CS 2113
Software Engineering

Lecture 13: The End.

Professor Tim Wood - The George Washington University

The End is Near

- This is our last class :(

- Project 3 due Sunday Dec 18 at 11:59PM
  - Earn 1 bonus point per day early up 5 (on Dec 13)

- Final exam: Wed Dec 14, 5:20-7:20PM

- Today:
  - Software Engineering
  - Your future
  - Your skills

Office Hours

- **Prof. Wood:** Friday 11:30am-2:30pm
- **UTF:**
  - **Friday 3pm-5pm TOMP 402?**
  - **Sunday 1-3pm TOMP 402**
  - **Monday 4:30-6:30 SEH 4040**
  - **Tuesday 13th: 10AM - SEH1300? (review)**
- **TA:** Monday 10:10-11am, 1:30-2:10pm TOMP 405

Final Exam

- Short answer (~35%)
  - Java OOP, GUIs, Networking, Threading concepts
  - C memory diagram!
- Small program (~20%)
  - Basic java syntax, arrays, loops, logic
- OOP program (~45%)
  - Will involve GUIs, networking, and/or threads
- (weights are only an approximation)

- I will give you: GUI/net/thread sample code
- You can bring: a double sided sheet of handwritten notes

2 hours (shorter than midterm)

Exam Material

- C - memory diagram (stack/heap/malloc/free)
- OOP
  - Inheritance, polymorphism, abstract, interfaces
- GUIs
  - Event handlers, Inner classes
- Networking/Files
  - Sockets, sending/receiving
- Multi-threading
  - Defining a thread's task, starting threads

Mental Model: Threads

- multiple cash registers versus 1, multiple lanes
- Workers + jobs
- Some class that implements Runnable
  - run() ---> job to be done
- Thread object (worker)
  - Give a job when we start it

Programming Practice

- Practice problems from lab
- Threaded Factoring
  - What is the basic approach?

Mental Model: Networking

- Postal system - send mail, receive mail
- Power grid: electrical sockets
  - stream of current in and stream of current out
- Water system -

- socket -- connection to somebody else
- BufferedReader: input stream - waits for input
- PrintWriter: output stream
- protocol: define message ordering

Programming Practice

- Practice problems from lab

- Email Client
  - What is the basic approach?

Software Engineering

- In this course...

- Programming "in the small"
  - for, if, while
  - Basic logic of each method

- Object oriented design
  - Classes, interfaces, inheritance
  - Sharing data

Software Engineering

- In the real world...
- Planning how to build software
- Managing the process of building software
- Testing that software works correctly

How to Plan a Project

- Don't start by writing code
  - It worked in CS1111 and CS1112
  - It will **not** work in your future classes
  - It will **not** work in your future career

- Think about structure and goals of your program
  - What are the main components?
  - What does do they need to do?
  - What are the main pieces of data?

- DO NOT:
  - Search google for something similar.

Design…

- A game:
  - Candy Crush
  - Pokemon Go
- An office app:
  - PowerPoint
  - Word
- Media/Entertainment:
  - iTunes
  - Facebook

Application structure?

Classes?

Important data and methods?

Threads?

Networking?

