Objectives

- Introduction to Java
- Introduction to Version Control

1

Review

- What questions should you ask to determine if some state should be a local, instance, or class variable?
- What are some Unix commands? What task do they do? How do you use them?

2

INTRODUCTION TO JAVA

3

What is Java?

… and, why should I learn it?

- From Sun Microsystems
  - 1995, James Gosling and Patrick Naughton
  - Specifications
- Object-oriented
- Rich and **large** library
- Develop cross-platform applications
  - Web, desktop, embedded
- Widely used
  - Frameworks to enable easier development

http://www.tiobe.com/tiobe-index/

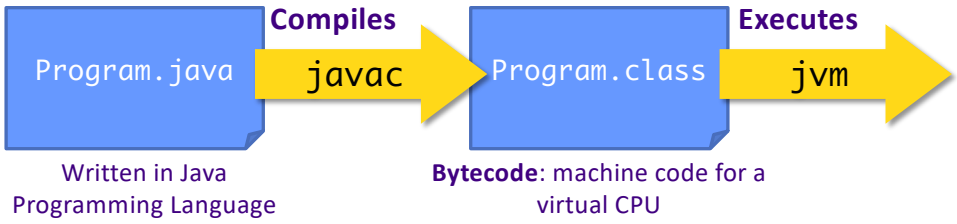
4

What is Java?

- Java Programming Language
- Java Virtual Machine
- Java Class Libraries

5

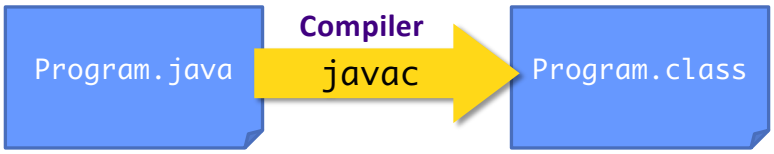
Overview: Compiling, Executing Java Programs



6

Compiling Java Programs

Step 1:



Written in Java Programming Language

Bytecode: machine code for a virtual CPU

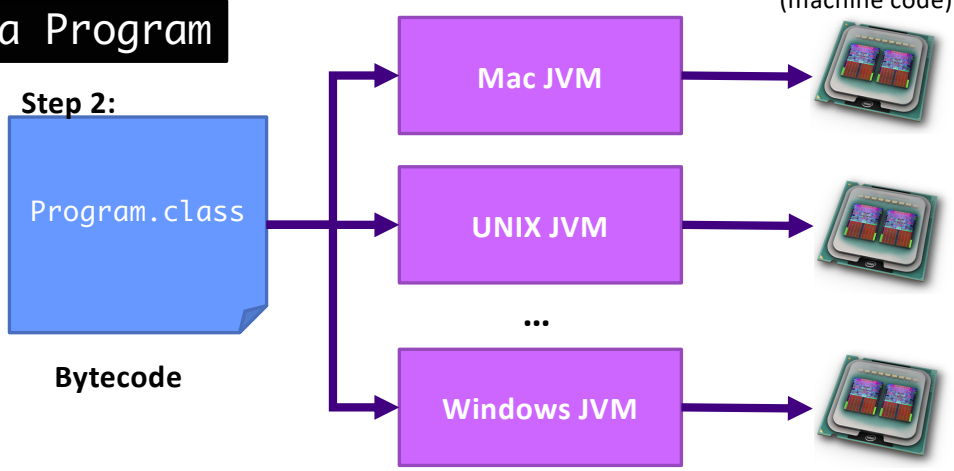
```
javac Program.java
```

- Compiler catches some errors
- Need to fix those errors and recompile

Executing Java Programs

```
java Program
```

Step 2:



- Same **bytecode** is executed on each platform
- Don't need to provide the source code

Java Virtual Machine (JVM)

- Emulates the CPU
  - Usually specified in software (rather than hardware)
- Executes the program's ***bytecode***
  - Bytecode: virtual machine code
- JVMs available for each Java-supported platform
  - Enables program *portability*
- HotSpot VM
  - Code dynamically compiled to machine code
- Garbage Collection

Traditional (C/C++) Program Execution

- Example: I use my Mac-specific compiler to compile program into a Mac-specific executable
- Limitation: Executable is not portable

How does Java's approach affect distribution of software?

2 - How does software being Java-based affect its distribution?

