Supplementary Readings:
(a.) The Top 10 Programming Languages, IEEE Spectrum, October 2011.
(b.) java packages, en.wikipedia.org/wiki/Java_package
(c.) Data Structures, en.wikipedia.org/wiki/Data_structure
(d.) Computer Science, en.wikipedia.org/wiki/Computer_science
(e.) Problem Solving, en.wikipedia.org/wiki/Problem_solving
* Summarize
  * Key points in this course
  * Big-picture

* Reflect
  * Celebrate Successes
  * Challenges & Opportunities

* Look ahead
  * Computer Science Courses
  * Other opportunities
* Abstract Data Types: a fundamental Software Architecture
  * Interface - What operations are available to public?
  * Implementation - How operations are carried out?
  * Java language: Classes, Generic data types, Arrays, Linked Chains, ...

* Collection ADTs for Data Structures
  * Linear: Queues, (DeQues, Priority Qs), Stacks, Lists, (Sorted Lists)
  * Associative: Dictionaries, Hash Tables
  * Non-linear: Trees, Binary Search Trees
  * Compositions, e.g., a list of dictionaries (HW 5), an array of queues (Radix sort)

* Algorithms
  * Sorting Collections: Selection, Merge and Radix Sort
  * Searching Collections: Linear, Binary, and Hash Search
  * Evaluation: Asymptotic complexity, Big-Oh
Summarize: Big-Picture

* Structure of Computer Programming (Csci 1902)
  * Algorithms + Data Structures + Architecture

* Programming Competencies and Ideas
  * (Computer) Science
    * Objective, reproducible, generalizable ideas, e.g., big-Oh
    * **Ultimate limits**: What is the complexity of sorting? What is not computable?
    * What best possible algorithms for sorting?
  * (Software/Computer) **Engineering**:
    * Apply science to practical use and application
    * Example: IBM Watson, Facebook, Google
    * Q? What can be done today? How may it be done economically?
  * (Software/User Interface) **Design & Architecture**:
    * To plan and fashion artistically the form and structure of
      * an object, artwork, decorative scheme
    * Example: Abstract Data Type, Object-Oriented
    * Example: Aesthetic User Interface design
* Teamwork
  * We made friends and bonded with common programming war stories!
  * We met future colleagues to grow into careers together!
  * We met potential mentors

* Java, the most popular programming language!
  * Eclipse, JUnit, Ant, Debugger
  * Read, write, test, and debug multi-class Java programs.

* Fundamentals, which will help in Csci courses and beyond!
  * Collections (e.g., stack, hash-table), which are used widely!
  * State-of-the-art Searching and sorting algorithms
  * Asymptotic complexity to compare algorithms!
  * Revisit Homework 0 (last question) to assess our learning

* Problem solving
  * An important skill values in many careers and organizations
  * Ex. Interviews at Google and Amazon
Challenge: Large Variation in Java Skills

* Java is not a pre-requisite. Java is very different from Scheme.
* Many completed Csci 1103 (or AP course), many did not
* We provided Java skill-building opportunities in early weeks.
* Q? Are more Java activities needed, e.g., Java for “Scheme”-rs?

Challenge: Large Variation in Problem Solving Skills

* We introduced problem solving in Lecture, Home-works and Quizzes!
* A shift from previous semester, which emphasized programming
* Q? Is more (e.g. composition of collections) needed in homework?
* Q? Should there be practice quizzes or practice examinations?

Challenge: ...

Q? What should we give up to make room for new activities?
* Explore Csci 1902 topics in more depth
* Use what we learned in Csci 1902
Java is the most popular programming language!
It opens door for internships, jobs, careers!
Source: IEEE Spectrum, October 2011, (The Top 10 Programming Languages, R. S. King)
* javax.swing - Web-based User Interface
* java.awt - GUI components, e.g., checkboxes, radio-buttons, menus
* javax.applet -
* java.sql: Java database connectivity (jdbc) to access databases
* java.net - networking operations, sockets, DNS lookup, ...
* Java.security - key generation, encryption, decryption
* ...
* Smartphones - Android
  * Ex. Hello World (5-minute videos from CodePicture):
    * Part I: www.youtube.com/watch?v=Ltsg2LcCL1E
    * Part II: www.youtube.com/watch?v=_s8aZlpubso
  * Ex. developer.android.com/resources/tutorials/hello-world.html
Short Title: From GPS and Google Earth to Spatial Computing

Tu., Th. 4pm - 5:15pm in 3-125 Keller Hall; 24611

Registration: Ask front desk, Computer Science Department, 4-192 Keller Hall

Long Title: Spatial Computing: How are GPS, Location-based Social Networks, Geo-Social Media and Cell-phone based Location Based Services transforming computing?

Hands on laboratories: Android (GPS), Google Maps & Earth (KML), SQL / OGIS, location API from Twitter/HTML5, Arc/GIS, etc.

Topics: Spatial ADTS (e.g., reference frames, points, line-strings, polygons, road-maps), data-structures (e.g., R-tree, Vornoi diagram), algorithms (e.g., routing), models (e.g., raster, vector), mathematics (e.g., topology, geometry, graphs), data analytics (e.g., spatial queries, geo-statistics), Geo-visualization (e.g., cartography).
* Read Textbook with active learning
  * Self-test questions - solve those before looking at the answer!
  * Examples - Write your own answers before reviewing one by author!
  * Exercise at end of chapter
    * Solve yourself, then compare your answer with that of your study partner

* Examples of True/False, Multiple Choice, Matching
  * Ex. 23-2. What is the height of the shortest binary tree with 21 nodes? Is it full?
  * Ex. 23-8: Classify trees in Figure 23-23 into
    (a) binary search tree, (b) maxheap, (c.) other
  * Ex. 23-9: Draw shortest possible binary search tree with a, b, c, d, e, j1, j2, k, l, p, r, s
  * Ex. 23-12: Can a binary search tree ever be a maxheap? Explain.

* Read and Understand Code - Example
  * Chapter. 23. Given code (on next slide), recognize one of the four traversals of a binary tree?

* Write Code Fragments - Examples
  * Chapters 11, 23 :. Implement a priority queue using a maxheap binary tree.
  * Ch. 11, 21, 22: Implement hash-table (separate chaining) using an array of Queues

* Design simple algorithms and Write Code - Example
  * Ex. 23-4: Write a recursive algorithm to count the nodes in a binary tree.
* Read and Understand Code - Example -

* Q? Which traversal (of a binary tree) is implemented by aTraversal() method?

```java
public void aTraversal(TreeNode root) {
    Queue queue = new LinkedList();
    TreeNode flagNode = new TreeNode("");
    queue.add(root);
    queue.add(flagNode);
    while (!queue.isEmpty()) {
        TreeNode node = queue.dequeue();
        if (node != flagNode) {
            System.out.print(node.val + " ");
            if (node.left != null) queue.add(node.left);
            if (node.right != null) queue.add(node.right);
        } else if (!queue.isEmpty()) queue.add(flagNode);
    }
}
```

* Q? Which traversal will result if Queue is replaced by Stack in above code?
  * Dequeue() is replaced by pop() and add() is replaced by push().
Thank you.

It was a lot fun working together.

Hope to see you in

- Csci 5980 Spatial Computing (Fall 2012)
- Database Courses: Csci 4707, 5707, 5708, 8715

- Shashi Shekhar