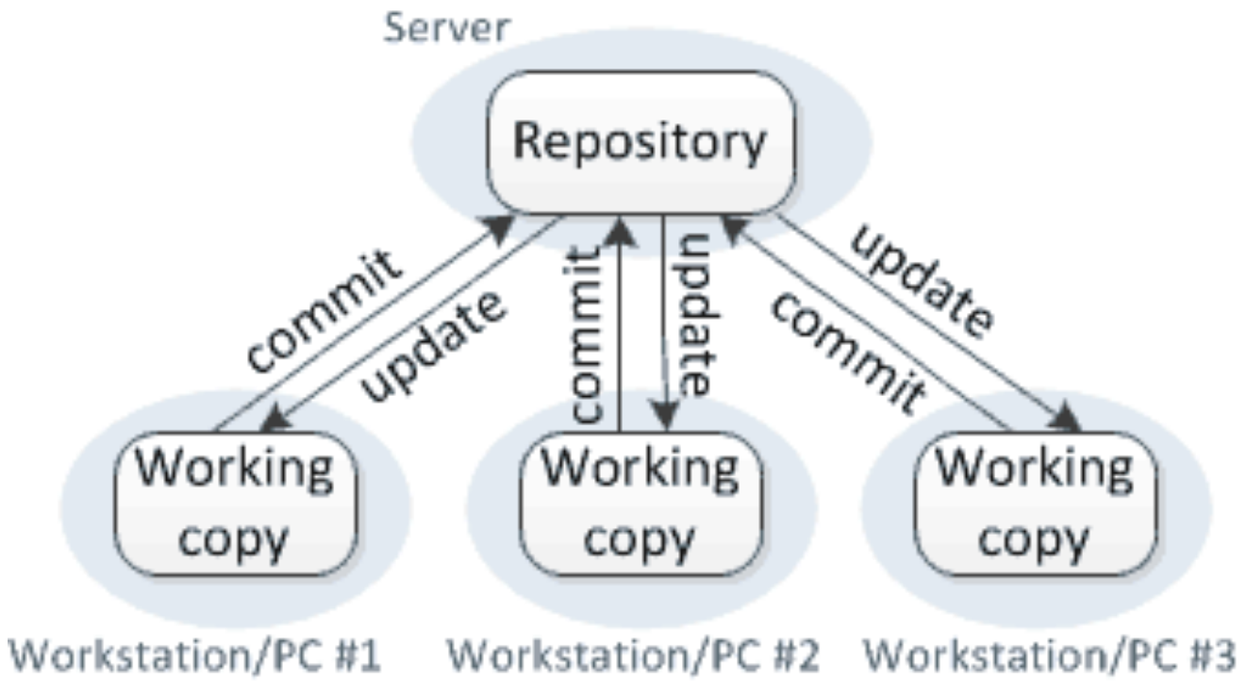
Agile Development

- Traditional software engineering (Waterfall)
  - Spend a lot of time making a detailed design plan
  - Build the software following the plan
  - Release the software
  - Hope the software works and does what the client wanted
- Agile Development (Scrum)
  - Iterate between design and development
  - Each iteration produces a working prototype
  - Test code and communicate with client throughout process

Version Control

- You won't be the only person working on a project
- How to keep files in sync?

Centralized version control



Understandable Code

- You will spend a lot of time reading other people's code, and they will read yours
- Avoid:

```
// Stupid Code
  1:   int length= 0;
  2:   for(int idx = 0; idx < a.length; i++){
  3:       length++;
  4:   }
  5:   System.out.println("length is : " + length);


// Misleading Code
if ( a )
   if ( b ) x=y;
else x=z;
```

Understandable Code

- Use reasonable names

IDENTIFIER	NAMING RULES	EXAMPLE

Variables	A short, but meaningful, name that communicates to the casual observer what the variable represents, rather than how it is used. Begin with a lowercase letter and use camel case (mixed case, starting with lower case).	`mass`
		`hourlyWage`
		`isPrime`

Constant	Use all capital letters and separate internal words with the underscore character.	`N`
		`BOLTZMANN`
		`MAX_HEIGHT`

Class	A noun that communicates what the class represents. Begin with an uppercase letter and use camel case for internal words.	`class Complex`
		`class Charge`
		`class PhoneNumber`

Method	A verb that communicates what the method does. Begin with a lowercase letter and use camelCase for internal words.	`move()`
		`draw()`
		`enqueue()`

Testing

- How do you know your code works?

- Unit testing
  - Simple tests for every component of your program
  - Eclipse can automatically run them
  - Easily verify that nothing breaks when you make a change

Software Engineering

- **Course Summary**

- Write code in multiple languages and understand the differences between them
  - Syntax, memory management, run time environment, etc

- Organize a project into components
  - Reuse code, improve maintainability and understandability

- Use advanced libraries and language features
  - Threads, networking, database connections, etc

Get a Job

- You are in a good field*
- US News & Reports
  Top 10 Jobs for 2012:

1. Registered Nurse
2. **Software Developer**
3. Pharmacist
4. Medical Assistant
5. **Database Administrator**
6. **Web Developer**
7. **Computer Systems Analyst**
8. Physical Therapist
9. **Computer Programmer**
10. Occupational Therapist

