CISC 3115 TY2

Writing Java Programs from Command Line

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Authoring Java programs
- Compiling and running Java programs from command line
- Submitting class Journal (using git and Github classroom)
- CodeLab Registration

Review: Authoring a Java Program

- Let’s consider the following 5 components
  - Requirement
  - Design
  - Implementation
  - Verification (commonly, testing)
  - Validation
- Call them 5 components instead of 5 steps, because it is not necessary to follow them in the above order

Requirements

- About answering question:
- What does the “customer” want? Call the answer the requirement.
  - In the class:
    - What does the instructor want?
  - For your own exploration:
    - What do I want?

Design

- About answering question:
- What is the program supposed to do to meet the requirement? Call the answer the specification.
  - What is the functionality? How should the program “behave”?
  - What data structures should I use?
  - What is the algorithm?
  - Additionally,
    - Is there any limitation on where the program is supposed to run? e.g., how much memory do I have? how fast should the program run? what programming language(s) must I use?

Implementation

- About writing the code as specified
- For simple Java programs,
  - Create and edit Java program files
  - Compile the program, revise it if error
  - Run it, revise the program/find a way to run it if error

Verification and Testing

- About answering the question:
- Does the implementation meet the specification? (Am I *building the thing right*?)
  - Commonly via testing
    - Develop test cases: the scenarios under which the program produces intended result
      - Input, output, and interaction
    - Run test cases and verify the output is identical to the intended one specified in the test cases
    - Revise design and/or implementation till all test cases pass

Validation

- About answering question:
- Do the design and implementation meet the requirements? (Am I *building the right thing*?)

Questions?

- What are major components when authoring a program?

Review: Authoring a Java Program

1. **Requirement**: write a shortest java program, and compile and run it.
2. **Design**: a Java program that prints out “Hello, World!” on the standard output
3. **Implement**
   A. Create/Revise a HelloWorld.java using an editor
      A. Using: the Atom editor, the Visual Studio Code, notepad++ for Windows; SlickEdit ($$$) for Mac OS X, …
      B. The instructor will use Atom for demo in class.

   B. Compile the program, if error, go to step A
4. **Test**
   - Test the program, if failed, go to step 2 (can also be steps 1 and 3)

Demo for Authoring a Java Program

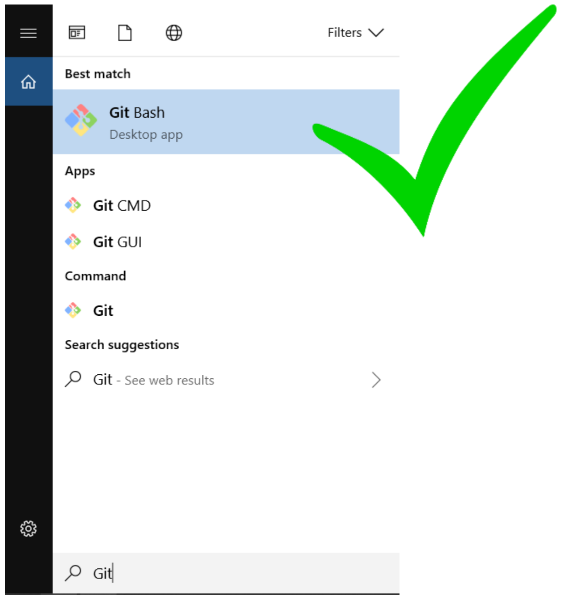
1. Prepare the working environment
   a) Install the git client (if not already installed)
   b) Install the Atom editor (if not already installed)
2. Create HelloWorld.java using the Atom editor
3. Compile the program
4. Test the program

