Computer Science 210

Data Structures

Welcome to Data Structures!

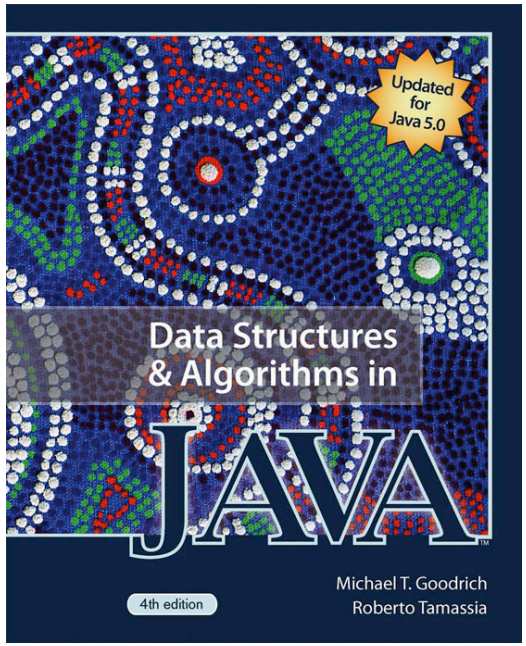
- Data structures are fundamental building blocks of algorithms and programs

- Csci 210 is a study of data structures
  - design
  - efficiency
  - implementation
  - use

- Prerequisites:
  - csci 101 (at Bowdoin or in high-school)
  - In other words
    - beginner knowledge of programming (in Java)
    - enjoy programming and problem solving

Logistics

- Instructor: Laura Toma
  - office: Searles 219
  - Office hours:
    - T, W, Th 4-5pm
    - by appointment
    - quick questions any time I am in the office
- TAs: Kristopher Koch, Drew Kantor, Yuna Oh
  - office hours: TBA
- Textbook:
  - Goodrich & Tamassia
  - online: Sedgewick & Wayne, Programming in Java
- Website:
  - http://www.bowdoin.edu/~ltoma/teaching/cs210/spring09/



Course outline

- Week 1: Java review.
- Week 2: Java graphics. Arrays.
- Week 3: Linked lists.
- Week 4: Recursion.
- Week 5: Program analysis.
- Week 6: Stacks and queues.
- Week 7: Searching and backtracking.
- -------- Exam 1
- Week 8: Vectors, array lists and iterators.
- Week 9: Trees and search trees.
- Week 10: Maps and hash tables.
- Week 11: Priority queues.
- Week 12: Sorting.
- Week 13: Graphs.
- -------- Exam 2

Work and grading policy

- Class work:
  - weekly lab assignments (approx. 45%)
  - 2 exams (approx 45%)
  - readings, class participation, in-class assignments (approx. 10%)
- The class is programming-intensive
- Lab assignments are not meant to be finished during lab time. You have one week to complete them.
  - Handing in: hard copy + email
  - on hard copy sign that you have followed class honor code
- Lab work: individual
- Late policy: 25% per day
  - Why? it is absolutely essential that you do not fall behind
  - failure to turn in a lab ==> fail the class
  - better turn in incomplete lab

Honor code

- Students are expected to follow the Bowdoin Computer Use Policy and the Academic Honor Code.
- You are encouraged to discuss ideas and techniques broadly with other class members, but not specifics of assigned problems except as part of group projects.
- Discussions should be limited to questions that can be asked and answered without using any written medium (e.g. pencil and paper or email).
- This means that at no time should a student read any code written by another student unless they are part of the same group.
- Sharing of code or intermediate designs is expressly prohibited.
- The same rules apply once you have finished the course: sharing your code with other students will be considered a violation of Bowdoin's honor code.
- Violation of this policy is grounds for me to initiate an action that would be filed with the Dean's office and would come before the J Board.
- If you have any questions about this policy, PLEASE do not hesitate to contact me. This will be a zero-tolerance policy.

  Do not leave lab for the last night.

