CISC 3115 MY3

Developing Simple Java Programs

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Authoring Java programs
- Compiling and running Java programs from command line
- CodeLab Registration

Developing Simple Java Programs

1. Understand *problem* to solve
2. Develop solution: design a Java program
3. Implement solution
    I. Create source code file
    II. Compile and test
4. Repeat 1 – 3 until we solve the problem

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 *problem* does the “customer” want to solve?
- 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

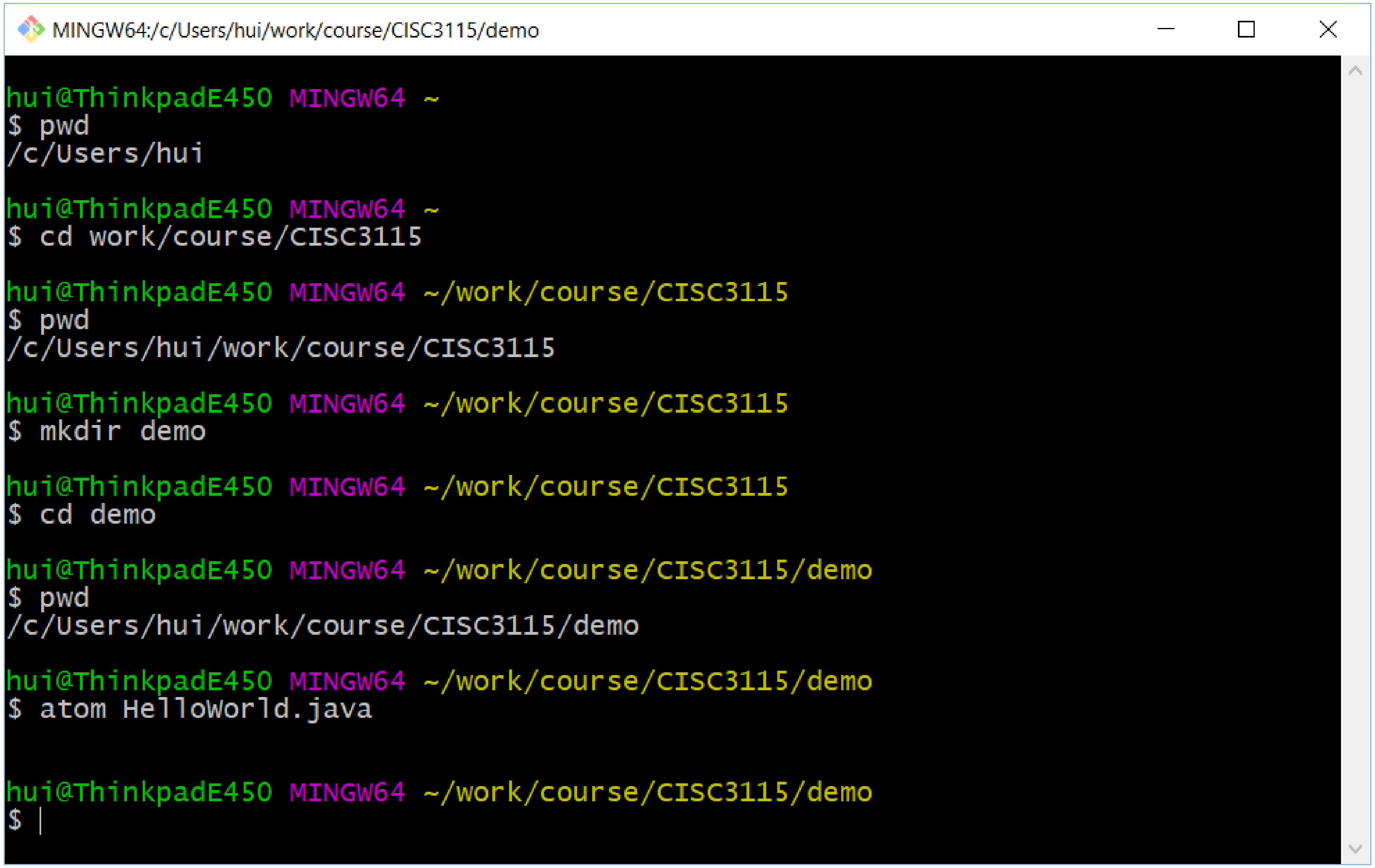
1. **Problem:**
   ____?
2. **Requirement**: write a shortest java program, and compile and run it.
3. **Design**: a Java program that prints out “Hello, World!” on the standard output
4. **Implement**
   - A. Create/Revise a HelloWorld.java ***source code file*** using an editor
     - The instructor will use Atom for demo in class.
   - B. Compile the program (the source code files) into the ***Java bytecode*** files, if error, go to step A
5. **Test**
   - Test the program, if failed, go to step 2 (can also be steps 1 and 3)