Prepare the Working Environment

1. Install the git client (if not already installed)
2. Install the Atom editor (if not already installed)

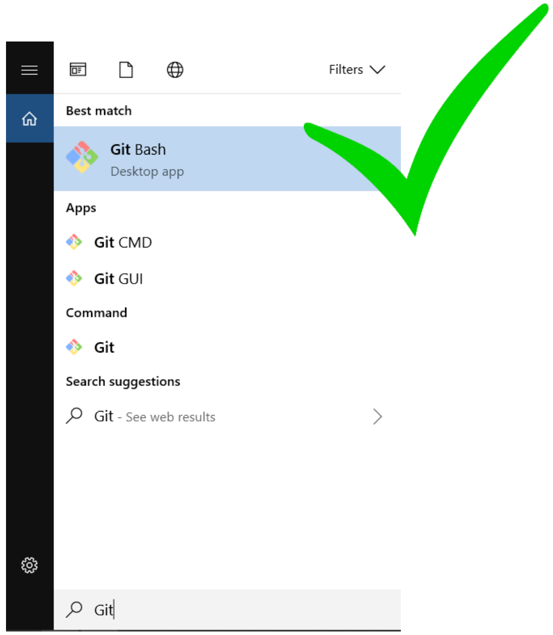
Verify Whether You Have Git Client

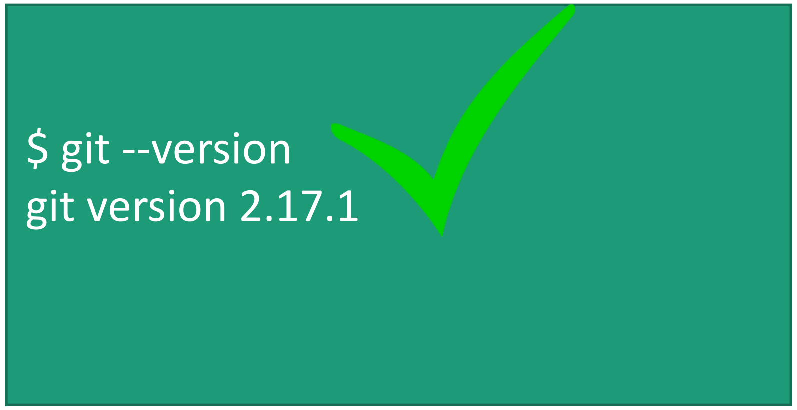
- Verify if you have had the Git client installed already
- Windows
  - Attempt to run "Git Bash"
- Unix (OS X or Linux):
  - Open a terminal window
  - Run "git --version", i.e., type "git --version" (without quotes) and hit the Enter key



Have I Had Git Client Installed?

- Windows and Unix



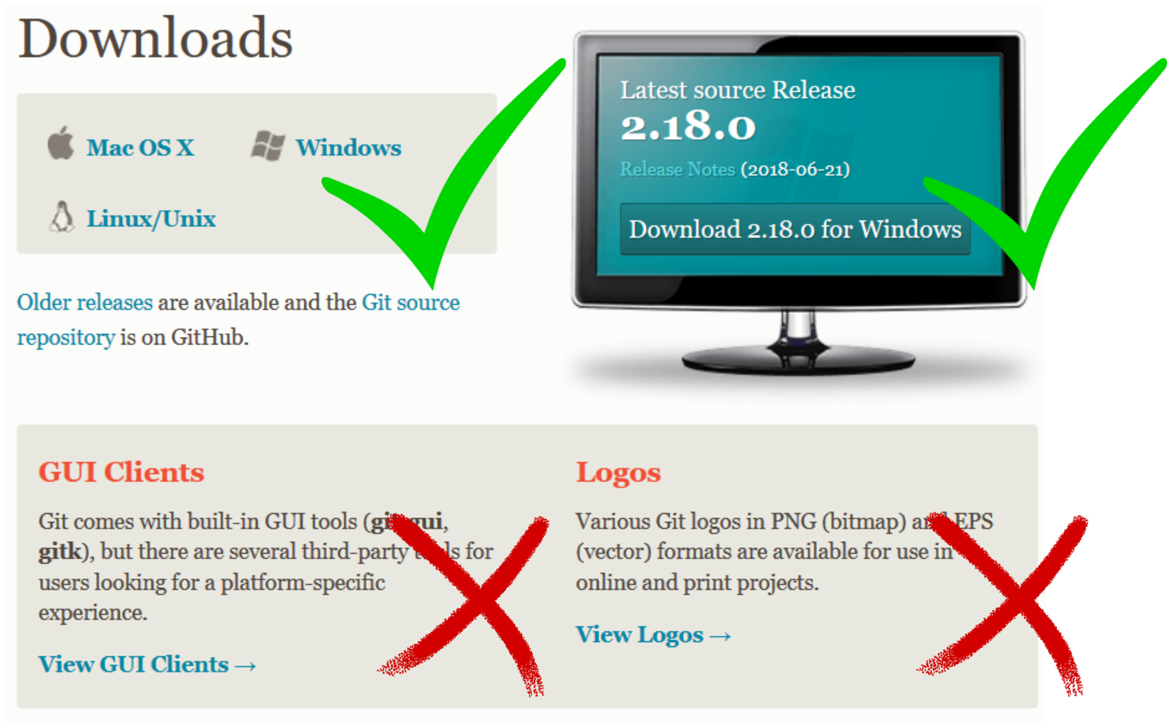


```
$ git --version
git version 2.17.1
```

- If not, download and install it

Download Git Client

- Visit https://git-scm.com/downloads using your favorite Web browser



Downloads

Mac OS X Windows

Linux/Unix

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.18.0
Release Notes (2018-06-21)

Download 2.18.0 for Windows

GUI Clients

Git comes with built-in GUI tools (git-gui, gitk), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

