

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

# 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

- Requirement: write a shortest java program, and compile and run it.
- Design: a Java program that prints out “Hello, World!” on a terminal window
- Implement and test
  - Create a HelloWorld.java using an editor
    - Recommend: the Atom editor, the Visual Studio Code, notepad++ for Windows; SlickEdit (\$\$\$) for Mac OS X
    - The instructor will use Atom for demo in class, and recommend strongly that you use Atom.
  - Compile and run the program
  - Test the program

# Demo for Authoring a Java Program

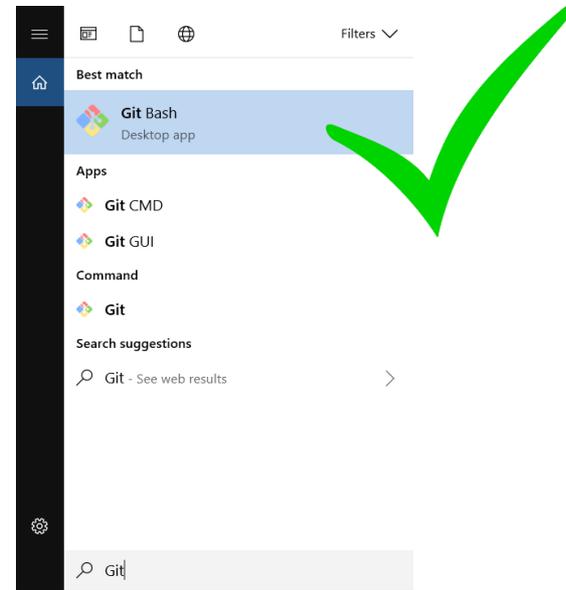
1. Prepare the working environment
  - a) Install the git client (if not already installed)
  - b) Install the Atom editor (if not already installed)
2. Create HelloWorld.java using the Atom editor
3. Compile and run the program

# Prepare the Working Environment

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

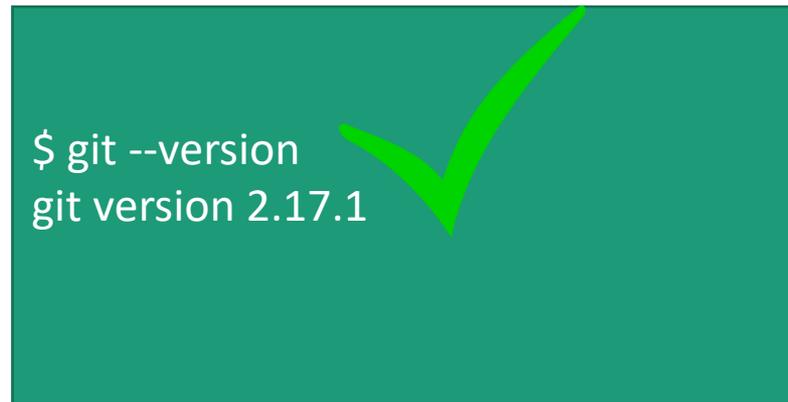
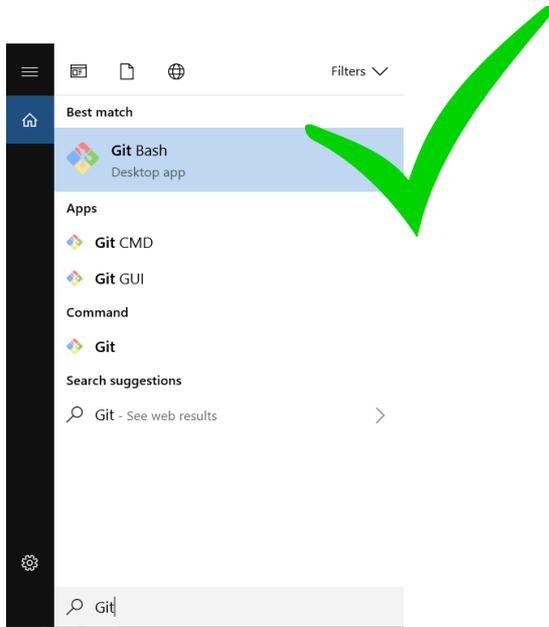
# Verify Whether You Have Git Client

- 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



- If not, download and install it

# Download Git Client

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

The screenshot shows the 'Downloads' section of the Git website. It features three operating system options: Mac OS X, Windows, and Linux/Unix, each with a green checkmark. To the right is a monitor displaying the latest source release '2.18.0' and a 'Download 2.18.0 for Windows' button, also with a green checkmark. Below these are two sections: 'GUI Clients' and 'Logos', both of which have a large red X mark over them. The 'GUI Clients' section mentions built-in tools like 'git-gui' and 'gitk', and provides a link to 'View GUI Clients'. The 'Logos' section mentions various Git logos in PNG and EPS formats and provides a link to 'View Logos'.