Demo for Authoring a Java Program

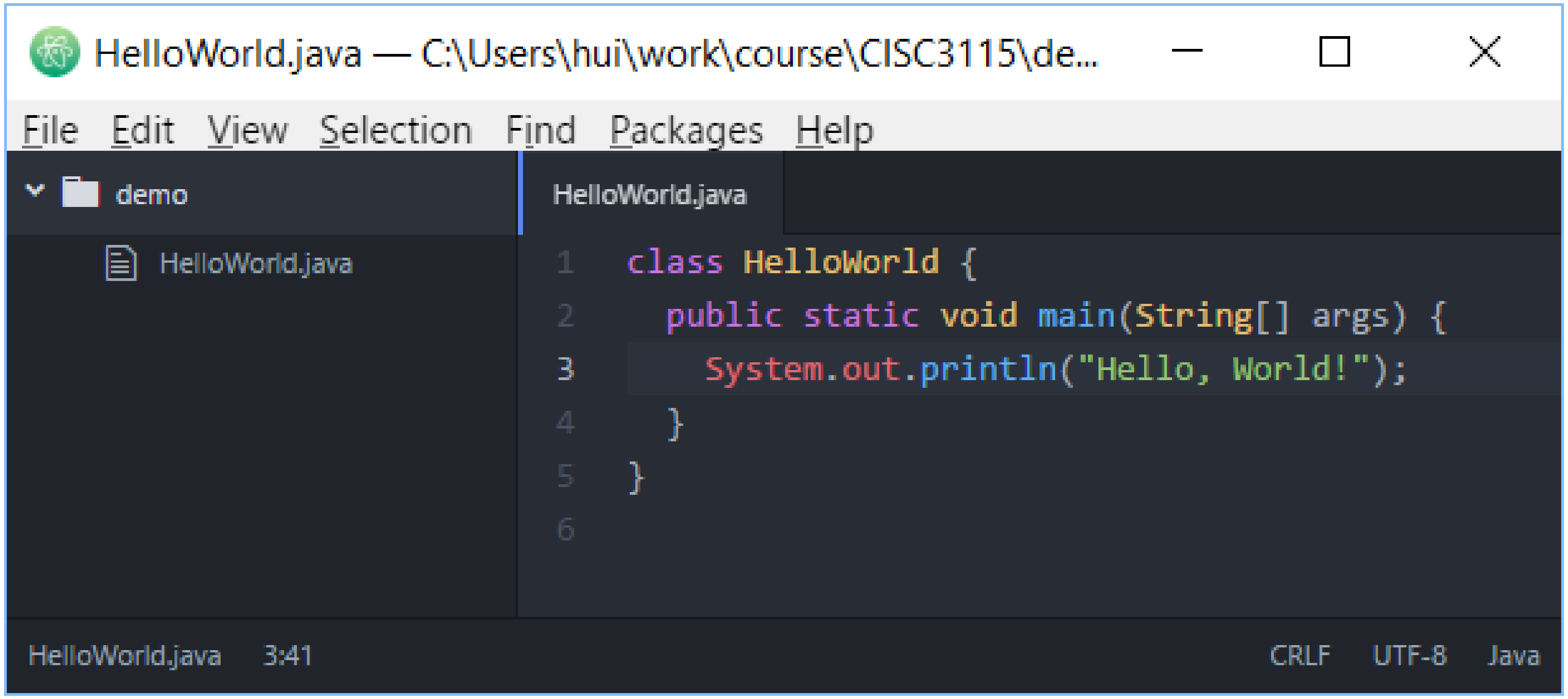
1. Prepare the working environment
   a) Install the git client (if not already installed)
   b) Install the Atom editor (if not already installed)
2. Create HelloWorld.java using the Atom editor
3. Compile the program
4. Test the program