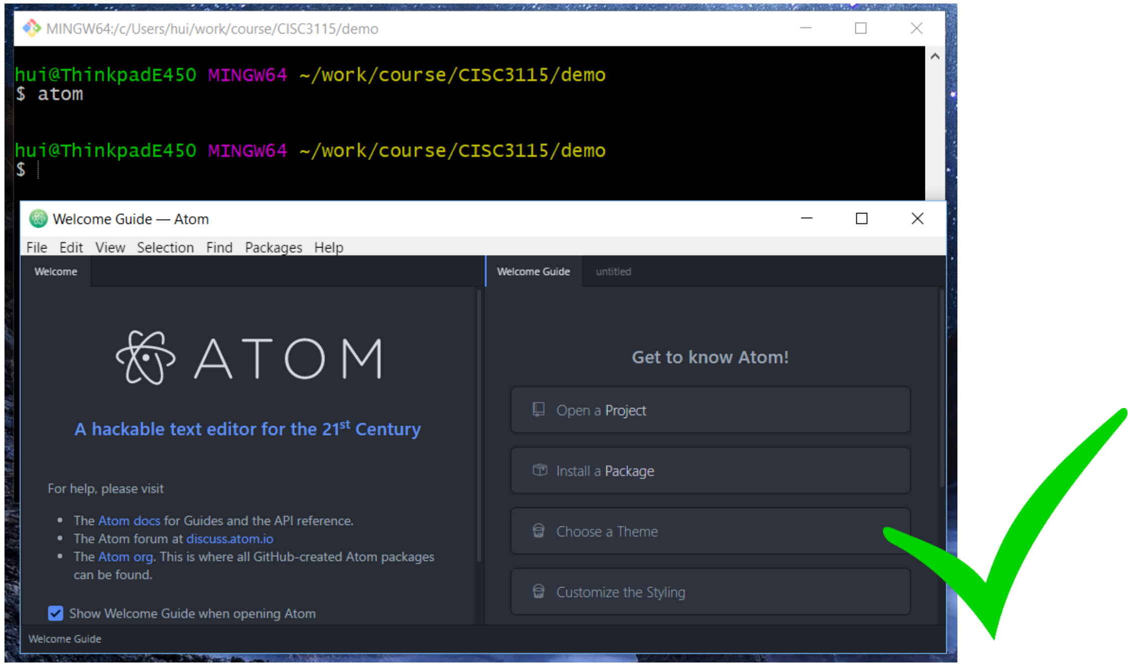
View Logos →

Git Bash on Windows

- Provides a terminal where you can run Unix commands
- The instructor shall use the Git Bash from now on so that the instructions are identical to both Windows and Unix (e.g., OS X) users
- Window users: Use the Git Bash terminal
- Unix users: just use a terminal (e.g., the terminal on OS X)

Verify Whether You Have Atom Installed

- Verify if you have had the Atom editor installed already
  - Type atom on the Command Line



```
MINGW64:/c/Users/hui/work/course/CISC3115/demo

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ atom

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$
```

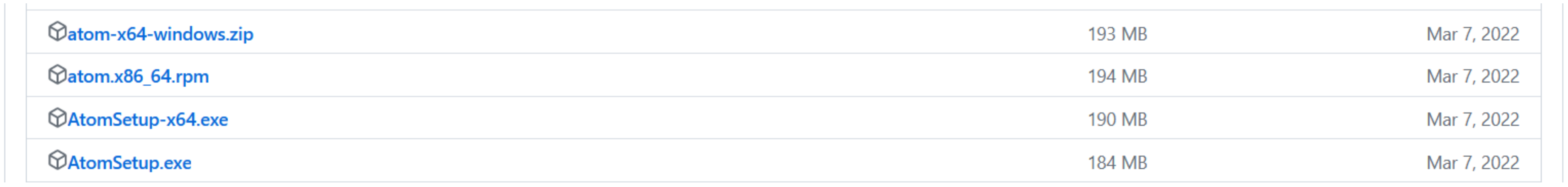
For help, please visit

- The Atom docs for Guides and the API reference.
- The Atom forum at discuss.atom.io
- The Atom org. This is where all GitHub-created Atom packages can be found.

✓ Show Welcome Guide when opening Atom

Download and Install the Atom Editor

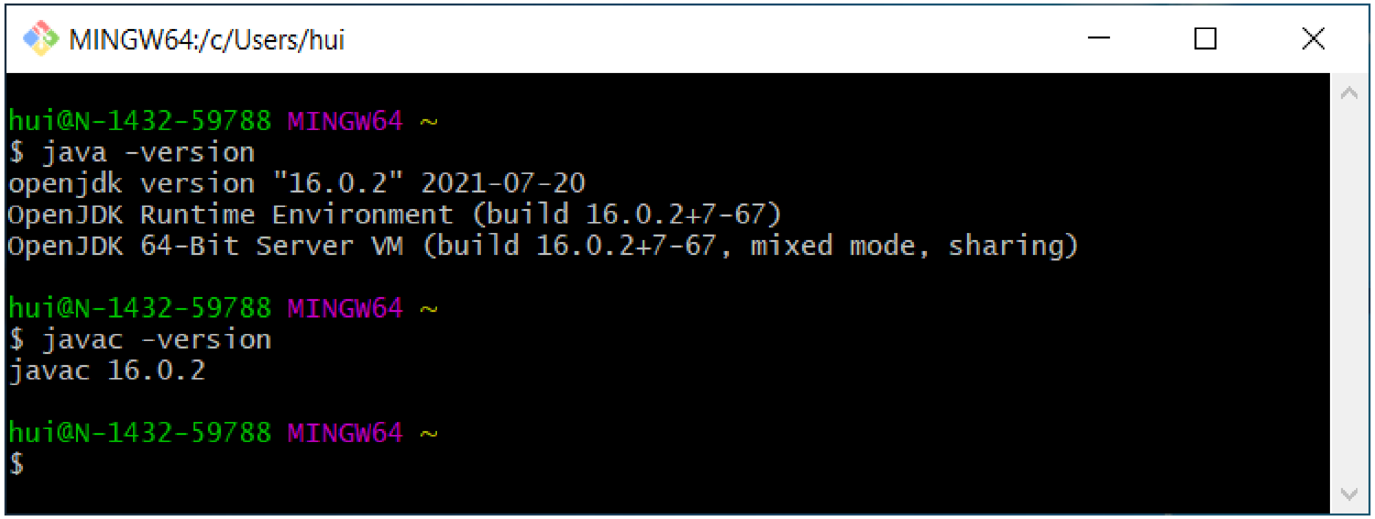
- If you have not had the Atom Editor installed, download and install the Atom editor
- Visit
  - https://github.com/atom/atom/releases/tag/v1.60.0



atom-x64-windows.zip	193 MB	Mar 7, 2022
atom.x86_64.rpm	194 MB	Mar 7, 2022
AtomSetup-x64.exe	190 MB	Mar 7, 2022
AtomSetup.exe	184 MB	Mar 7, 2022

Checking on Java and Javac

- Check whether both java & javac are found, and have an identical version. Otherwise, next slide.



