

CS 342: Software Design

Overview

- Class Overview
- Git Basics
- Break
- Intro to Java

Class Goals

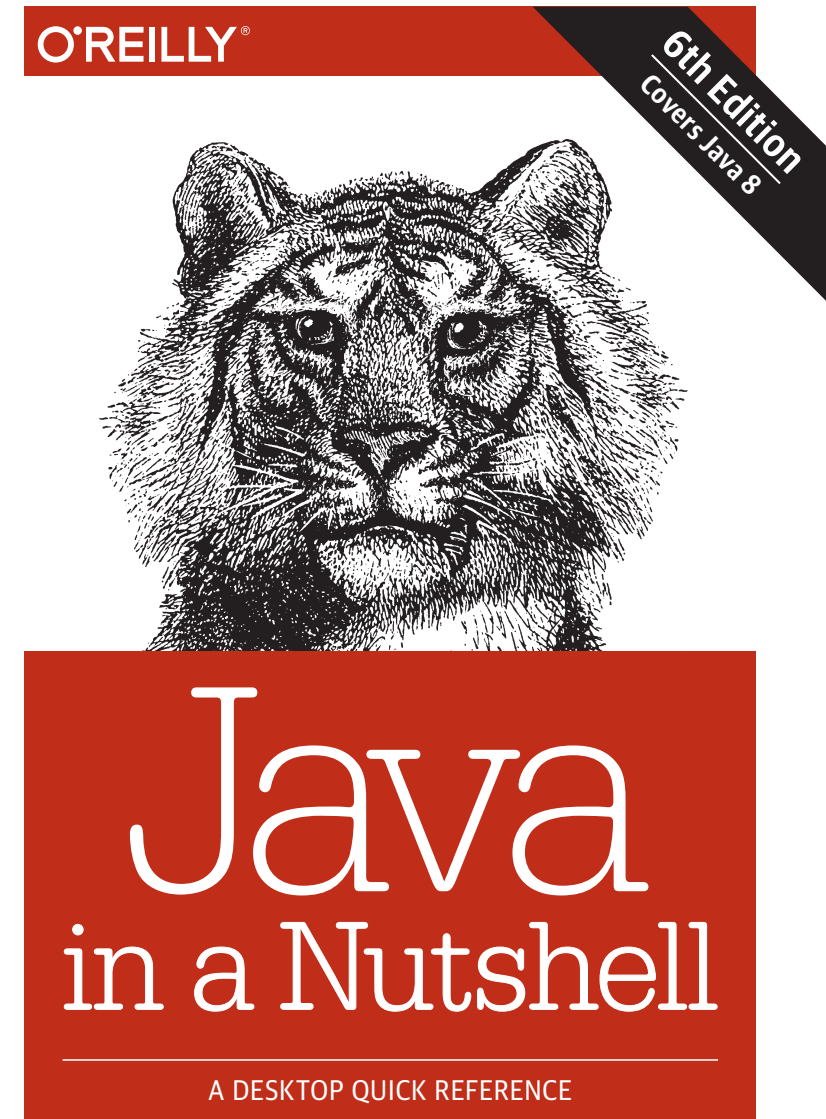
- Learn the Java language
- Use Java to do interesting things
- Generalizable methods, techniques and principals
- Good development practices

Experience

- Java?
- Packages?
- Exceptions?
- Patterns?
- Async?
- Lambda / functional programming?

Grades

- Readings
- Class participation / discussion
- Homework
- Midterm / final



Benjamin J. Evans & David Flanagan

Getting Help

- <https://www.cs.uic.edu/~psnyder/cs342-summer2017/>
- Questions in class
- Office Hours
 - Me: SEO 1218, 2-4pm, Wed and Fri
 - TA: SEO 1380, TBA
- Piazza

Class Framework

- Brief reading quiz (clickers)
- Discussion on quiz
- Lecture (and discussion)
- Break
- Lecture (and discussion)

Updates and Info

- Follow Piazza
- Website for readings and dates
- Email for emergency situations

Getting Setup

- Java 8
- Git
- SSH
- Linux
- Basic Java environment (ie choose your IDE)

Version Control

Git

- Open source
- Keeps track of changes on disk
- Has useful tools to synchronize remotely
- Authentication tools
- Tooling (github, bitbucket, gitk, etc.)