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

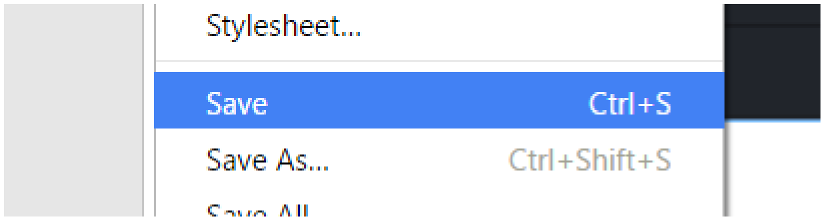
```
hui@ThinkpadE450 MINGW64 ~
$ pwd
/c/Users/hui

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

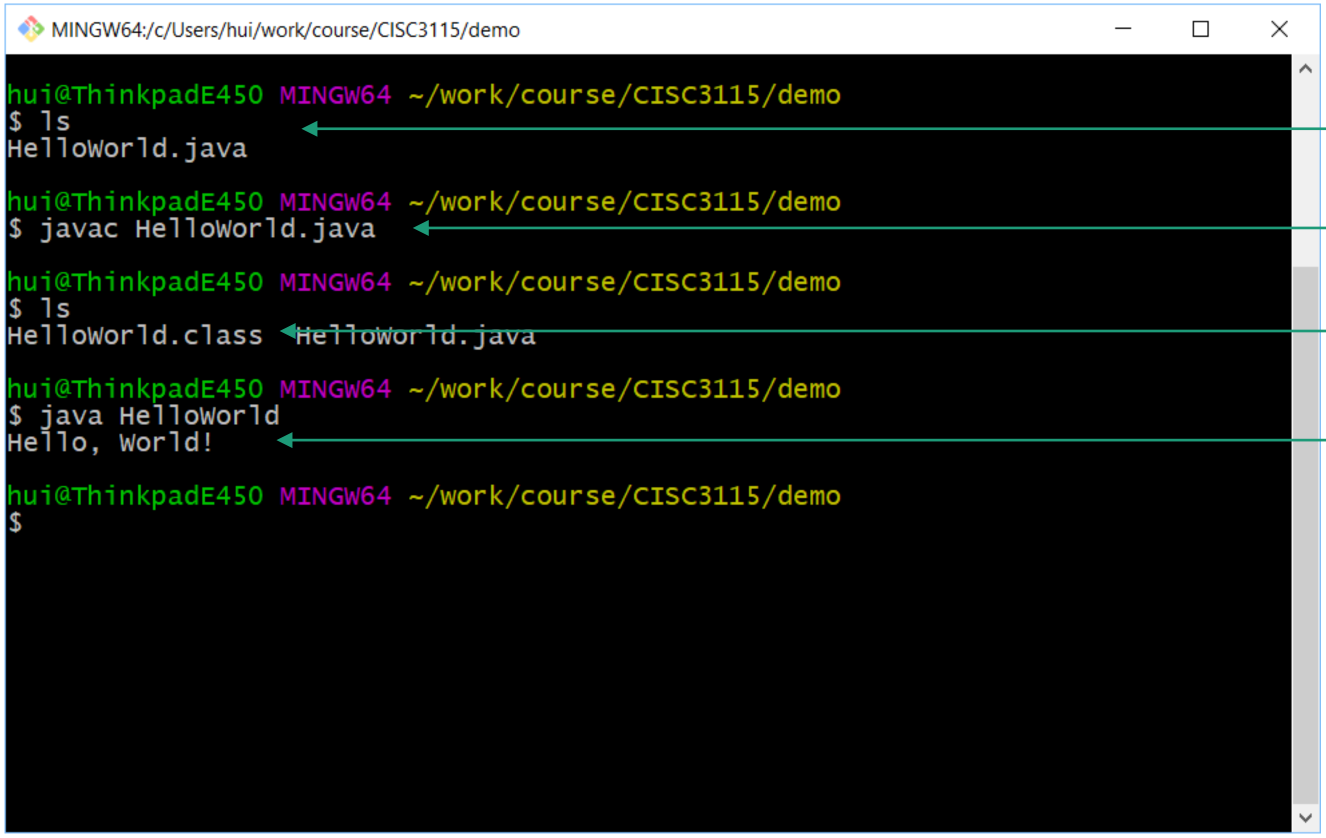
```
class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