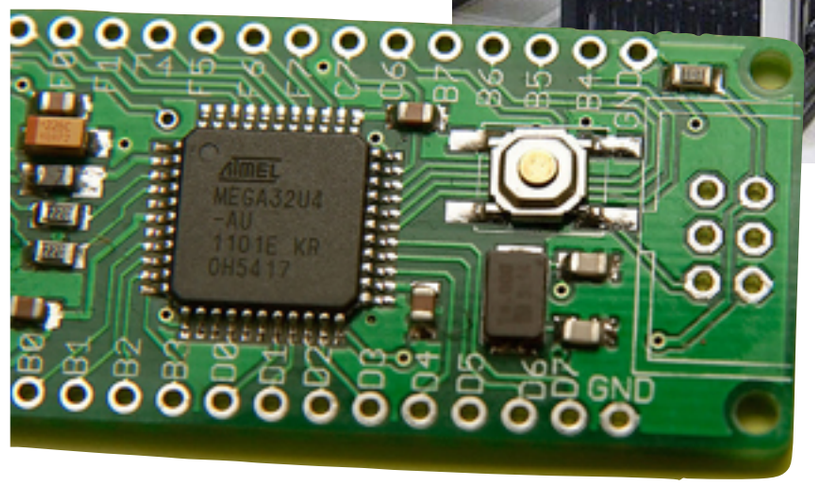
Source: indeed.com

*unless you are one of the non-CS majors in the room

Software Trends

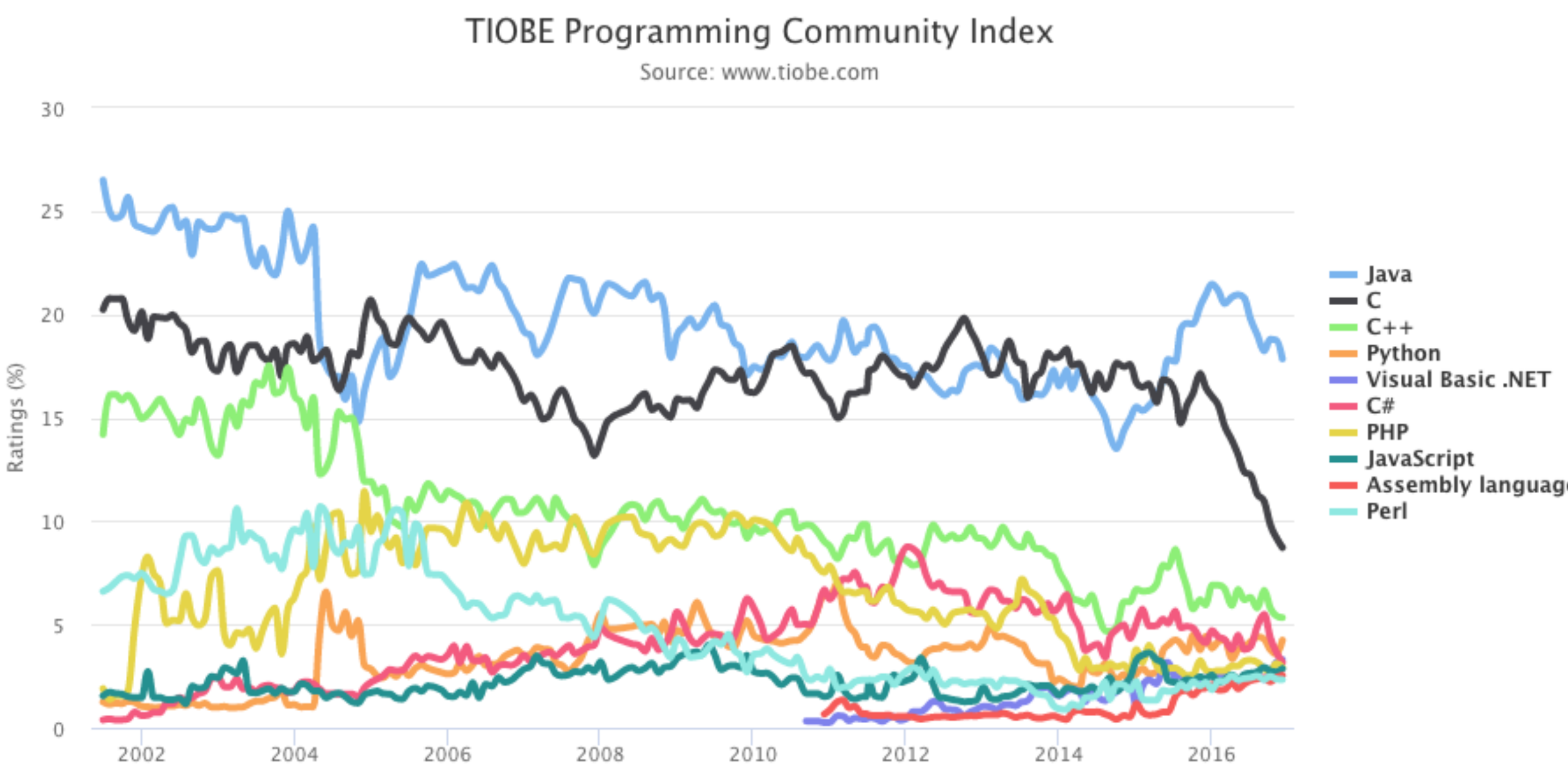
- Software is changing

Then



Now

Language Trends



Change is good

- Technology is always changing
- Some of the skills you are learning now will be useless in 5 years

