CS136: Data Structures & Advanced Programming

Spring 2020
Williams College

Administrative Details

- Lab 1 is online
  - Complete Pre-lab Step 0 by 4pm today
  - Getting to Know you form
  - Check that you can login to your CS account — else see Mary Bailey from 2-4pm
- TA hours start on Wednesday: see the TA and Office Hours calendar on the course webpage
- Reminder: no class Friday in celebration of ~~Valentine's day~~ Winter Carnival

Last Time

Course overview, syllabus details

Essential Unix commands so far:

- Compile
  - `$ javac File.java`
- Run
  - `$ java File`
- Directory/file system navigation
  - `$ ls`
  - `$ cd new_directory`

Today

We'll write some code

- Hello World!
- Sum two numbers
- Nim

We'll think about our first lab

- How do we break down a complex program into discrete tasks?
- How should we begin to think about program design?
- What exactly is the first lab?

Our First Program: Hello World

Of note:

- `public static void main(String args[])`
  - The entry point into any Java program
- `System.out.println(...)`
  - How we communicate text to the outside world (i.e., the terminal)
  - Arguments are converted to String objects using their toString() method
    - More on this later!
- Everything in Java is a class, even if there is nothing in it!
  - `public class Hello`, but we never create a `Hello` object or call `Hello` methods...

Programs accepting input: Sum.java

Of note:

- `args[]` array contains command line arguments, one String per element
- Must convert to appropriate type — see Javadoc when you have questions!
- `java.util.Scanner` lets us receive user input *interactively*
- `java.util.Random` lets us pull numbers from a pseudo-random number generator

Interactive Program Example: Nim

Nim is a "popular" game played with piles of matchsticks

- Players take turns removing matchsticks from piles
  - Each turn, player must remove a nonzero number of matches from a single pile
- A player loses when it is their turn and there are no matchsticks left to remove

Let's play a demo!

Interactive Program Example: Nim

Design process from last class:

1. Identify data for a 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

Let's focus on 1, 2, and 3... we'll tackle 4 tomorrow!

Tips for writing Java

- Start with comments
  - Make a plan before you write any code — debug your logic before your Java
- Always Be Compiling
  - It is easier to catch errors early, one at a time
- Write in testable units
  - Compile, run, compile, run, compile, run …
- When possible, move code into methods so you can reuse it!
  - Common operations? Displaying the board, asking for inputs, etc.

Lab 1: Coinstrip

Similar game to Nim:

- Players take turns moving coins
- Player who can't move loses
- Rules are slightly more interesting
- Representing the game state gives more design flexibility/choice

Demo!