**Downloads**

Mac OS X Windows Linux/Unix

Latest source Release  
**2.18.0**  
Release Notes (2018-06-21)  
Download 2.18.0 for Windows

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

**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 →](#)

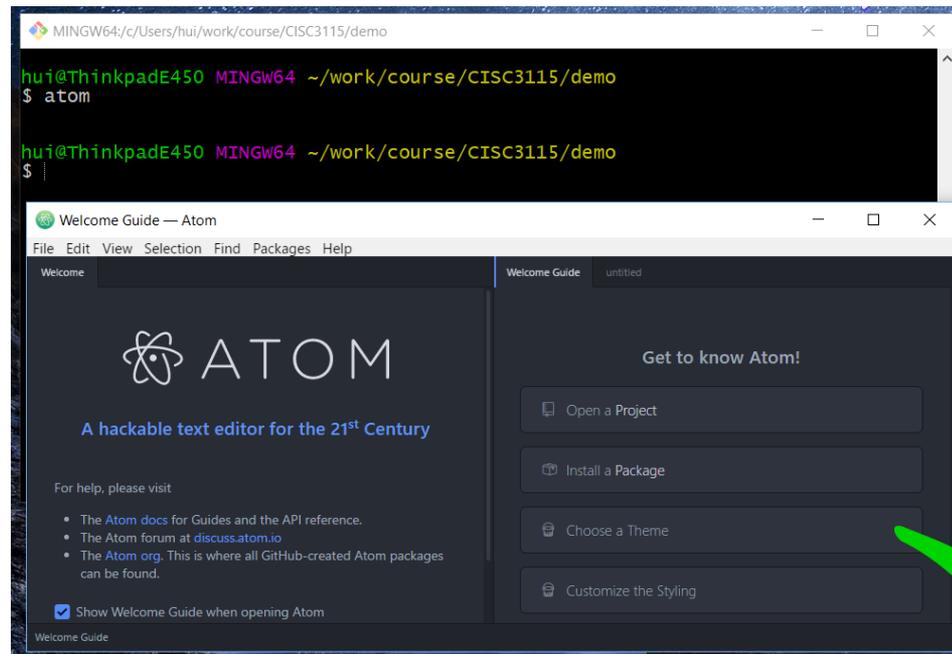
**Logos**  
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.  
[View Logos →](#)

# 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)

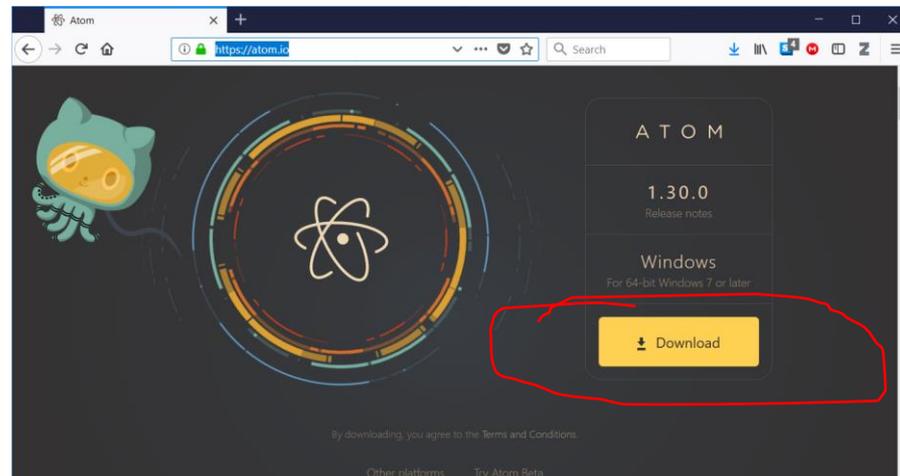
# Verify Whether You Have Atom Installed