Git Basics

- `git init`
- `git status`
- `git add <some file(s)>`
- `git commit -m <something>`
- `git checkout`


Git Networking

- git clone
- git pull
- git push

Much Much More Git

- git remote
- git log
- git revert
- branching, tags, diff
- Website

Git in this Class

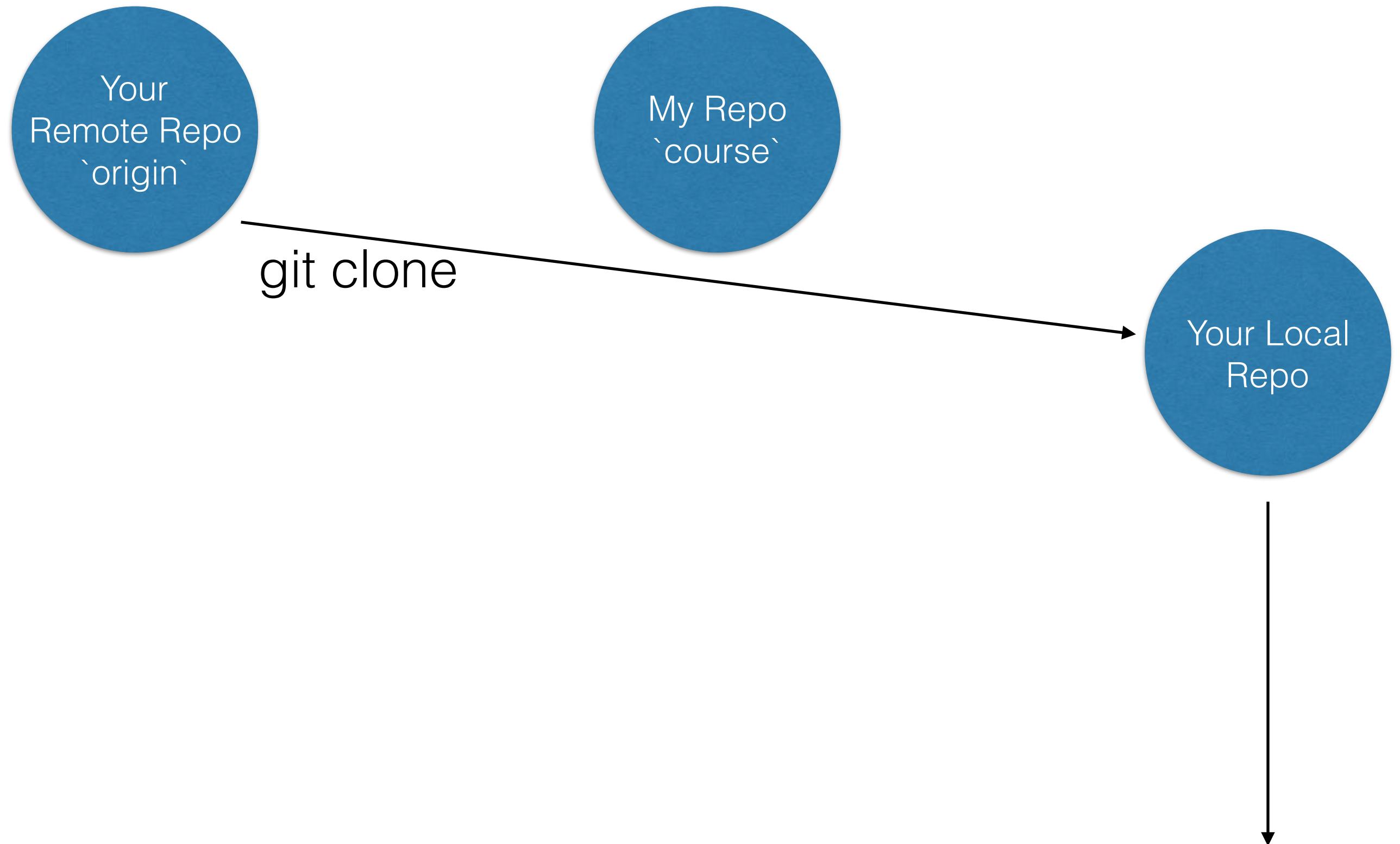


Your
Remote Repo
