CSCI 136
Data Structures & Advanced Programming

Fall 2019

Instructors

Bill Lenhart & Samuel McCauley

Administrative Details

- Class roster: Who's here?
  - And who's trying to get in?
- Handout: Class syllabus
- Lecture location: Schow 030a
- Lab: Wed 12-2 or 2-4 (go to assigned lab!)
- Lab location: TCL 217a (Lenhart) & 216 (McCauley)
- Lab entry code: I hope you memorized it in classs!
- Course Webpage:

  http://cs.williams.edu/~cs136/index.html

Today's Outline

- Course Preview
- Course Bureaucracy
- Java (re)fresher–Part 1

Why Take CS136?

- To learn about:
  - Data Structures
    - Effective ways to store and manipulate data
  - Advanced Programming
    - Use structures and techniques to write programs that solve interesting and important problems
  - Basics of Algorithm Analysis
    - Measuring algorithm complexity
    - Establishing algorithm correctness

Squad* Goals

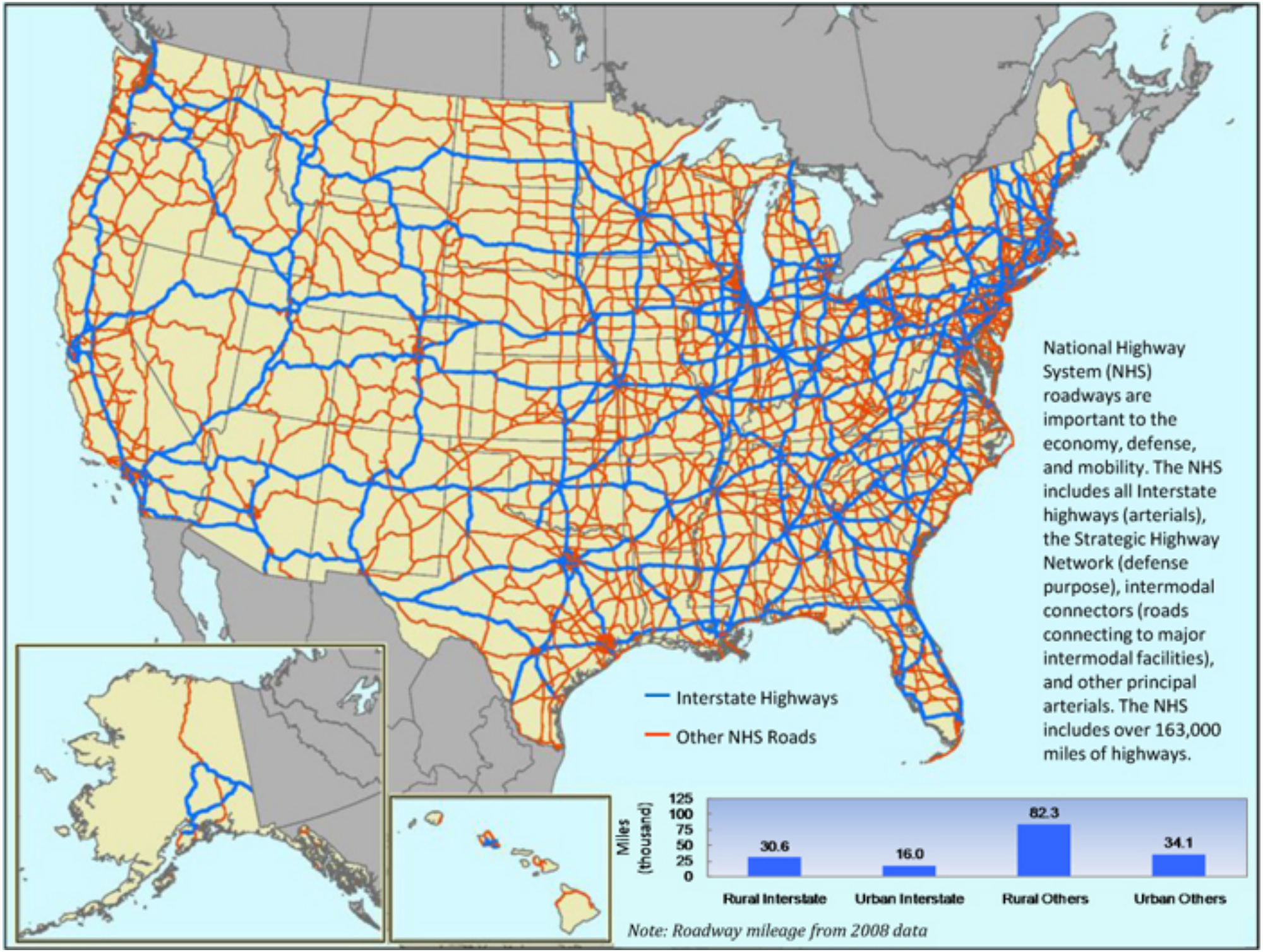
- Identify basic data structures
  - list, stack, array, tree, graph, hash table, and more
- Implement these structures in Java
- Learn how to evaluate and visualize data structures
  - Linked lists and arrays both represent lists of items
  - Different representations of data
  - Different algorithms for manipulating/accessing/storing data
- Learn how to design larger programs that are easier to modify, extend, and debug
- **Have fun!**

*Bill L has a teenage daughter....

