Notes

Have a great Spring Break!

Lab09b hard—sorry about that.

Java File I/O Revisited

```
File students = new File("/users/phaskell/CS112/roster.txt");

boolean students.canRead();
boolean students.canWrite();
boolean students.isDirectory();
boolean students.delete(); // careful!
boolean students.exists();
int students.length(); // if 'students' is a file
String[] students.list(); // if 'students' is a directory
```

A File just links to a file in the computer filesystem. Can't do any reading/writing with it

Java Stream Types

FileReader, Scanner

- `FileReader fr = new FileReader(new File(path));`
- Reads a char or an array of chars from the File
- Handles mapping the computer's native character set to (from) Java's UTF-16
- Scanner…

FileWriter, PrintWriter

- FileWriter writes a char or array of chars to a File
- PrintWriter prints standard data types as text
- print(char), print(double), print(String), println(…), etc

(Other Readers and Writers handle bytes, objects, etc)

Testing

Discussion on Testing

I have said "think about testing as much as about sw design" "Spend as much time testing as coding".
WHY?

- Studies have shown that a bug found in SW dev has 10x cost as bug found in design
- Bug found in testing has 10x cost of a bug found in SW dev
- Bug found by end user has 10x cost of a bug found in test

Famous bug stories
- Integrated circuits build with “silicon” and “metal”… A bug in chip design costs $1M or $2M, plus delay. Is SW different?

- One of a supplier: compiler bug. 2 hr mtgs with 15 senior people daily for 3 months. 3 month delay to product launch. Millions of dollars
- One of my team: DISH counter rollover. I had my team spend months modifying our SW so we could simulate 4 years of run-time in 1.5 months. Team continuously runs this test, and it occasionally finds errors

HOW?

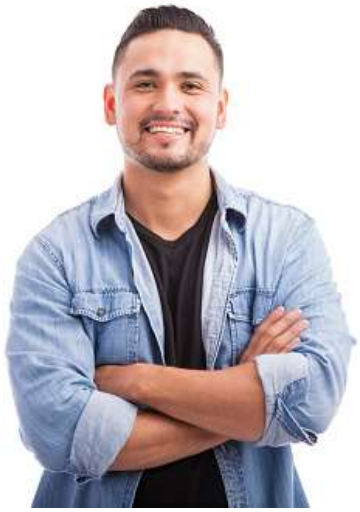
- Not easy to shift between developer and tester mindset

- I had to manage both groups for ~1 year and could not do a good job. But we have to do our best
- Design input cases to try to break your SW
- Expect bad "environment": networks don't respond, other programs crash, CPU stalls, etc
- And design test cases for every failure you can think of

AUTOMATION
- Any sw beyond a class assignment will be run repeatedly, and modified
- Need to run tests repeatedly. Can't have people (including yourself) do it. Too slow and expensive
- Write computer program to test other computer programs. SUPER HOT FIELD
- Every week I spend 3x as much time creating testers for your HW assignments as I do writing actual reference solutions
- Remember EightDoubles.java? You can create strings of possible inputs and expected outputs, and modify EightDoubles to run thru and see. Too few numbers, not actual numbers, etc.

DESIGN TESTS FOR MyCardDeck! Careful reading of the spec, creative thinking. Positive and Negative test cases

Coding Style



Coding Style

Sometimes called "coding conventions"…

In some ways this is an art class. Your code needs to have some style, not to be pretty, but to be clear and easy to follow.
Like good prose. Look at "weird.c"
Another part of style is how the source code looks: indentations, where you put the brackets, style for identifier names. People have a lot of opinions on these. I have negotiated several big company-wide exercises to settle on these. And every few years, people want to try again. Here are the conclusions:

- If you are working on an existing project, be consistent with existing style
- Be consistent with yourself
- Write good comments, JavaDoc comments for classes and public functions
- Pick good names for identifiers. "camelCase"
- Put class variables @ top of class defn, before methods
    - Method variables declared just before use
- Use 'final' whenever you can
- Use 'public' and 'private'. Really try to avoid nonfinal public class variables
- Break up long methods
- Indentation: go in one "level" for: class members, statements in a method, statements in a code block