`origin`

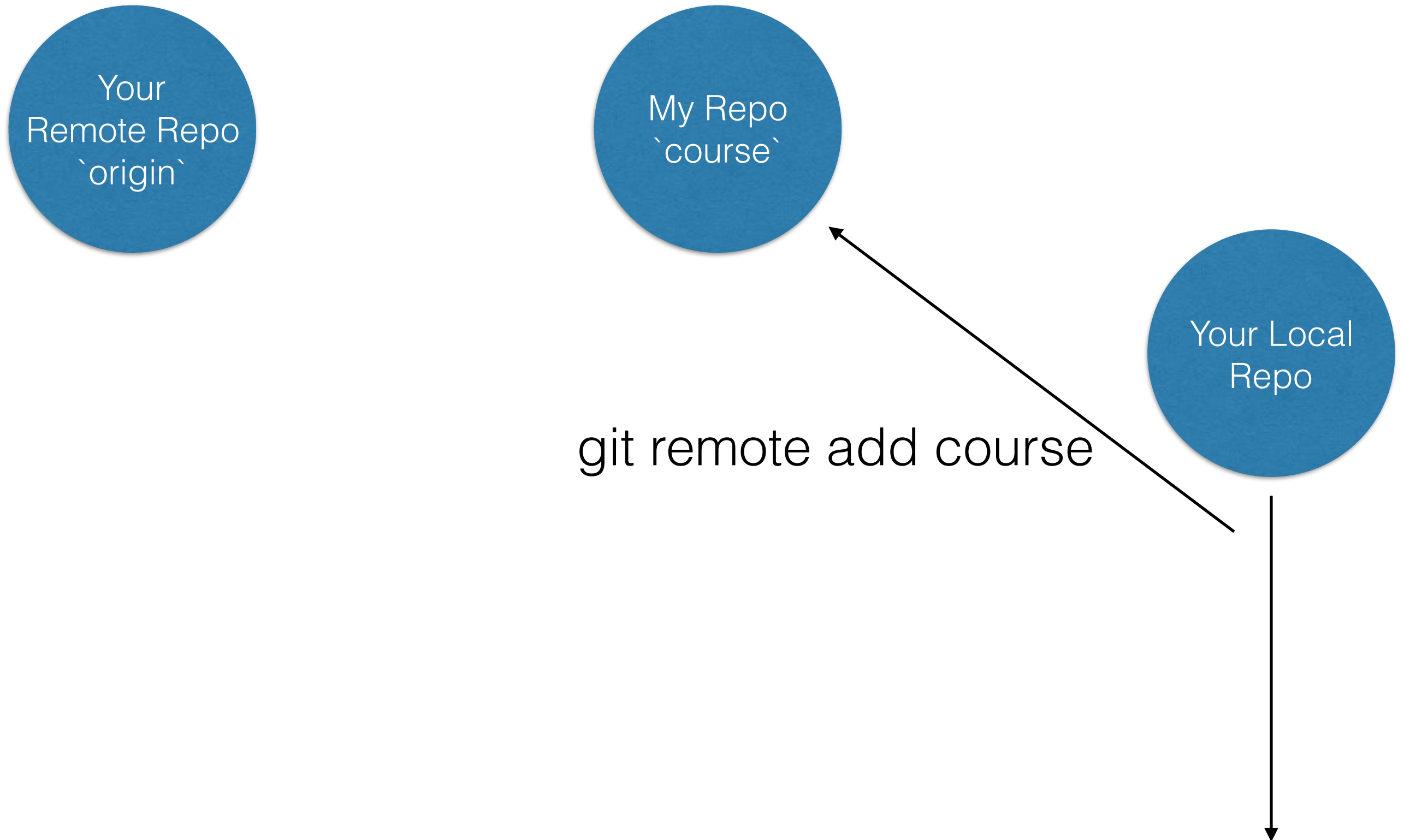


My Repo
`course`

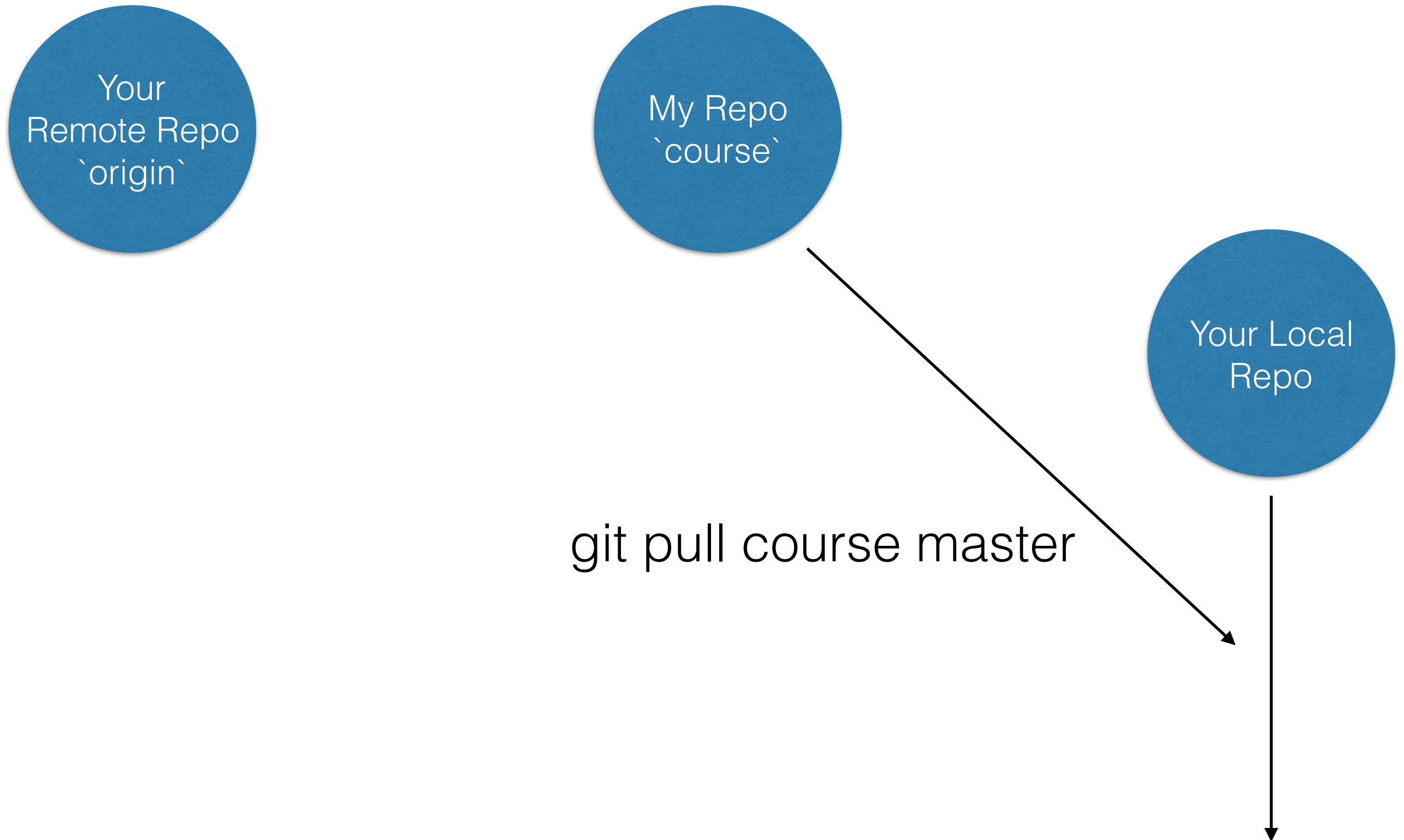
Git in this Class



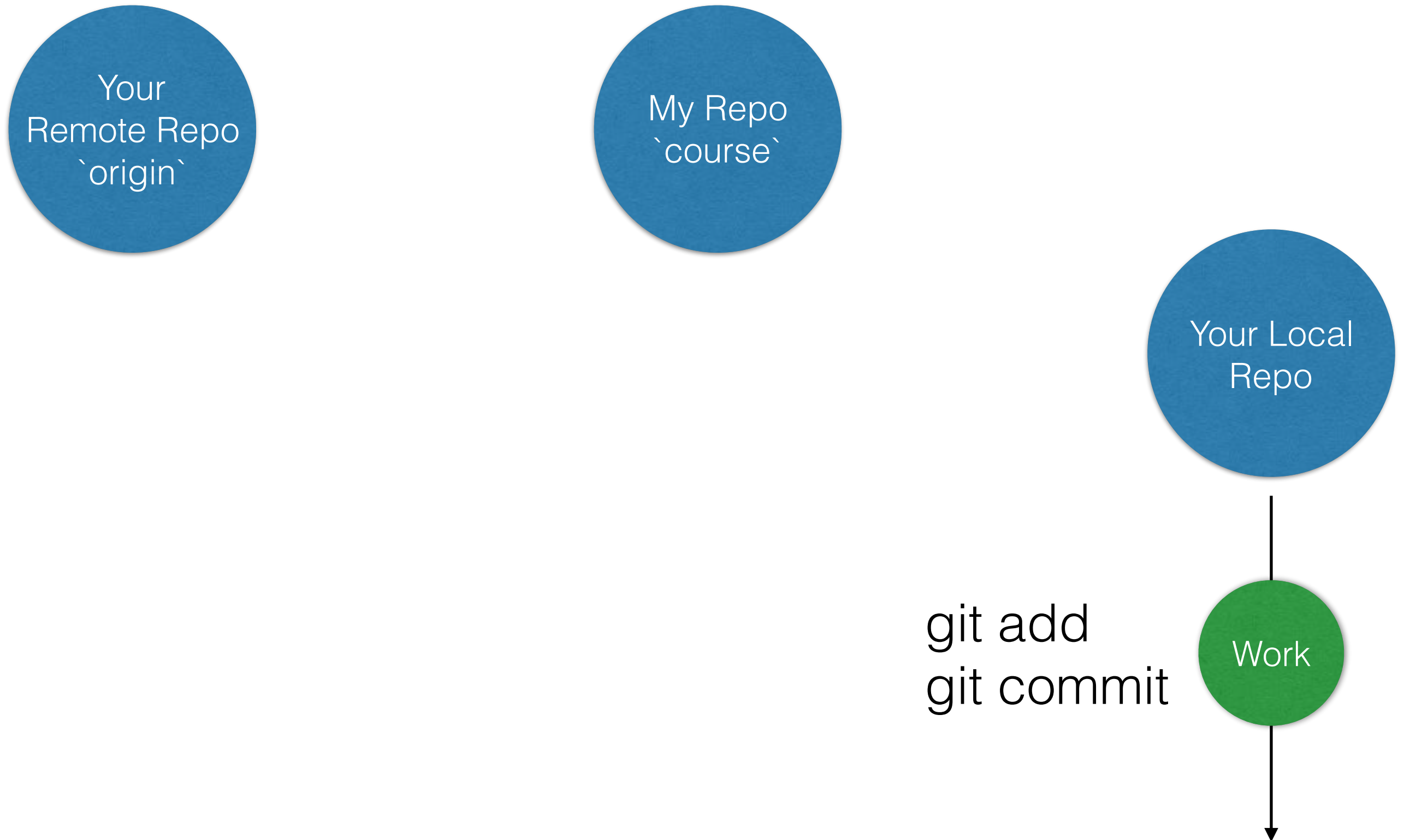