Common Themes

1. Identify data for problem
2. Identify questions to answer about data
3. Design data structures and algorithms to answer questions *correctly* and *efficiently* (Note: not all correct solutions are efficient, and vice versa!)
4. Implement solutions that are robust, adaptable, and reusable

Example: Shortest Paths in Networks


Interstate Highways
Other NHS Roads
National Highway System (NHS) roadways are important to the economy, defense, and mobility. The NHS includes all Interstate highways (arterials), the Strategic Highway Network (defense purpose), intermodal connectors (roads connecting to major intermodal facilities), and other principal arterials. The NHS includes over 163,000 miles of highways.
Miles (thousand)
125
100
75
50
25
0
30.6
16.0
82.3
34.1
Rural Interstate
Urban Interstate
Rural Others
Urban Others
Note: Roadway mileage from 2008 data

Finding Shortest Paths

- The data: road segments
  - Road segment: Source, destination, length (weight)
- The question
  - Given source and destination, compute the shortest path from source
- The algorithm: Dijkstra's Algorithm
- The data structures (spoiler alert!)
  - Graph: holds the road network in some useful form
  - Priority Queue: holds not-yet-inspected edges
  - Also uses: Lists, arrays, stacks, ...
- A quick demo….

Course Outline

- Java review
- Basic structures
  - Lists, vectors, queues, stacks
- Advanced structures
  - Graphs, heaps, trees, dictionaries
- Foundations (throughout semester)
  - Vocabulary
  - Analysis tools
  - Recursion & Induction
  - Methodology

Syllabus Highlights

- How to contact us
  - Bill Lenhart (TPL 304)
    - Office hours: TBA
    - mailto:wlenhart@williams.edu
  - Sam McCauley (TCL 209)
    - Office hours: TBA
    - mailto:sam@cs.williams.edu
- Textbook
  - Java Structures: Data Structures in Java for the Principled Programmer, √7 Edition (by Duane Bailey)
  - Take one: You're already paying for it!
- Weekly labs, problem sets, mid-term & final exams....

Honor Code and Ethics

- College Honor Code and Computer Ethics guidelines can be found here:
  - https://sites.williams.edu/honor-system/
  - https://oit.williams.edu/policies/ethics/
- You should also know the CS Department computer usage policy.
  - https://csci.williams.edu/the-cs-honor-code-and-computer-usage-policy/
  - If you are not familiar with these items, please review them.
- We take these things very seriously…

Your Responsibilities

- Come to lab and lecture on time
- Read assigned material before class and lab
  - Bring textbook to lab (or be prepared to use PDF)
  - Bring paper/pen(cil) to lab for brain-storming, …
- **Come to lab prepared**
  - Bring design docs for program
  - 1 Prof + 1TA == help for you: take advantage of this
- Do NOT accept (prolonged) confusion! Ask questions
- Your work should be your own. Unsure? Ask!
- Participate

Accounts and Passwords

- Mandatory: Before the first lab
  - Talk to Lida Doret about your CS account
- Lida manages our systems. She will be available
  - Today: 9/6: 10:00-10:45 am, 12:00-12:45 pm
  - Mon., 9/9: 10:00-10:45 am
  - Tues., 9/10: 11:00-11:45 am

- Her office is on the second floor (TCL 205)
- Get this sorted out before lab on Wednesday!

Why Java?

- There are lots of programming languages…
  - C, Pascal, C++, Java, C#, Python
- Java was designed in 1990s to support Internet programming
- Why Java?
  - It's easier (than predecessors like C++) to write correct programs
  - Object-oriented – good for large systems
  - Good support for abstraction, extension, modularization
  - Automatically handles low-level memory management
  - Very portable

How we will code

- Command-line tools
- Atom: A modern, easy-to-use code editor

- Set up already on lab computers
- Start now if you want to code at home
  - Ask us for help if you need it, especially with Windows

Java Over/Review (Crash Course)

Java

```
/*
 * This program prints a message.
 */

public class Hello {
      // Print a message.

      public static void main(String[] args) {

            System.out.println("Hello CS136!");

      }

}
```

Java

Edit/Compile/Run cycle

- Edit: Save Java source code in file `Hello.java`
- Compile: `javac Hello.java`
  - Produces Java *bytecode* file named `Hello.class`
- Execute: `java Hello`
  - Searches `Hello.class` for a method with *signature*

```
public static void main(String[])
```

  - Executes that method (if it exists)

Java

Notes

- Multi- and single-line comments
- Code is wrapped in a *class declaration*
  - Everything is (in) a class in Java
  - File name should be same as declared class name
  - `System` is a Java class holding an object called out
  - `out` is of class type `PrintStream`
- The parameter `args` is an array of `String`
  - Passed to the main method from the *command line*
  - Contains every string on the command line after `java Hello`
- This allows passing values into program
- Can replace args with any other variable name…

