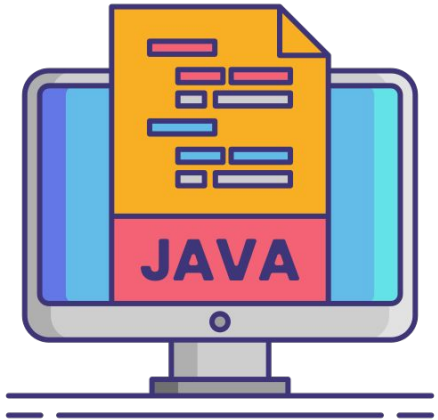
CISC 1115: Introduction to Programming Using Java

Summer 2025 Prep Workshop



But First: Connecting to BC WiFi

- In order to use the Brooklyn College wireless network, you must follow these steps:

- **Windows PC:**
  - Download & Install **the SecureW2 Program**
  - Find the network name (SSID): **BC-WiFi** and connect to it
  - Configure through the W2 Program by putting in **your username (EMPL ID) and Password* (FLMM/DD/YY)**

- **ios/macOS:**
  - Find the network name (SSID): **BC-WiFi** and connect to it
  - Configure it by putting in **your username (EMPL ID) and Password* (FLMM/DD/YY)**

Itinerary

01 Introduction to Java

02 Downloading and Installing the JDK

03 Writing Your First Java Program (Part 1 - Using the CLI)

04 Introduction to IDEs

05 Writing Your First Java Program (Part 2 - Using the IDE)

06 Overview and Next Steps

01

Introduction to java

What is Java?

- **Java** is a high-level, class-based, popular **object-oriented programming language**, created in 1995.
- It is owned by Oracle, and more than 3 billion devices run Java.
- The rules and syntax of Java are based on the C and C++ languages.
- It is used for:
  - Mobile applications (specially Android apps)
  - Desktop and Web applications
  - Web servers and application servers
  - Games
  - Database connection
  - And much, much more!

Why use Java?

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the most popular programming languages in the world
- It has a large demand in the current job market
- It is easy to learn and simple to use
- It is open-source and free
- It is secure, fast and powerful
- It has huge community support (tens of millions of developers)
- Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa

Getting Started With Java

- To create an application using Java, you will need:
  - to **download and install the Java Development Kit (JDK)**, which is available for Windows, macOS, and Linux.

  - A **command-line interface (CLI)** on your computer that allows you to create and delete files, run programs, and navigate through folders and files **(On a Mac, it's called Terminal, and on Windows, it's Command Prompt)** OR (preferably) **an integrated development environment (IDE)** - a software application that helps programmers develop software code efficiently by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application, such as Eclipse, NetBeans, IntelliJ, DrJava, etc.

02

Downloading and Installing the JDK (Java Development Kit)

What is the JDK?

- The **Java Software Development Kit (JDK)** is a set of tools that are used to develop Java applications. It includes everything you need to compile, debug, and run Java programs.

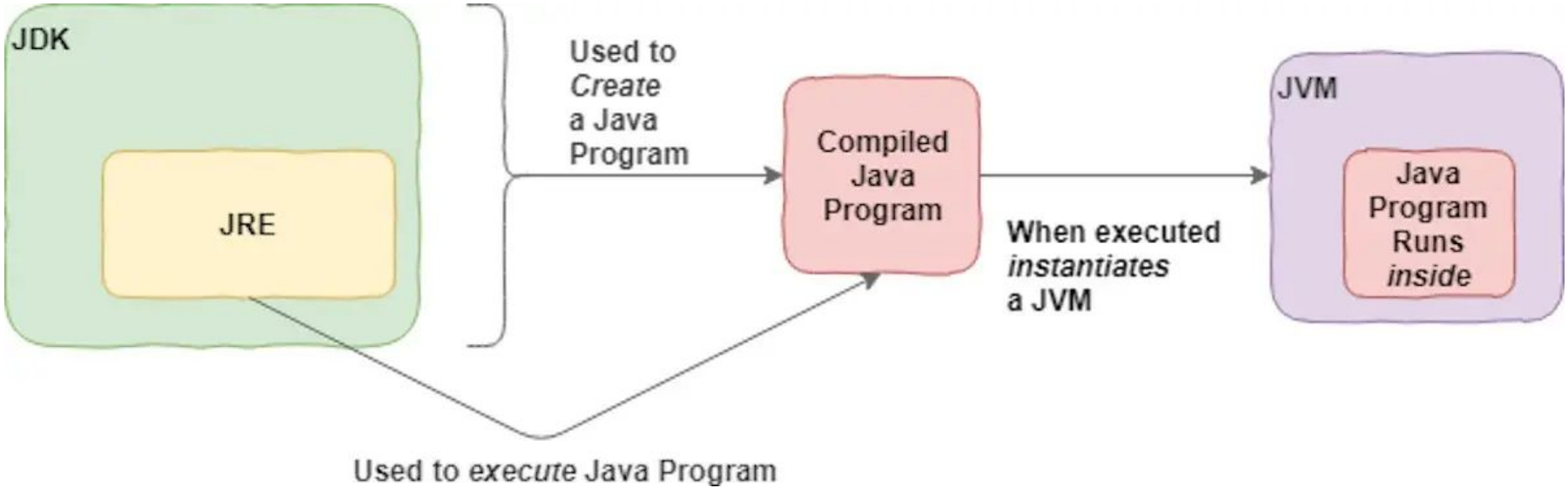
- The JDK is an essential tool for Java programmers as it **provides the necessary libraries, executables, and tools to develop Java applications**.

