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 30a
- Lab: Wed 12-2 or 2-4 (go to assigned lab!)
- Lab location: TCL 217a (Lenhart) & 216 (McCauley)
- Lab entry code: ****** (memorize now!) Course Webpage:

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

Today’s Outline

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

Purpose of the Class

- More (and better) programming!
  - Learn Java

- Data structures: how to use them and how they work

- Fundamentals of analyzing performance

What is a Data Structure?

- Stores data
- Supports operations that query the data

Example: Dictionary. What operations does it support? How does it work?

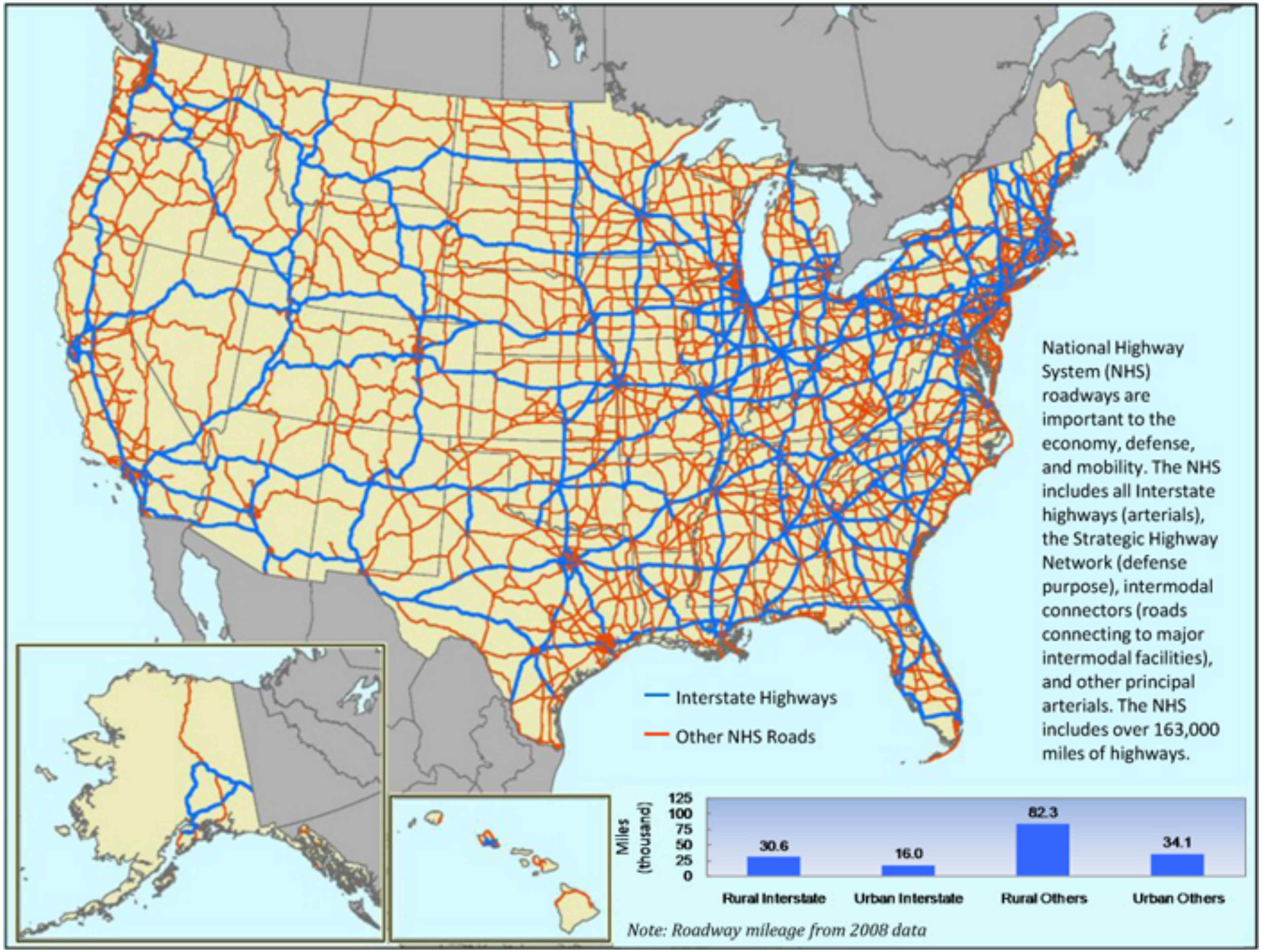
The Importance of Data Structures

- Allow you to design large, efficient programs that are easier to modify, extend, and debug

The Importance of Data Structures

- Allow you to design large, efficient programs that are easier to modify, extend, and debug

- Let's say I want to find the shortest route between two cities in the US. What do I need to store? What kind of data structures would I need?



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.

Note: Roadway mileage from 2008 data

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: T 2:30-4:00pm, R 1:00-4:00pm, F 1:00-2:00pm
    - mailto:wlenhart@williams.edu
  - Sam McCauley (TCL 209)
    - Office hours: M 3:00-5:00pm; F 12:00-2:00pm
    - 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.

Academic Dishonesty

- Taken very seriously at Williams and in this class
- In lab we will give examples to make our rules clear
- *Ask me if you don't know what's right!*

Academic Dishonesty

- Exam problems may not ever be discussed
- Assignments:
  - All code (etc.) must be written *only* by you (or your partner when applicable)
  - Always *understand* what you submit
  - You may brainstorm high-level ideas (IF you cite!)
  - May have another student help with syntax
  - May use code from the textbook and course materials, but not from the internet
  - Do not google the answers to assignment questions

Academic Dishonesty

The department's policy here: https://csci.williams.edu/the-cs-honor-code-and-computer-usage-policy/

Reading, Lectures, and Labs

- We expect you to attend/read all three
- Materials reinforce each other; all are on-topic for assignments and exams

Reading, Lectures, and Labs

- Lab attendance is **mandatory**
  - I will take attendance
  - Email me if there is an issue

- Bring your textbook and pen/paper to lab
- Course attendance and readings are required, but I will not generally be taking attendance

Labs

- Main assignments in the course (there will also be three problem sets and two exams)
- Start Wednesday during lab; due Sunday at 10PM

- Three late days during semester
  - Need to tell me if you're taking a late day!
  - Leave a note for the graders

Getting Help

- We are here to help you!
  - Seriously.
- Important: students in this course have very different backgrounds

Getting Help

- We are here to help you!
- Important: students in this course have very different backgrounds

TA Hours: M-Th evening, Sat 1-6, Sun 1-10

My Office hours: M 3-5pm, F 12-2pm TCL 209

These are available on the course calendar

Getting Help

- I'm always* available for quick questions via email
    - *may not answer right away depending
- I will only help with code in person
- 24 hour policy: I may take 24 hours to think before answering course policy questions

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 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
• Monday, 9/9:	10:00-10:45 am
• Tuesday, 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?

- Why Java?
  - Popular
  - 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

- Ability to learn new programming languages is essential

How we will code

- Command-line tools (unix)
- Atom: A modern, easy-to-use code editor
- Git (version control system)

- 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

```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 I

```
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