Makes it harder because you have to install a JVM on every machine	A
Makes it more secure because you don't provide the source code	B
Makes it easier because same bytecode can be run on multiple platforms	C
Makes it easier because many machines already have Java installed	D
None of the above	E

11

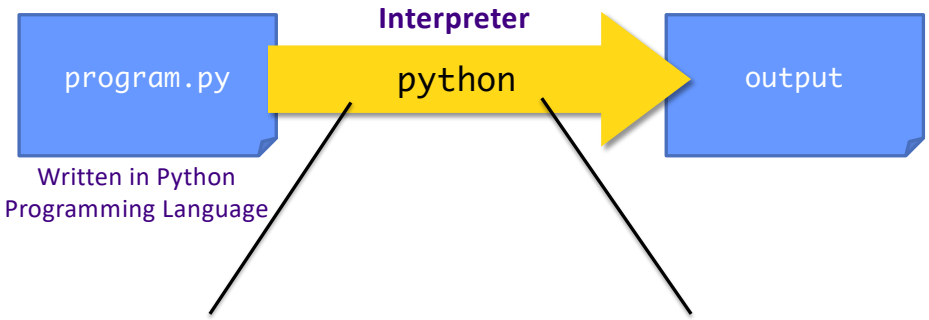
Traditional (C/C++) Program Execution

- Example: I use my Mac-specific compiler to compile program into a Mac-specific executable
- Limitation: Executable is not portable to any OS

What is Python's approach?
How are (I) Java and (II) the traditional approach the same and different from Python's approach?

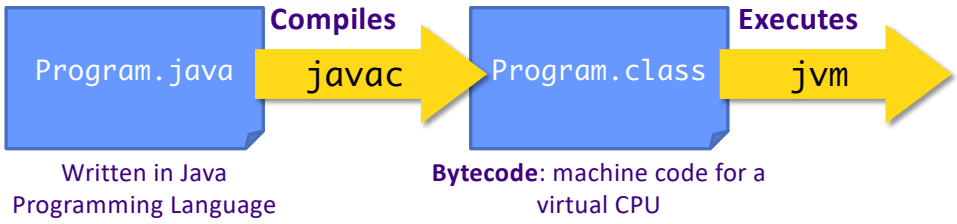
12

Executing Python Programs



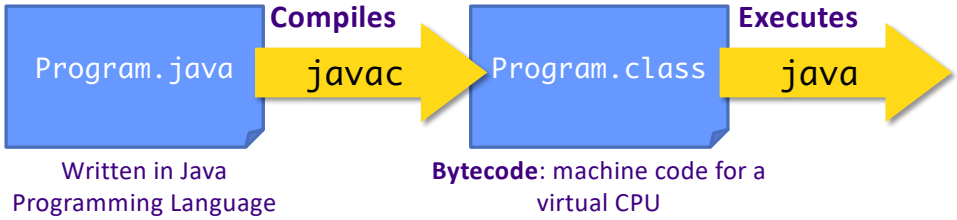
1. Syntax validation
   - exit if not valid
2. Translate Python code to Python bytecode
   - Bytecode stored in __pycache__ directory
3. Python virtual machine execute Python bytecode

Overview: Compiling, Executing Java Programs



JDK: Java Development Kit

- Contains
  - `javac`: Java compiler
  - `java`: Java Virtual Machine
  - Java class libraries



Java Class Libraries

- Pre-defined classes
  - Included with Java Development Kit (JDK) and Java Runtime Environment (JRE)
  - View the available classes online:

https://docs.oracle.com/en/java/javase/18/docs/api/index.html

- Similar in purpose to *modules* available for Python

What is Java?

- Java Programming Language
- Java Class Libraries

} What this course is about

- Java Virtual Machine
  - Use the JVM but won't learn about how it works
  - For more information on JVM:
    http://docs.oracle.com/javase/specs/

Bringing It Together: Benefits of Java

- Rapid development of programs
  - Large library of classes, including GUIs, Enterprise-level applications, Web applications
- Portability
  - Run program on multiple platforms without recompiling
- Compiled
  - Find some errors before execution!
    - Statically typed
  - Can give performance boost by doing optimizations