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

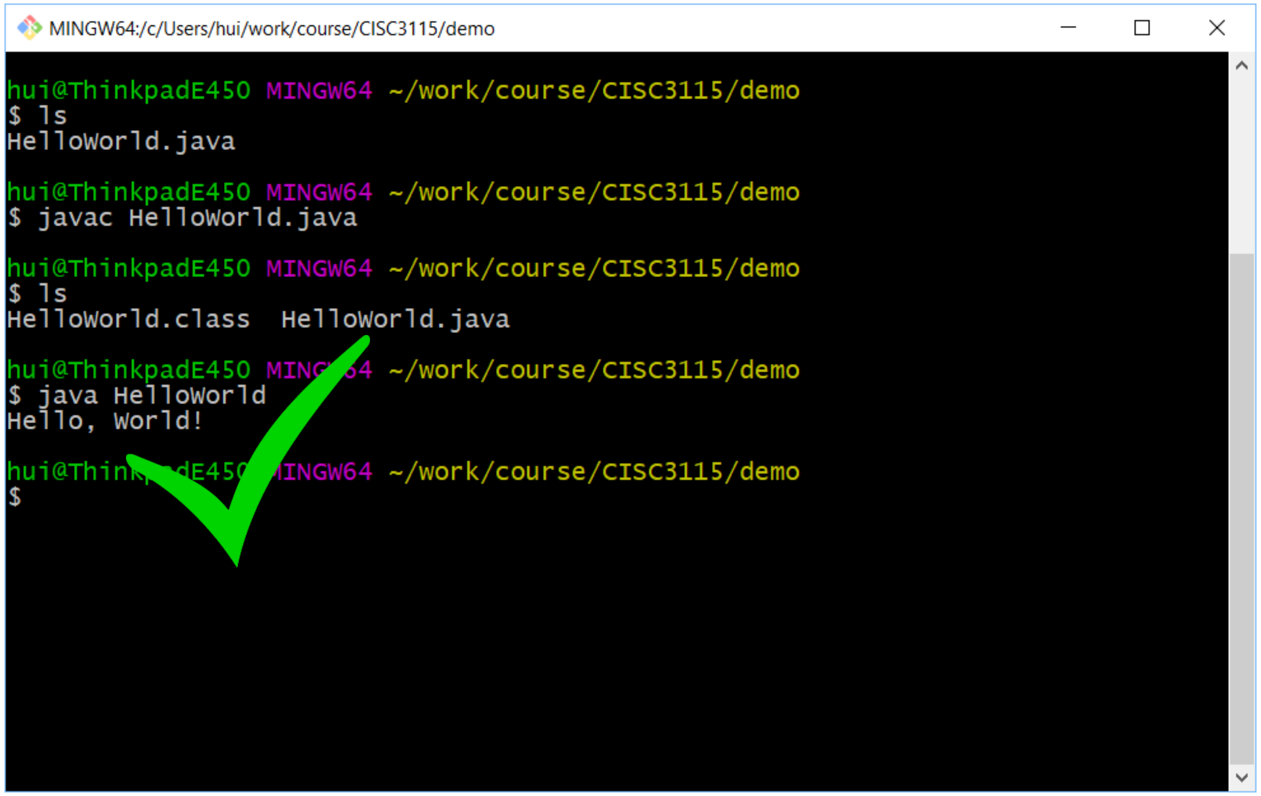
Compile the program

Verify the class file was created

Run the program

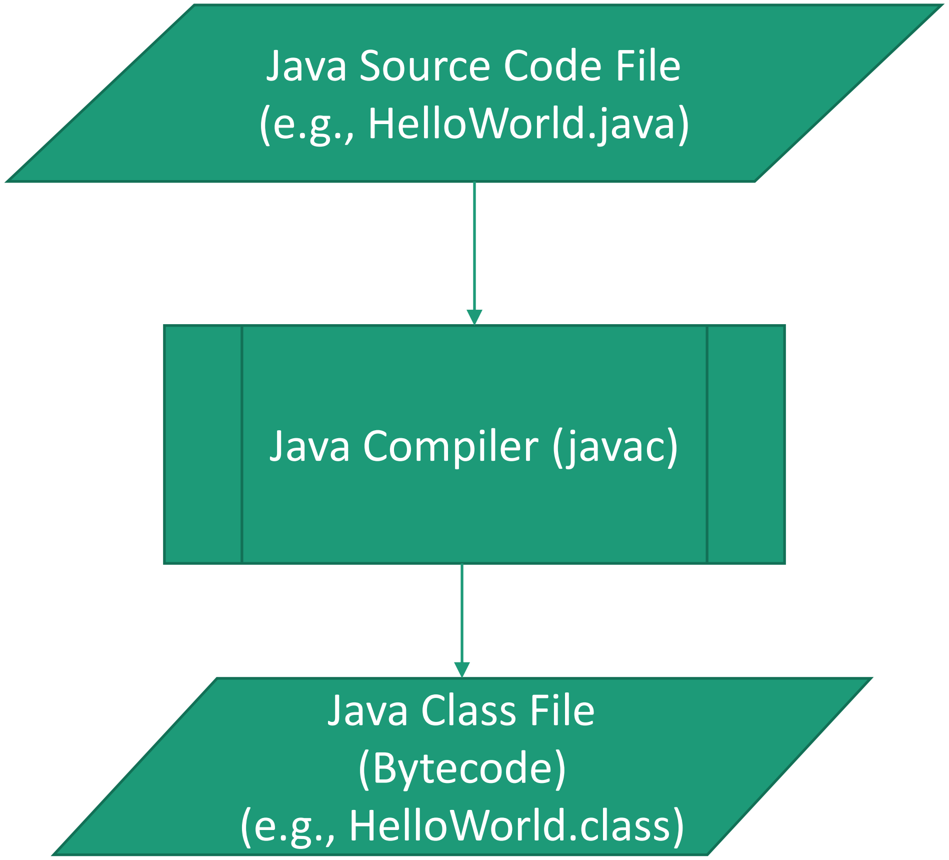
Verification

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



```
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 coding environment to solve computational problems by writing 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
- Ger organized and create a folder/directory for this exercise
- In the folder/directory, 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
- But, for what purpose as a computer scientist?