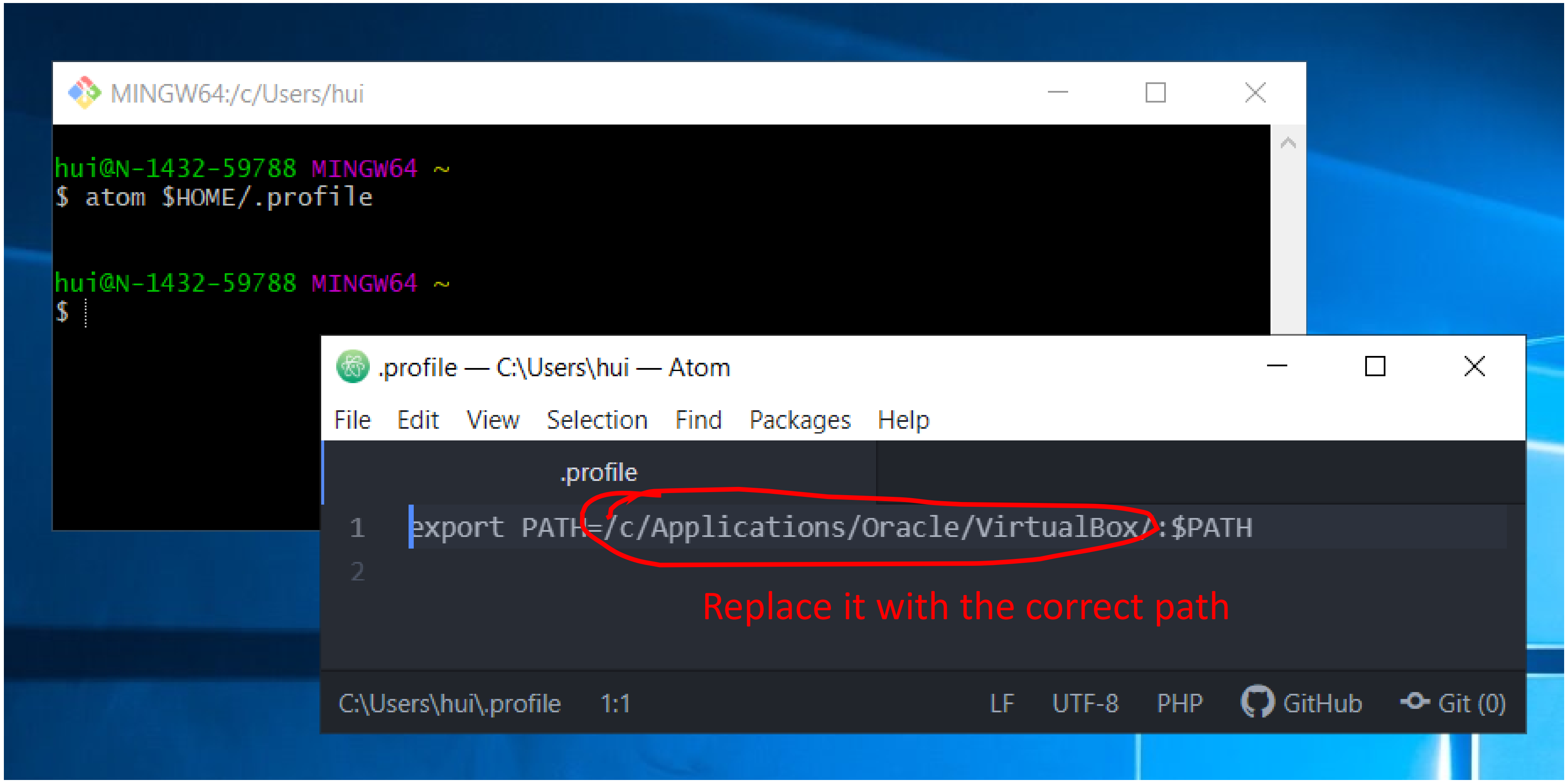
```
hui@N-1432-59788 MINGW64 ~
$ java -version
openjdk version "16.0.2" 2021-07-20
OpenJDK Runtime Environment (build 16.0.2+7-67)
OpenJDK 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)

hui@N-1432-59788 MINGW64 ~
$ javac -version
javac 16.0.2

hui@N-1432-59788 MINGW64 ~
$
```

- Based on the screenshot, did I have the required version of JDK set up?