- **Learn how to learn!**

- Build a mental model of code and software
  - When to use a for/while/if/switch
  - How to move data between classes/files/networks
  - How to structure components to build an application

Your future

- You have <= 2 summers to build your resume
- Get an internship or join a research project
- You need an experience or a piece of knowledge that will make you stand out
  - Good grades aren't enough

Learn Android

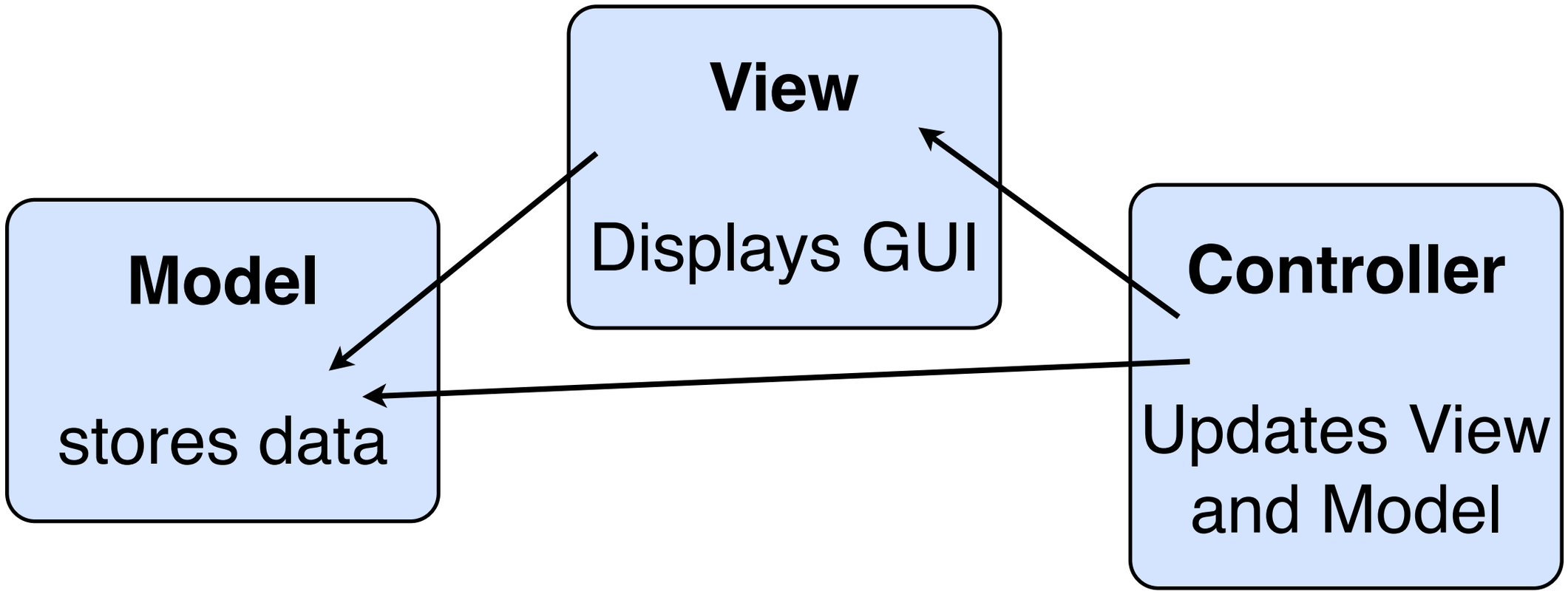
- Android apps are written in java!
- You could be the next app store millionaire!

- Easy to get started: http://developer.android.com/sdk/index.html
  - Install a custom version of Eclipse
  - Includes an emulator to test your apps in
  - Can also load your app onto your own phone

- If you register, you can release your app on the Google Play store

App Architecture

- Android, iPhone, and many web frameworks use the same program structure



- Benefits of this paradigm?

Learn Javascript

- Has become a very popular language for web applications

- Sadly javascript != java

- Buy a book, take an online class, read tutorials
- Tons of libraries to let you build cool things
  - three.js
  - node.js
  - jQuery
  - tracking.js
  - crafty

Evaluations

- Fill online versions and I will give extra credit
  - Only works if enough students complete the surveys

- I will send you a separate survey soon

- If you have any other feedback feel free to email me or come by my office

- How can we stay in touch?

The End.

- You have learned a lot (I hope)
- You have practiced the craft of programming

- Write a program over winter break
  - tell me what you do!

- Learn a new language, try a new library, modify an open source project, build better zombies