VERSION CONTROL: GIT

Motivating Version Control

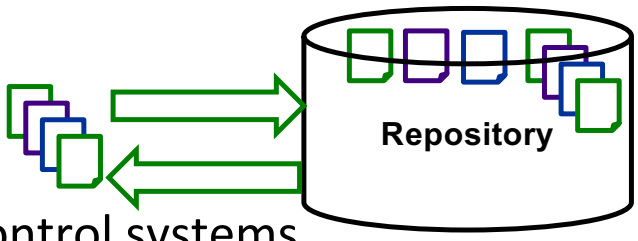
- Do you name your documents like the following?
  - Paper-final.pdf
  - Paper-final2.pdf
  - Paper-final_real.pdf
  - Paper-FINAL-final.pdf
- Do you sometimes break your code so badly and really want to go back to a previous state?
- Do you forget why you made a code change?
- Do you want to just try something out and, if it doesn't work, revert back?

Motivating Version Control: Collaboration

- How do you share documents with teammates?
  - Emailing with conflicting updates?
  - Google Docs/Box Notes → not meant for code
  - Merging contributions
  - Who has the *real* version of the documents?

Version Control Systems Can Help!

- Main idea: ***repository*** holds the code and all changes to it
  - Need to push and pull code to and from the repository



- Centralized version control systems
  - E.g., CVS, Subversion, …
- Distributed version control systems
  - E.g., Git, Mercurial, …

Git & GitHub

- We're going to use Git
  - Distributed version control system
- Our repositories will be hosted by GitHub
  - How you'll get code from me
  - How you'll submit assignments

GitHub's Octocat

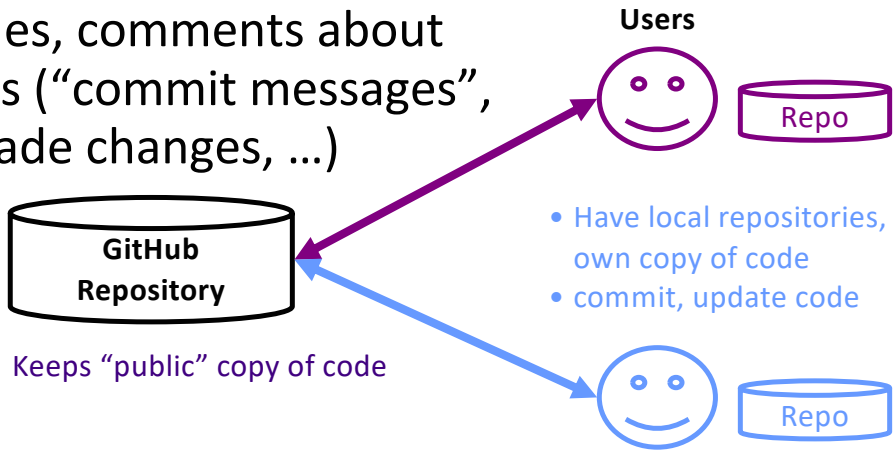
Common Git Commands

Command	What it does
clone	Clones a repository – sets up your repository so that you can coordinate
add <*file*>	Adds the *file* to the staging area
commit	Commits all the staged files (locally)
push	Push all your changes to the remote → You need your code to be pushed so that I can see it.
branch	List all local branches
branch <*name*>	Creates a new branch named *name*
checkout <*name*>	Switches to the branch named *name*

https://xkcd.com/1597/

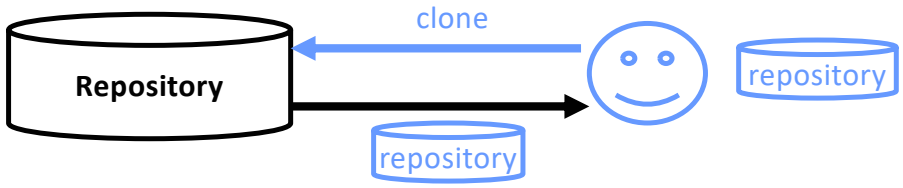
Using Git

- Git is a *distributed* VCS
- ***Repositories*** store all versions of all files, comments about changes ("commit messages", who made changes, …)



Using Version Control: `clone`

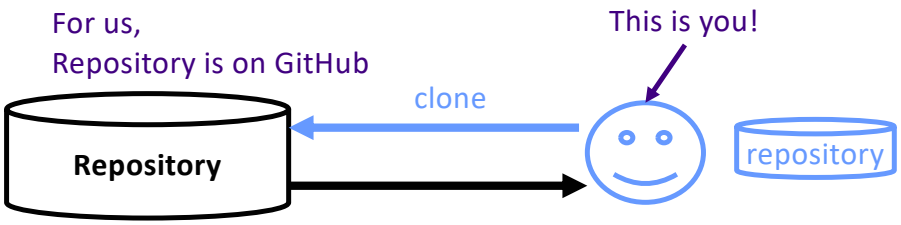
- Our typical workflow: first, **clone** the repository




```
git clone url_of_repository
```