  If not working, submit what you have.

More about the class

- The class is about designing, analyzing, implementing and using fundamental data structures.

- 101
  - you learnt how to use the basic constructs in Java. Put it differently, you learnt how to use a hammer and saw.
  - focus was on learning the tools available when writing a program
    - syntax, conditionals, loops, arrays, etc

- 210
  - Knowing how to use a hammer and saw does not mean you can build a house. In 210 you'll learn how to build a house.
  - you'll learn more tools, but most importantly
    - you'll learn to put them together to create a large program

More about the class

- It is programming intensive
- however ... is NOT about programming
  - but about program development
    - design + analysis + programming + debugging
- Programming language: Java. Why?
  - makes graphics and web applications easy
  - available on all platforms
  - new language, in fashion
- Most of the class will be independent of Java
  - maybe next semester ...Python?
  - You'll learn to distinguish between Java questions (check the Java doc pages to answer), and language-independent questions
- Java graphics NOT the core of the class

Labs

- Posted online ahead of time. Better to read before coming to lab.
- The labs are not meant to be finished during lab time
  - due one week after they are assigned
  - they are your homework
- The lab time is for you to understand the lab, plan your solution, get started.
- Labs are not all equal
  - generally speaking, progressively harder
- The labs are not always connected to the topic studied in class that week
- The labs are often harder than they look. You'll spend a lot of time understanding what the task is. It is a good idea to read the lab beforehand, so that you can ask many questions during lab time.
- Labs are challenging and fun. They are the most important learning tool
  - you will learn in class
  - you will REALLY learn while working out the labs
- At times the process will seem painful, and occasionally you will find a lab unfair.
- However, at the end of the class you'll find that you've learned a lot.

Expectations

- TOGETHER
  - During class time we'll talk about data structures concepts, we'll analyze various options and we'll work out the implementation details for some of these options
  - Often during class-time we'll program together as a group
  - Occasionally there'll be in-class assignments and team work

- YOU
  - you'll learn to think like a computer scientist
  - you'll learn to find out what it takes to get a task accomplished
  - you'll start your lab in a timely manner
  - the bulk of your effort will be to get the lab assignments to work
  - you'll get used to: develop flowchart, develop incrementally, debug, test
  - you need to develop your code so that it can be debugged!

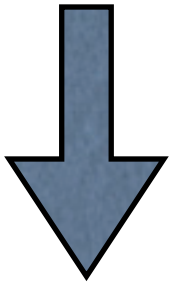
Scenario

- You develop all classes at once. Nothing works! HELP!!!
  - If code has too many errors, their combinations are infinite ==> impossible to debug
  - MORAL: you structure your code so that you implement one feature at a time, you debug and test it, and then go on.

- You get stuck in Java graphics (GUI) before solving the actual problem.
  - Why don't the buttons show?
  - MORAL: Solve the core of the problem first, with a simple interface! If you have time at the end you can make your GUI more fancy.

More Expectations

- Problem: various backgrounds
  - 101 A vs. 101B
  - 101 vs. highschool
  - highschool 1 vs. highschool 2