Java

```
/* This program prints words. */

public class Hello2 {

      public static void main(String[] CLParams) {

            for(int i = 0; i < CLParams.length; i++) {

                  System.out.println( CLParams[i] );

            }

      }

}
```

Java

Notes

- Changed `args` to `CLParams`
- Every array stores its size: CLParams.length
  - It's a data member, not a method call
- Java `for` loop

```
for(initialization; continuation; update)
        { statement ; … statement ; }
```

- Equivalent to Java `while` loop

```
initialization;
while ( continuation ) {
        statement ; … statement ;
        update;
}
```

Java

```
/* This program prints words. */

public class Hello3 {

      public static void main(String[] CLParams) {

            int i = 0;
            while( i < CLParams.length ) {

                  System.out.println( CLParams[i] );
                  i++;

            }
      }
}
```

Java

```
/* This program prints words. */
public class Hello4 {

   public static void main(String[] CLParams) {

      if(CLParams.length == 0) {
         System.out.println("Hello CS136!");
      }

      else {
         for(int i = 0; i < CLParams.length; i++) {
            System.out.println( CLParams[i] );
         }
      }
   }
}
```

Java

```
/* This program prints words.
 * {} can be omitted for single-statement blocks
 */
public class Hello5 {
   public static void main(String[] CLParams) {

      if(CLParams.length == 0)

         System.out.println("Hello CS136!")

      else
         for(int i = 0; i < CLParams.length; i++)
            System.out.println( CLParams[i] );

   }
}
```

Primitive Types

- Provide numeric, character, and logical values
  - 11, -23, 4.21, 'c', false
- Can be associated with a name (*variable*)
- Variables *must* be declared before use

```
int age;      // A simple integer value
float speed; // A number with a 'decimal' part
char grade;  // A single character
bool loggedIn; // Either true or false
```

- Variables *can* be initialized when declared

```
int age = 21;
float speed = 47.25;
char grade = 'A';
bool loggedIn = true;
```

Array Types

- Holds a collection of values of some type
- Can be of any type

```
int[] ages;         // An array of integeras
float[] speeds;     // An array of floats
char[] grades;      // An array of characters
bool[] loggedIn;    // Either true or false
```

- Arrays can be initialized when declared

```
int[] ages = { 21, 20, 19, 19, 20 };
float[] speeds = { 47.25, 3.4, -2.13, 0.0 };
char[] grades = { 'A', 'B', 'C', 'D'  };
bool[] loggedIn = { true, true, false, true };
```

- Or just created with a standard default value

```
int[] ages = new int[15]; // array of 15 0s
```

Sum 1

```
class Sum1 {

   public static void main(String[] args) {

      if ( args.length < 2 )
         System.out.println( "Syntax: java Sum3 num1 num2" );

      else {
         int n0 = Integer.valueOf( args[0] );
         int n1 = Integer.valueOf( args[1] );
         System.out.println(n0 + " + " + n1 + " = " + (n0 + n1));
      }
   }
}
```

Sum 2

```
class Sum2 {

   public static void main(String[] args) {

      if ( args.length == 0 )
         System.out.println( 0 );

      else {
         int total = 0;
         for ( int i = 0; i < args.length; i++ )
            total = total + Integer.valueOf( args[i] );

         System.out.println( "The sum equals " + total );
      }
   }
}
```

Sum 3

```
class Sum3 {

    public static void main(String[] args) {

        if ( args.length == 0 )
            System.out.println( 0 );

        else {
            int total = 0;

            // 'for-each' version of for loop
            for ( String num : args )
                total = total + Integer.valueOf( num );

            System.out.println( "The sum equals " + total );
        }
    }
}
```

Sum 4

```
class Sum4 {

    // Create a new Scanner, read two integers, print their sum.

    public static void main(String[] args) {

        // create a new scanner for the terminal input
        Scanner in = new Scanner(System.in);

        System.out.print("Give me a number: ");
        int n1 = in.nextInt();
        System.out.print("Give me another number: ");
        int n2 = in.nextInt();

        System.out.println( n1 + " + " + n2 + " = " + (n1 + n2));
    }
}
```

Sum 5

```
class Sum5 {

    // Create a Scanner, read in integers, and print their sum.
    public static void main(String[] args) {

        // create a scanner for the terminal input
        Scanner in = new Scanner(System.in);

        int total = 0;   // running sum

        System.out.print("Give me a number (ctrl-d to end): ");
        while (in.hasNext()){
            int n = in.nextInt();
            total += n;
        }

        System.out.println("\nThe total is " + total);
    }
}
```

Sample Programs

- `Sum1.java … Sum5.java`
  - Programs that adds integers
- Of Note:
  - System.in is of type InputStream
  - Scanner class provides parsing of text streams (terminal input, files, Strings, etc)
  - Integer.valueOf(...) converts String to int
  - Static values/methods: in, out, valueOf, main

Summary

Basic Java elements so far

- Primitive and array types
- Variable declaration and assignment
- Some control structures
  - for, for-each, while, do-while

Some basic unix commands

- Edit (Atom), Compile (javac), run (java) cycle

Next time…

More Java and Object-oriented programming