Setting up Search Path for Java and Javac

- Sometimes we also need to do this step:
  - In “Git Bash” terminal, create (if not alrady exists) or edit the .profile file on your “home directory” (see next slide)
  - Then, restart “Git Bash” terminal, and check accessibility and versions of Java and Javac

Edit/Create .profile File



```
hui@N-1432-59788 MINGW64 ~
$ atom $HOME/.profile

hui@N-1432-59788 MINGW64 ~
$
```

.profile — C:\Users\hui — Atom
File Edit View Selection Find Packages Help
.profile

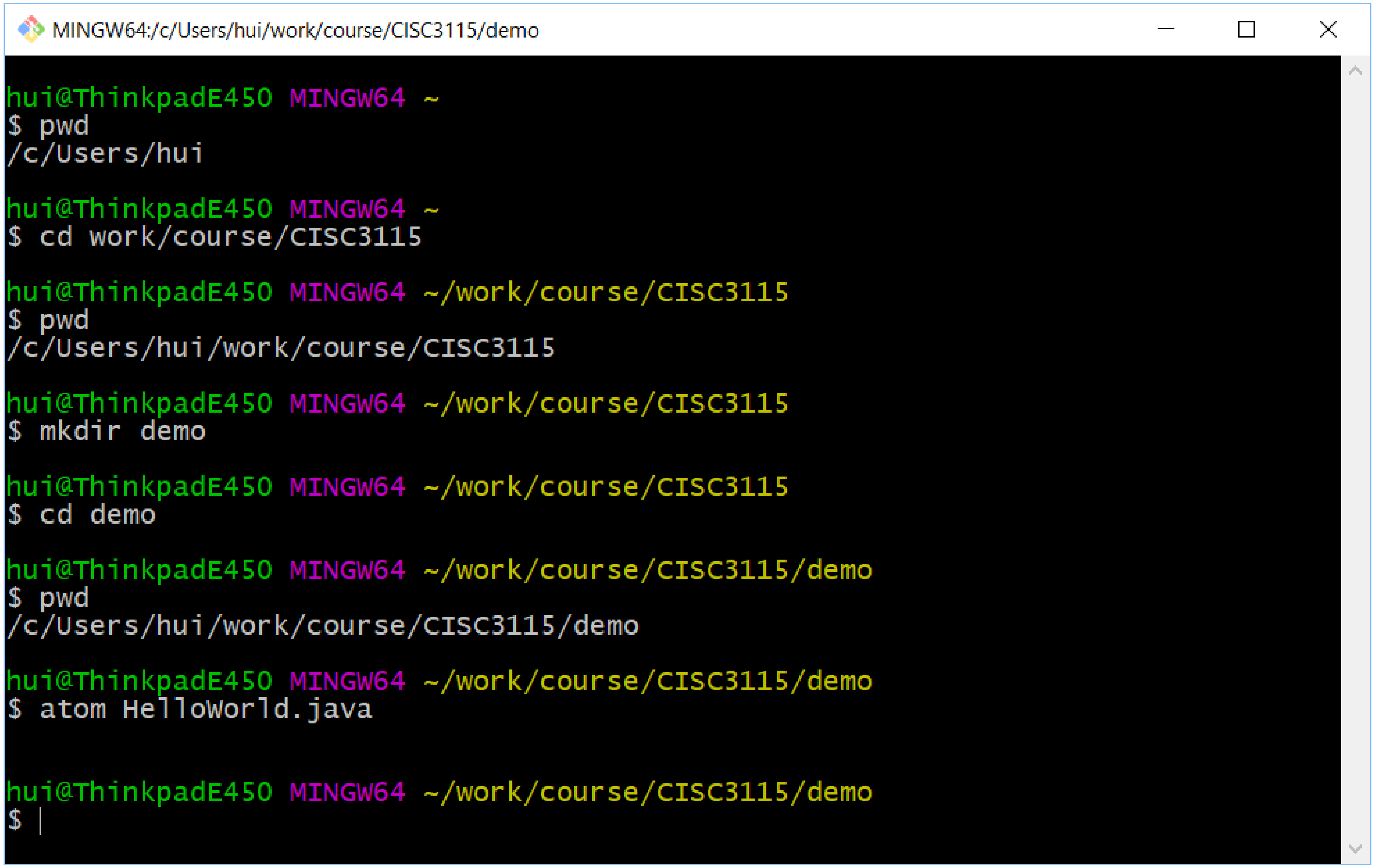
```
export PATH=/c/Applications/Oracle/VirtualBox/:$PATH
```