Git in this Class



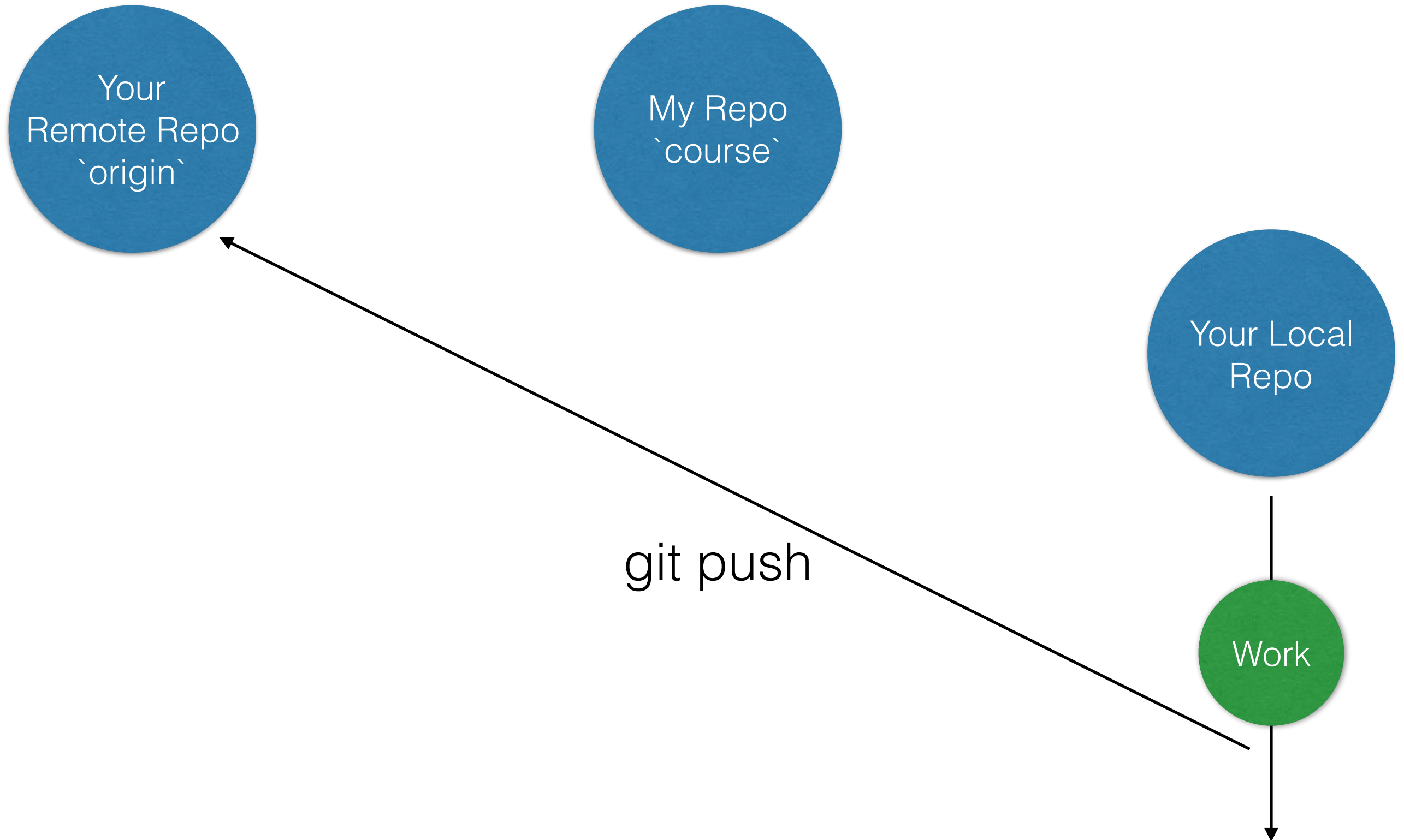
Git in this Class



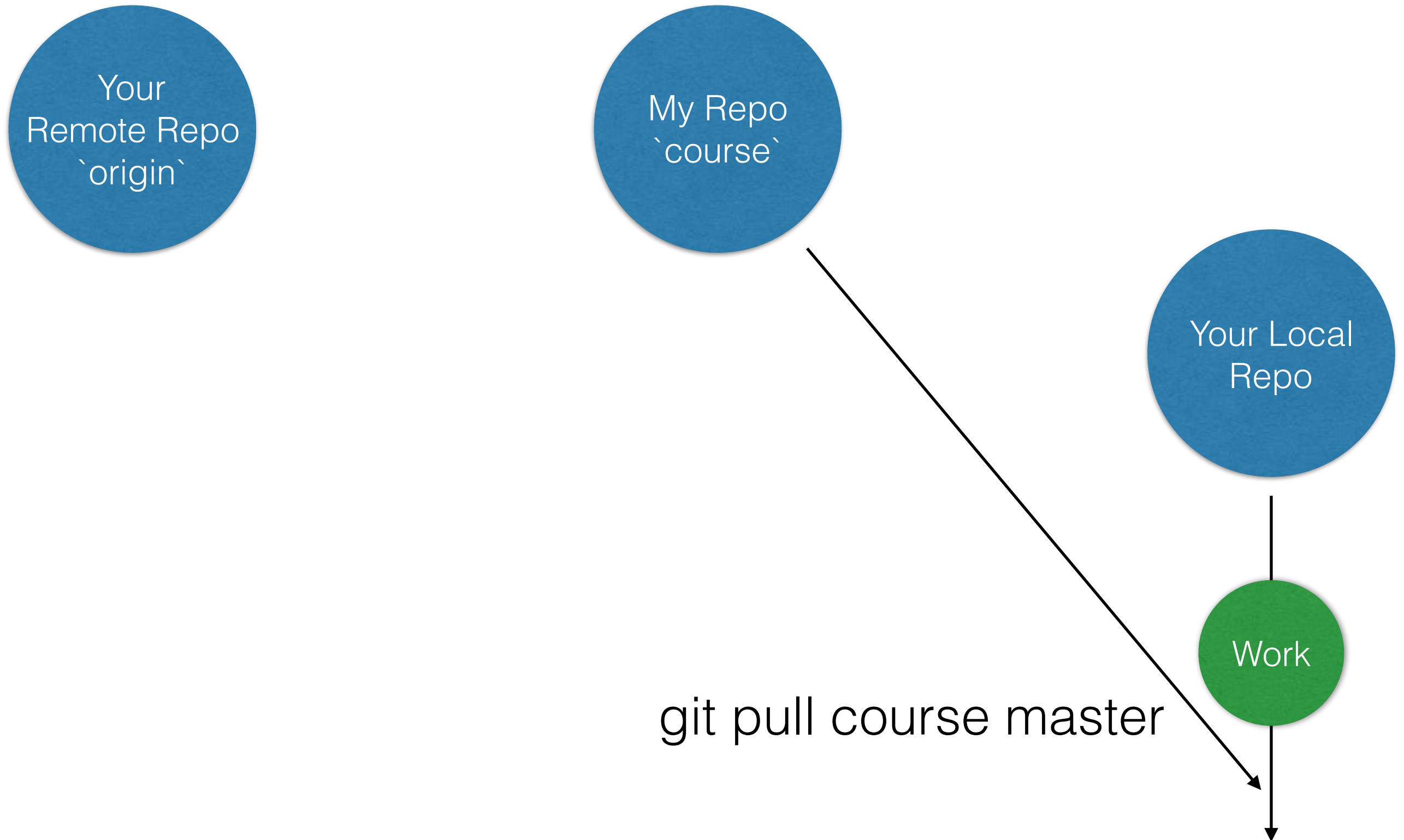
Git in this Class



Git in this Class



Git in this Class



Git Demo

Final Git Thoughts

- .gitignore
- .git/
- Its complicated!



SEE ME
SEE YOU

Programming
Languages?

Object Oriented
Experience?