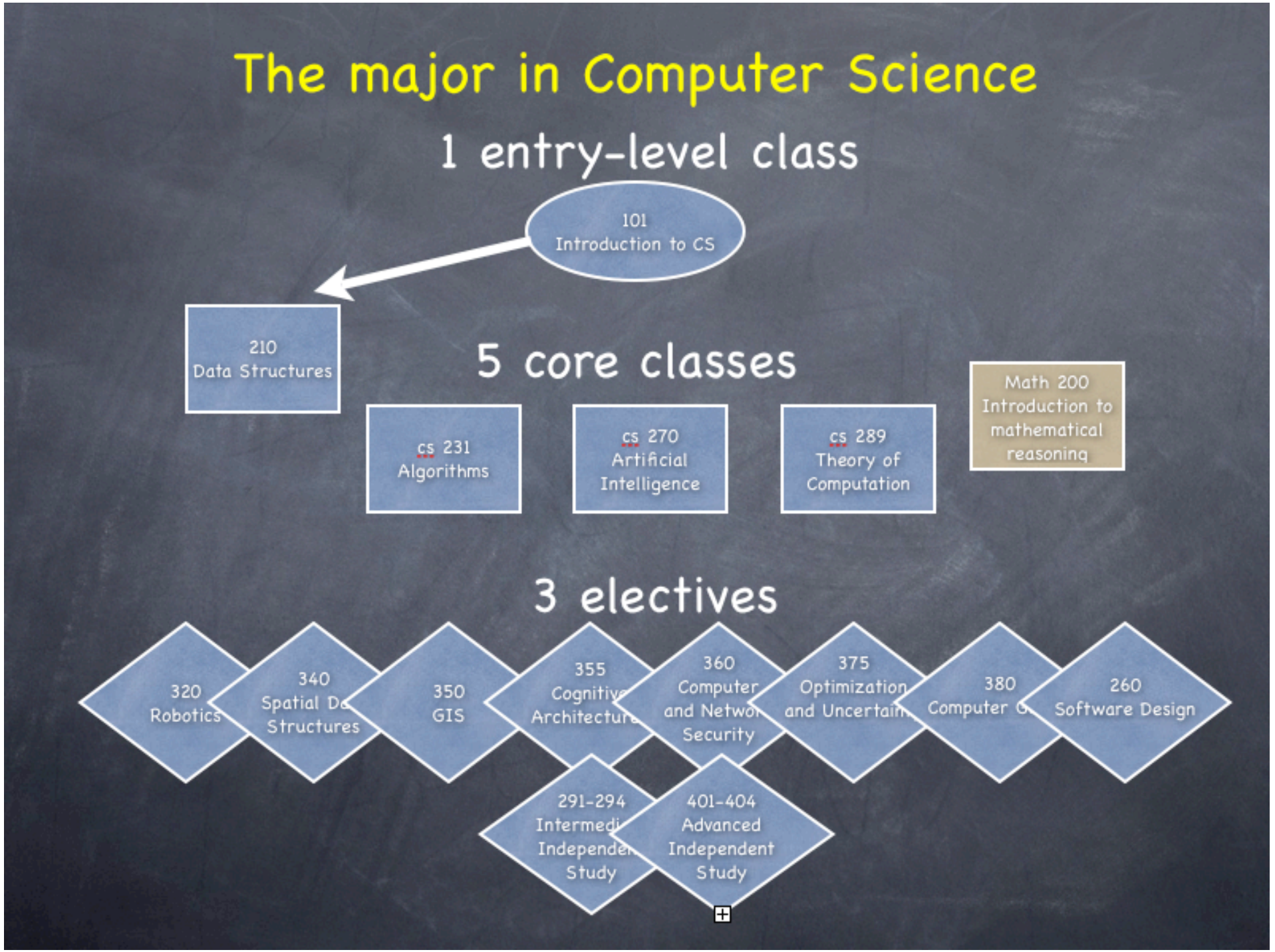
- Willingness to work in a group environment
- Patience with material that is not new and when class is slow
  - participate
  - share with others
- Ask plenty of questions when something is unclear
- Goal: we want to work as a class

Class Outcomes

- You will learn the fundamental data structures:
  - lists, vectors stacks, queues, priority queues, trees, hash tables and maps
- Design: you will learn to model and come up with a solution to a problem
  - modularity, data abstraction, building blocks
- Analysis: you will learn to analyze the efficiency of your solution
  - you will learn to use efficiency considerations to decide the choice of data structures
- Program development: you will learn the importance of each step in getting a program to work: design, debug, test
  - Practice of programming:
    - Simplicity
    - clarity
    - generality
- You'll learn to think like a computer scientist.
- You'll learn to find out what it takes to get a task accomplished

This being said...