Replace it with the correct path

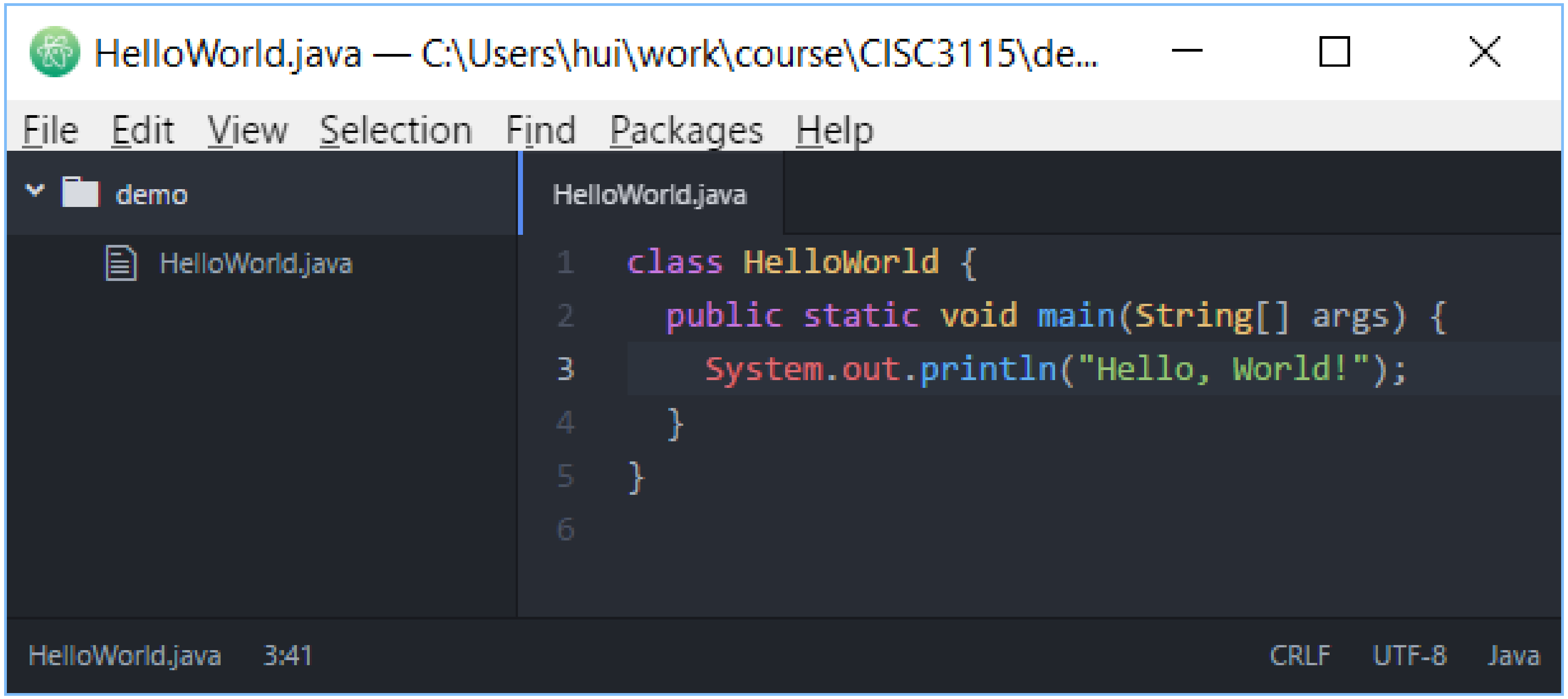
C:\Users\hui\.profile 1:1 LF UTF-8 PHP GitHub Git (0)

Implement the HelloWorld Java Program

- Open a terminal Window
- (Optional) Create a subdirectory under a desired directory
- Run “atom HelloWorld.java” from the Command Line at the subdirectory
- Type the code
- Save the file


MINGW64:/c/Users/hui/work/course/CISC3115/demo

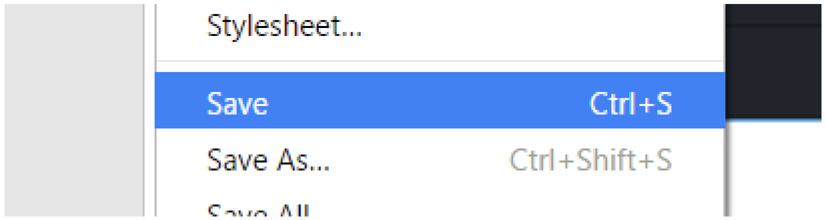
```
hui@ThinkpadE450 MINGW64 ~
$ pwd
/c/Users/hui

hui@ThinkpadE450 MINGW64 ~
$ cd work/course/CISC3115

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115
$ pwd
/c/Users/hui/work/course/CISC3115

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115
$ mkdir demo

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115
$ cd demo

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ pwd
/c/Users/hui/work/course/CISC3115/demo

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ atom HelloWorld.java


hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ |
```


HelloWorld.java — C:\Users\hui\work\course\CISC3115\de...
File Edit View Selection Find Packages Help
demo
HelloWorld.java
HelloWorld.java