Java Experience?

Java Basics

- Originally Oak, Java in 1995
- Commercial language
- Crazy popular
- “Boring” reputation



Java Highlights

- Memory managed
- Object oriented
- JVM / Runtime
- Fast (in some ways)

Java Highlights

- Bytecode
- Packages / namespaces
- Type safe
- Rich ecosystem
- Compiler improvements (unboxing, <>, etc.)

Java Lowlights

- **Verbose**

```
Integer coolInteger = new Integer(8);
```

```
List<String> greatStrings = new List<String>();  
greatStrings.add("Hachi Machi");
```

Packages

- **Heavy Tooling**

IDEs, refactoring

Java Lowlights

- **Fragile, Deep Type Hierarchies**

Taxonomies are difficult, reworking them is tricky

JSONReader, YAMLReader, SocketReader?

- **Industry / Enterprise Focus**

XML, less open source uptake (especially earlier)

Package management...

Java Lowlights

- **Creaky in Points**

Primitives vs. Arrays vs. Objects

```
int[] = {1, 2, 3};
```

```
ArrayList<Integer> aList = new ArrayList<Integer>();
```

```
aList.add(new Integer(4));
```

```
aList.add(new Integer(5));
```

- **Redundant**

So many ways to do the same thing...

Java Lowlights

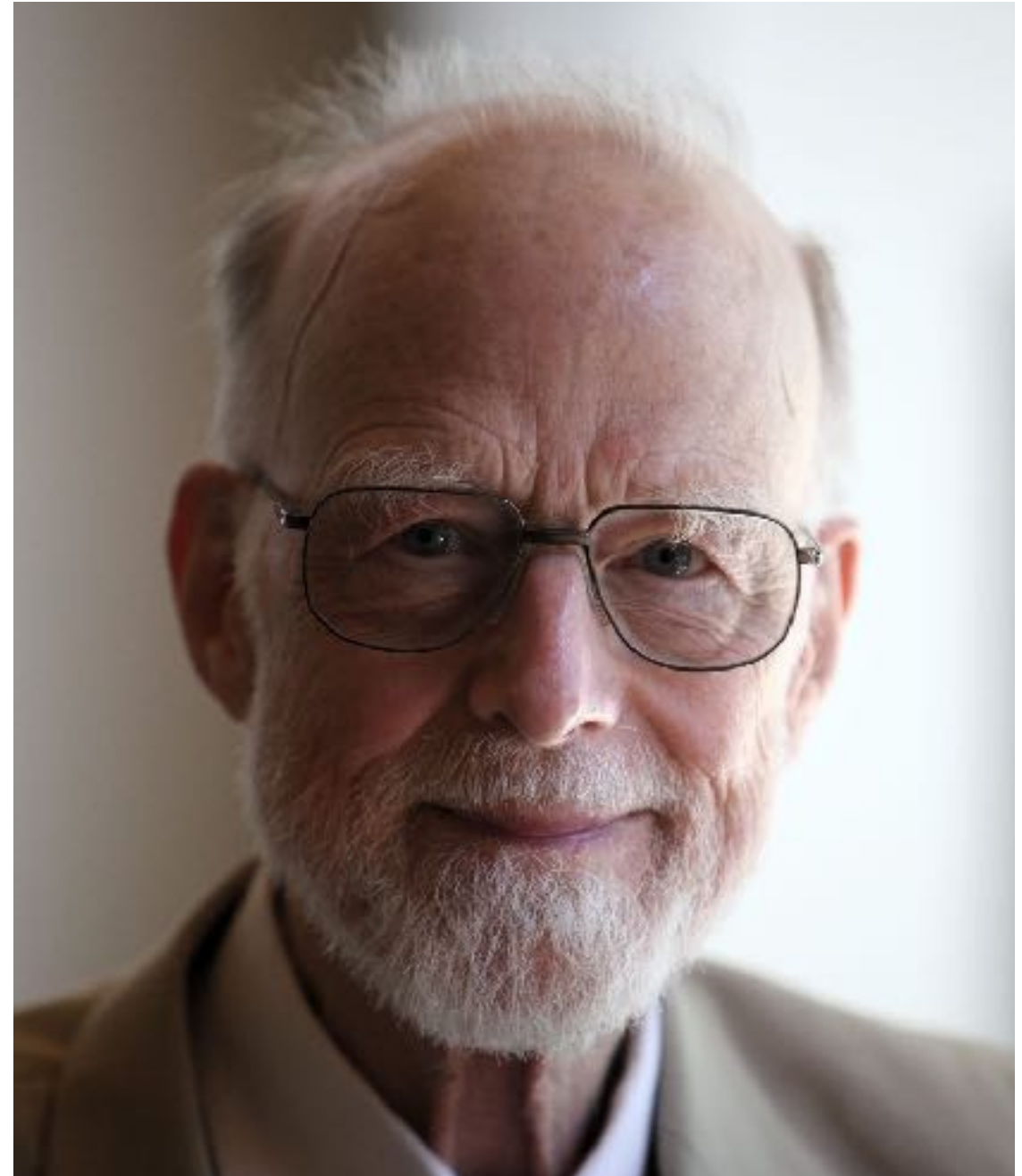
- **NULL**

“billion-dollar mistake”