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



# Download and Install the Atom Editor

- If you have not had the Atom Editor installed, download and install the Atom editor
- Visit <https://atom.io/> using your favorite Web browser



# 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

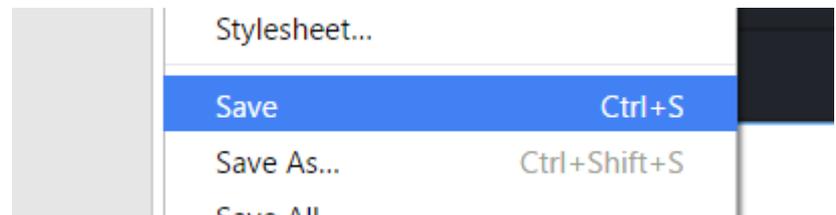
hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo
$ |
```

A screenshot of an IDE window titled "HelloWorld.java — C:\Users\hui\work\course\CISC3115\de...". The window has a menu bar with "File", "Edit", "View", "Selection", "Find", "Packages", and "Help". On the left, a file explorer shows a "demo" folder containing "HelloWorld.java". The main editor area shows the following Java code:

```
1 class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
6
```

The status bar at the bottom indicates "HelloWorld.java 3:41" and "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

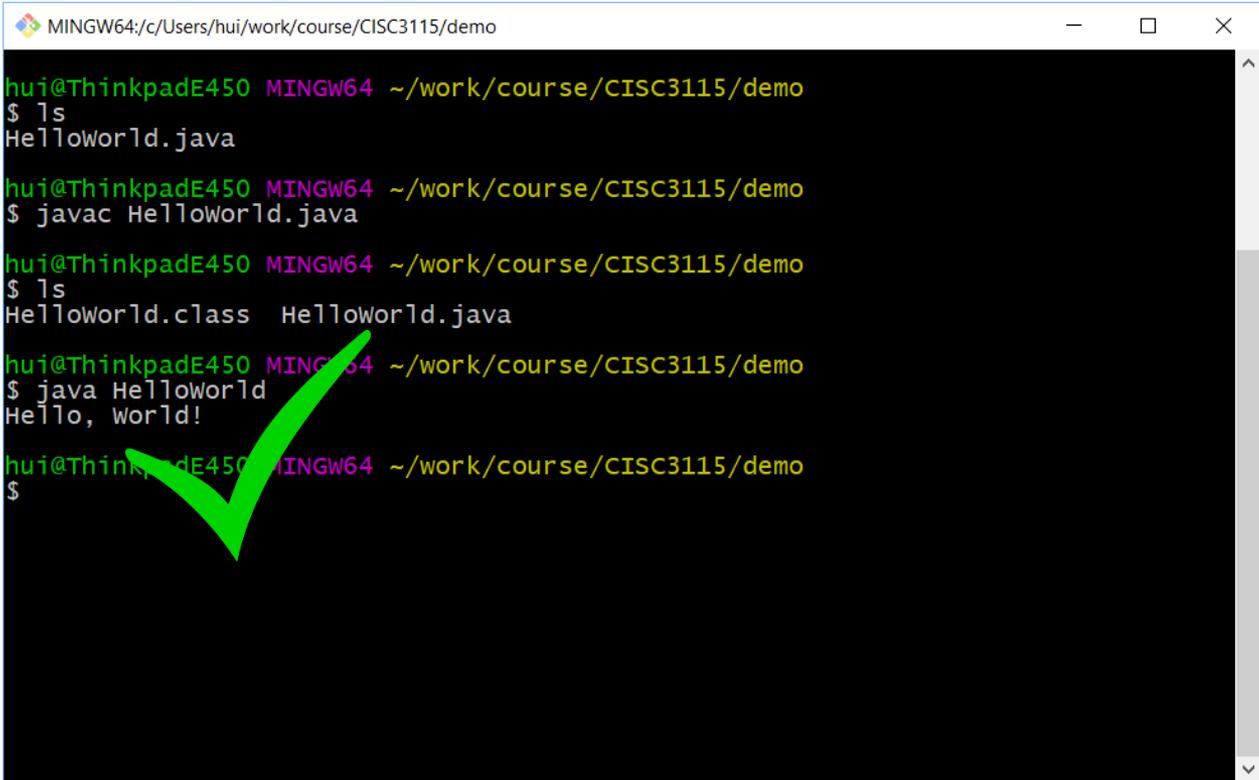
Compile the program

Verify the class file was created

Run the program

# Verification

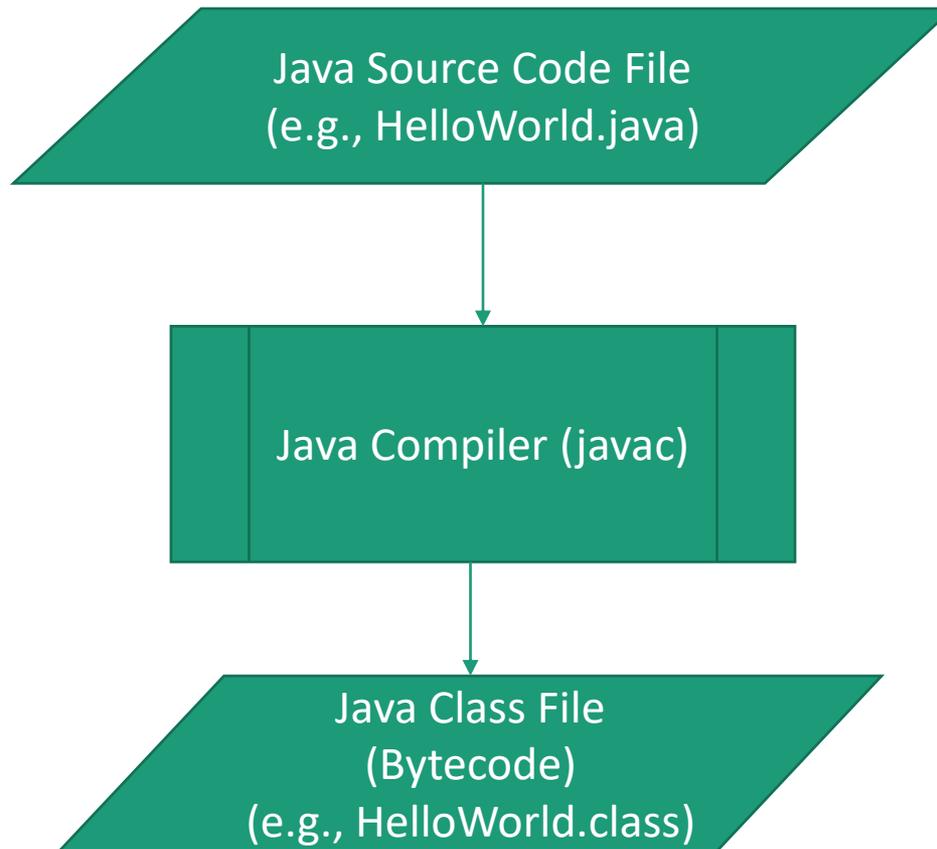
- Do I see “Hello, World!” when I run the program?



```
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
$
```

A terminal window with a black background and white text. The window title is "MINGW64:/c/Users/hui/work/course/CISC3115/demo". The terminal shows the following sequence of commands and output:   
1. `hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo`  
2. `$ ls` followed by `HelloWorld.java`  
3. `hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo`  
4. `$ javac HelloWorld.java`  
5. `hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo`  
6. `$ ls` followed by `HelloWorld.class HelloWorld.java`  
7. `hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo`  
8. `$ java HelloWorld` followed by `Hello, world!`  
9. `hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/demo`  
10. `$`  
A large green checkmark is drawn over the output "Hello, world!".

# 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 sometimes is accurate about what went wrong; sometimes not.
    - 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

# Exercise

- This is a group exercises. You must record this in the class journal.
- Verify you have git client. If not, install it
- Verify you have Atom. If not, install it
- Create a folder C0901 in the journal directory
- In C0901, 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

# Pushing the Journal to Github

- A student volunteer will make the demo following the instructor's instructions.

# Exercise

- Students submit the class journal by pushing the journals to Github

# Question?

- What about new journal content/entry?

# CodeLab Registration and Assignment

- Course Section Code is in Blackboard
- Assignment 1