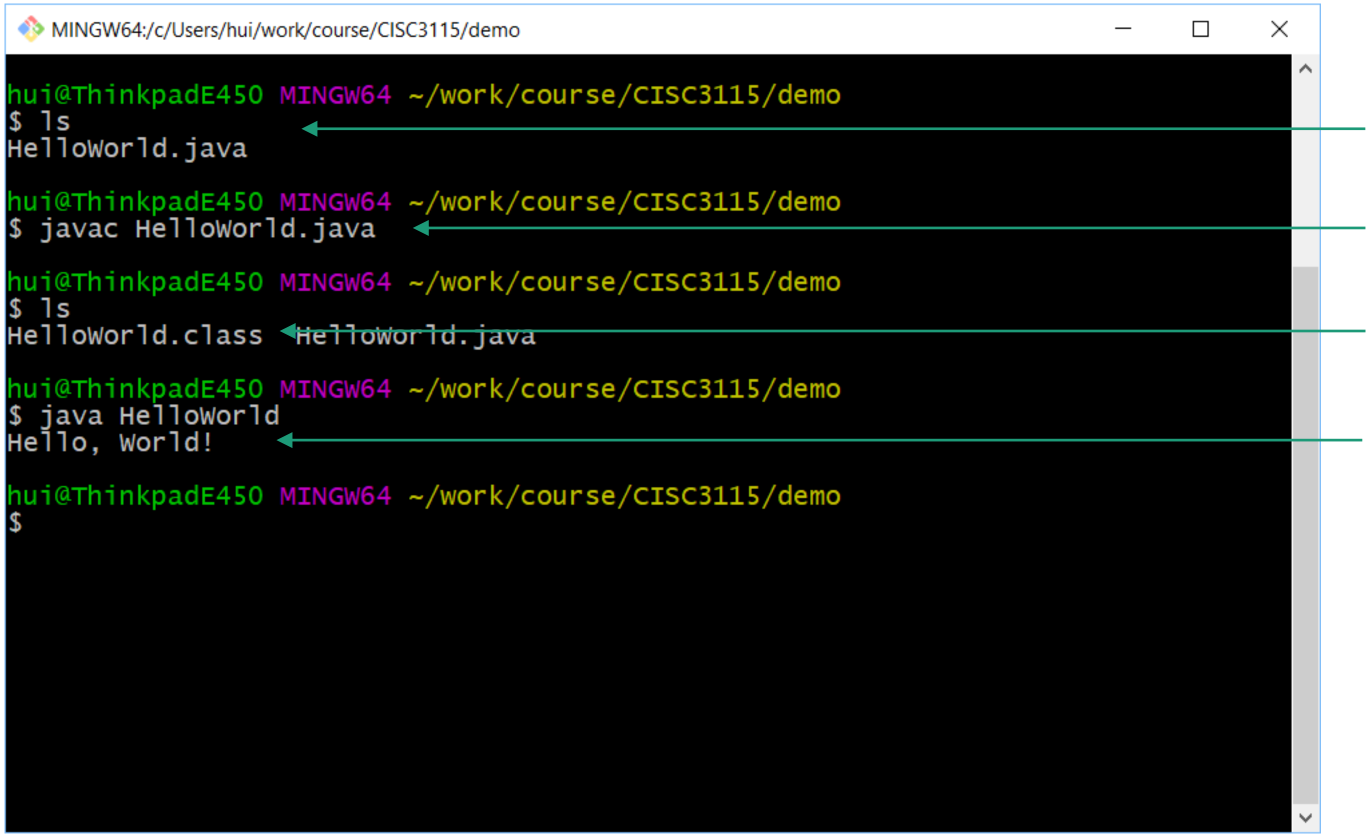
```
class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

HelloWorld.java 3:41
CRLF UTF-8 Java

- Press “CTRL-S” or click “Save” from the “File” menu to save the file



Compile and Run the Program


MINGW64:/c/Users/hui/work/course/CISC3115/demo

```
hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ ls
HelloWorld.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ javac HelloWorld.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ ls
HelloWorld.class  HelloWorld.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ java HelloWorld
Hello, World!

