list empty?
Dragon: No? Is first number odd?
Dragon: Yes? Print.
```

28

Pseudocode

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

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

29

Let's implement this recursive algorithm in Java

30

Discuss with a partner

How might we draw a Scribble **recursively**?

(A collection of lines drawn when the user `onMouseDrags`)

31

When should we choose loops (iteration) over recursion?

32

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!

33