- Yes, 210 will be challenging.
- But, most of the people who take 210
  - like it
  - say it is one of the most fun classes they took
  - continue with Computer Science
    - 210 is the pre-requisite for all other classes
  - If you like 210, you should think about majoring or minoring in computer science
- You are all here because you liked 101 and programming.
- Welcome, and have fun!


The major in Computer Science
1 entry-level class
101
Introduction to CS
210
Data Structures
5 core classes
cs 231
Algorithms
cs 270
Artificial
Intelligence
cs 289
Theory of
Computation
Math 200
Introduction to
mathematical
reasoning
3 electives
320
Robotics
340
Spatial Da
Structures
350
GIS
355
Cognitive
Architecture
360
Computer
and Networ
Security
375
Optimization
and Uncertain
380
Computer G
260
Software Design
291-294
Intermedi
Independen
Study
401-404
Advanced
Independent
Study

Java programming review

GT chapter 1

- Base types
  - `boolean, char, byte, short, int, long, float, double`
- `Class`
  - a type; a cookie cutter; blueprint from which individual objects are created
  - A class does not actually exist; it is just a "pattern"
  - A class contains data and methods
- Object
  - an instance of a class; the actual cookie
  - instance variables
  - creating an object
- Methods
  - Declaring methods; parameters, return types
  - Constructor methods; main method
- Expressions
  - operators, the dot operator, casting
- Statements
  - `if, switch, loops, return, break, continue`
- Arrays

Base types

- boolean
  - true or false
- char
  - 16 bit character
- byte
  - 8-bit signed integer
- short
  - 16-bit signed integer
- int
  - 32-bit signed integer
- long
  - 64-bit signed integer
- float
  - 32-bit floating point number
- double
  - 64-bit floating point number

Declaring

- variables
  - `<type> <varriable-name>;`
- constants
  - static final int MONDAY = 0;
- classes

```
[abstract| public|final]  class <class-name> extends <super-
classname>   implements <interface_1> <interface_2>.....  {
//instance variables
//methods
}
```

- abstract class
  - class has (some) abstract methods (later)
- final class
  - can have no subclasses
- public class: class can be instantiated and extended by by anything in the same package or by anything that imports the class

Declaring objects

```
//class definition
class Gnome { ...};
//declares an object g of type Gnome
Gnome g;
//object g does not yet exist; to create an object call new
g = new Gnome(...);
```

- Constructor
  - a special method that is used to create objects
  - the constructor allocates memory to hold the object and returns a reference to this memory ; this address is then stored in the object variable (g)
- Number objects
  - we sometimes want to store integers ar objects
  - x = new Integer(10);
    - an object that represents integer 10

Instance variables

- represent the data associated with the object
- scope
  - public
    - anyone can access public instance variables
  - private
    - only methods of the same class (not subclass) can access private vars
  - protected
    - only methods of the same package and subclasses can access protected vars
- static
  - a static variable is associated with the class
  - used to store global information about the class
- final
  - a constant
  - must have an initial value, which cannot be changed

Methods

- Method: code that can be called on a particular object
- Declaring methods
  - parameters
  - method modifiers:
    - public , protected , private. abstract , final , static
  - return values and types
- constructor methods
  - a special kind of method that is used to initialize newly created objects
- main method
  - needed in classes that are meant to define stand-alone programs
- java Gnome
  - Java-system invokes the main method in class Gnome
  - main must be public and static

Operators

- assignment
    - a = b;
- dot
    - obj.methodname(...)
- arithmetic
    - +, -, *, /, %
    - ++, --
- logical operators
    - <, <=, >, >= , ==, !=
- operators on booleans: ! , &&, ||
- bitwise operators

- casting
- if statements
- break
- continue
- switch
- for loops

```
for (initialization; condition; increment)
      body
```

- while loops

- Output
  - `System.out.print()`
- Input

- Writing a java program
  - design
  - coding
  - readability and style
  - testing and debugging
  -
- For next time:
  - read GT: Chapter 1