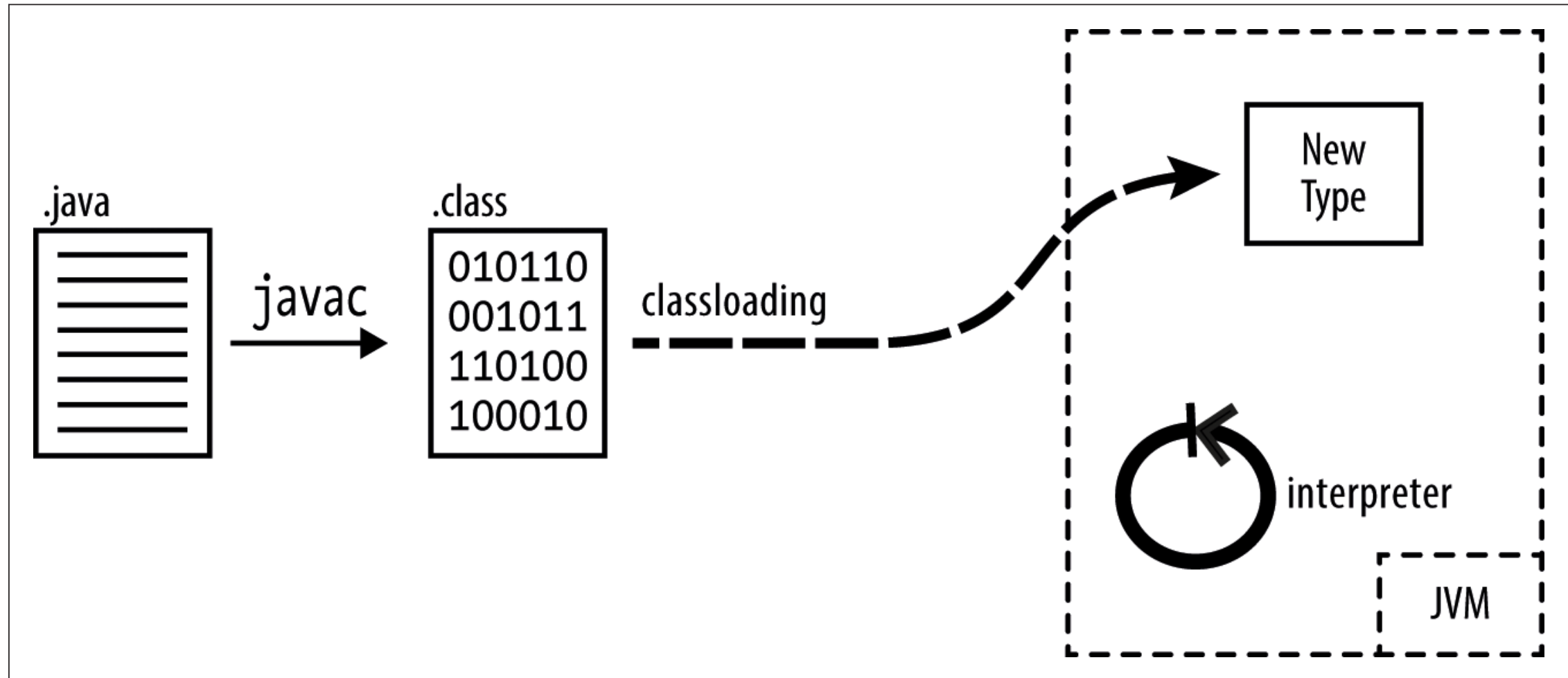
```
String radString = null;  
if (radString != null) {  
    radString.stringThing();  
}
```

- **Oppressive Type System**

Glue classes, functional programming, etc



Using Java



Java Hello World

- Boiler Plate and ``main``
- Primitive types
- Control flow
- Comments
- References
- Calling methods
- Types
- Building
- Running

Java Demo

Wrapping Up

- Feedback is welcome, all the time
- <https://www.cs.uic.edu/~psnyder/cs342-summer2017/>
- Register for Piazza
- Email me
 - ssh public key
 - two or three languages you're most familiar with
 - discussion group preferences