Using Version Control: `clone`

- Our typical workflow: first, **clone** the repository



Using Version Control: `commit`

- After you make changes that you want to document, **commit** your version
  - Include comments about changes you made and ***why***

- Updates each modified file
- Records comments with updated files

```
git commit
```

Using Version Control: `add, delete`

- You need to **add** and **delete** files and directories to the *staging area*, then **commit**

- Marks the files that will be part of the next commit
- When you commit, these files are added to your local repository

- Add, delete files and directories

```
git add myfile
```


Using Version Control: Commit Messages

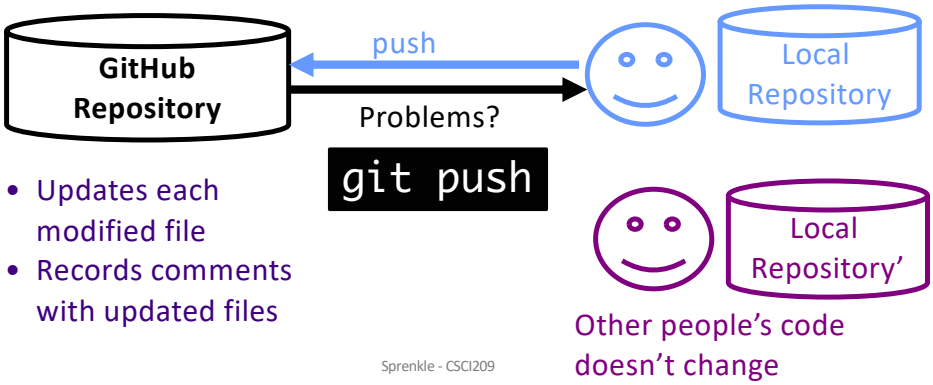
- Many different conventions
- Make your messages meaningful and descriptive
  - Emphasis on the *why*
  - Your future self and contributors will thank you
  - Especially as you move to bigger projects with more collaborators

When Should I Commit?

- Depends – up to you
- Every time you get the next "thing" working
- Every time you fix a bug
- Best practice:
  - Don't wait until you're completely done an assignment before committing

Using Version Control: **push**

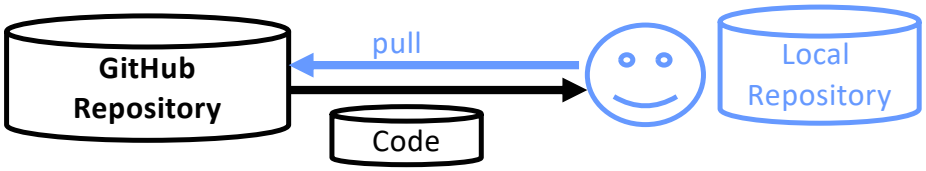
- After you make changes that you want others (at first, that's just me) to see, **push** your version
  - Sends your previous commits and associated comments



- Updates each modified file
- Records comments with updated files

Using Version Control: `pull`

- To see the *current* version of the code in the remote repository, **pull**
  - Resolve conflicts (more on this later this term)




```
git pull
```


Using Git: Branches

- We create branches when we want to create a new "sandbox" to play in for …
  - New functionality
  - Bug fixes
  - Different approach

Why the Command Line?

- Because you *should* know it
  - Alumni feedback
- It can make your development process quicker
  - After you get used to it
- Because you look so badass using it

Common Git Commands

Command	What it does
clone	Clones a repository – sets up your repository so that you can coordinate
add <*file*>	Adds the *file* to the staging area
commit	Commits all the staged files (locally)
push	Push all your changes to the remote → You need your code to be pushed so that I can see it.
branch	List all local branches
branch <*name*>	Creates a new branch named *name*
checkout <*name*>	Switches to the branch named *name*

Got it?
Let's practice!

https://xkcd.com/1597/

Looking Ahead

- Rest of today: working on Git Lab
  - Due Monday before class
  - In general, reload your assignments in the browser when you return to them
    - I may have updated based on student questions