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$
```

Verify the program file exists

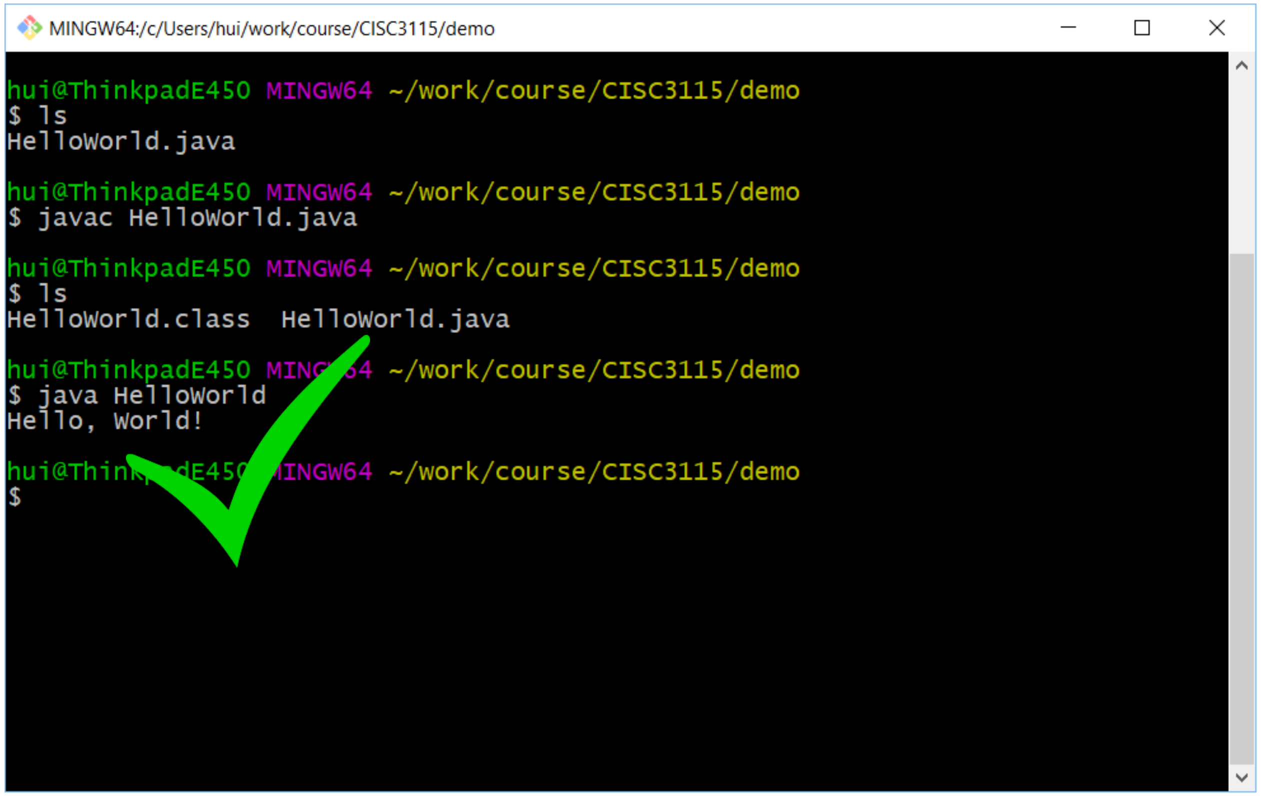
Compile the program

Verify the class file was created

Run the program

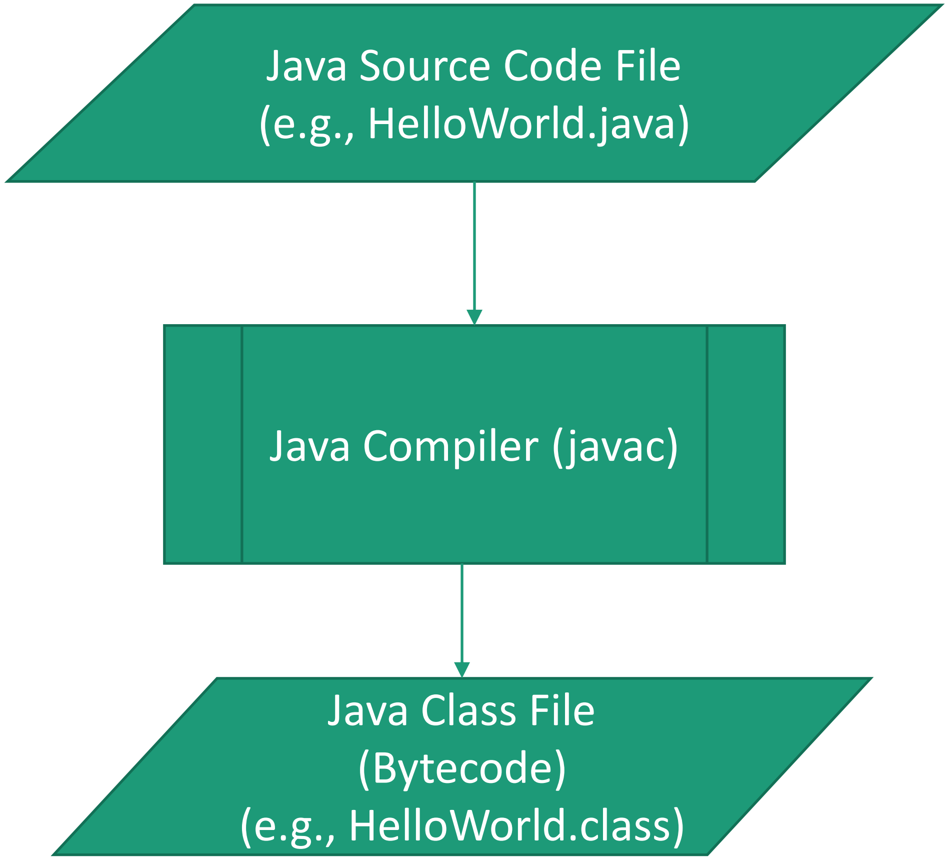
Verification

- Do I see “Hello, World!” when I run the program?


MINGW64:/c/Users/hui/work/course/CISC3115/demo

```
hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ ls
HelloWorld.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ javac HelloWorld.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ ls
HelloWorld.class  HelloWorld.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ java HelloWorld
Hello, World!

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$
```

Compilation



Running Java Program

- You are running Java class files containing Java bytecode
- Example: java HelloWorld
  - The java program launches a Java Virtual Machine (JVM)
  - load the HelloWorld.class (and its dependencies), and start executing the bytecode in the class files

Troubleshooting

- Read the compilation error message carefully
  - Caveat:
    - The error message is often inaccurate about what went wrong.
    - The compiler is more accurate at pinpointing where an error was found than telling what went wrong.
- Figure out what might be wrong, revise and compile it again
- Best practice: save often, compile often, don’t have to wait.

Questions

- Prepare the environment to write Java programs
  - Git and Git Bash
  - Atom (or other your favorite editors)
  - In this class, the instructor prefer not to use an Integrated Developer Environment software (IDE, e.g., Net Beans, Eclipse, IntelliJ)
- Review the process of authoring a simple Java program

In-Class Exercise

- Verify you have git client. If not, install it
- Verify you have Atom. If not, install it
- Create a folder C0831 in the journal directory
- In C0831, Create, compile and run the HelloWorld Java program
- Copy HelloWorld.java to HelloTeam.java, and revise “HelloTeam.java”, and let it print “Hello, Team!” instead
- Compile and run the HelloTeam.java
- If you haven’t encountered any compilation error, introduce one
  - Examples:
    - Misspell “class”, “main” etc deliberately, compile and observe error message
    - Remove a “;” deliberately, compile and observe error message
    - Remove a parenthesis, i.e., (or), or a brace, i.e., { or } deliberately, compile and observe error message

Questions?

- Write, compile, and run Java programs
- Remove compilation errors