- It includes:
  - the **Java Runtime Environment (JRE)**
  - **compiler (javac)**
  - **interpreter (java)**
  - and **other tools needed for Java development**

What is the JDK?

- The **JDK** is one of three core technology packages used in Java programming, along with the **JVM (Java Virtual Machine)** and the aforementioned **JRE.**
- It's important to differentiate between these three technologies and understand how they're connected:
    - The **JVM is the runtime that hosts running programs.**
    - The **JRE is the on-disk part of Java that creates the JVM and loads programs into them.**
    - The **JDK provides the tools necessary to write Java programs that can be executed and run by the JVM and JRE.**
- Developers new to Java often confuse the Java Development Kit and the Java Runtime Environment. The distinction is that **the JDK is a package of tools for developing Java-based software**, whereas the JRE is a package of tools for running Java code.

The figure below shows how the JDK fits into the Java application development lifecycle:



Downloading the JDK

- The Oracle corporations own the current Java, and now it is commercially available. Yet, there is still a free java version available, which is called OpenJDK.
- **The current Java JDK version from Oracle corporation (as of March 2025) is the JDK 24.**
- JDK 22 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).
- **JDK 24 will receive updates under these terms, until September 2025, when it will be superseded by JDK 25. (So you may need to update in the future)**
- The JDK can be installed on the following Platforms:
  - Microsoft Windows
  - Linux
  - macOS

Downloading the JDK (Windows ver.)

- The very first step is to download the **Oracle Java Development Kit (JDK)** from the Official Oracle Website.
- For that, Head over to the Official Website: https://www.oracle.com/java/technologies/downloads/

- You need to identify your system specifications to choose the Product/file description. The website will contain the latest version for your corresponding system.

- **We will be downloading the latest x64 Installer of Java SE Development Kit 24.**

- After the download is complete, proceed to install the JDK by following the bootstrapped steps.

Downloading the JDK (Windows ver.)

- After the installation is complete, we have to **configure environment variables** to notify the system about the directory in which JDK files are located.
- Proceed to **C:\Program Files\Java\jdk-{YOUR_JDK_VERSION}\bin** (replace {-} with your JDK version) (this will serve as our path address)
- To set the Environment Variables, you need to search Environment Variables in the Task Bar and click on “Edit the system environment variables”.
- Under the Advanced section, Click on “Environment Variables”.
- Under System variables, select the “Path” variable and click on “Edit”. Click on “New” then paste the Path Address. Click on “OK”.
- Now, in the Environment Variables dialogue, under System variables, click on “New” and then under Variable name: JAVA_HOME and Variable value: paste address i.e. C:\Program Files\Java\jdk-{YOUR_JDK_VERSION}. Click on OK ⇒ OK ⇒ OK.

Downloading the JDK (Windows ver.)

- Now we can check the Java version installed, open Command Prompt and enter the following commands:

```
java -version
javac -version
```

- You should see the Java version installed, i.e. java version “”
- Congratulations! You have installed the JDK!

Downloading the JDK (macOS ver.)

- We will install the JDK similarly as we did on Windows only using the macOS equivalent.
- **For Mac, we'll be downloading the latest x64 DMG Installer of Java SE Development Kit 24.**
- After the download is complete, proceed to install the JDK by following the bootstrapped steps.

- Now to configure, we have to open the terminal and pass the following commands. To find the Location of the JAVA_HOME, run the following command (substituting the version i.e. -v24) :

```
/usr/libexec/java_home -v{YOUR_VERSION}
```

Downloading the JDK (macOS ver.)

- We have to **set this output as our JAVA_HOME Environment Variable.** You can use any command or code editor to edit the file, here we are using VS Code (but you can use any text editor (nano, vi, etc.)):

```
code ~/. bash_profile
```

- At the very bottom, we have to export the path we obtained earlier i.e.

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-{YOUR_VERSION}.jdk/Contents/Home
```

Downloading the JDK (macOS ver.)

- Now we have to refresh the environment file by using this command:

```
source ~/.bash_profile
```

- And echo the JAVA_HOME variable:

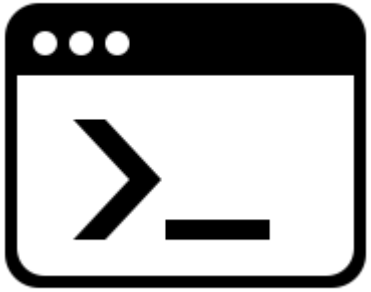
```
echo $JAVA_HOME
```

Downloading the JDK (macOS ver.)

- Now we can check the Java version installed, in the terminal, enter the following commands:

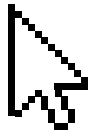
```
java -version
javac -version
```

- You should see the Java version installed, i.e. java version “”
- Congratulations! You have installed the JDK!



03

Writing Your First Java Program
(Part 1 - Using the CLI)



The Command-line Interface

- When you are starting programming, it may be simpler to use a command-line interface (CLI) for compiling/running.
- You can write Java programs with any text editor (program that lets you edit standard ASCII text files) **saving your file under a .java extension.**
- Unlike C++ programs, but like Scala programs, Java programs don't compile to code that can be executed directly. Instead **they compile to "bytecode" (.class files) meant to be executed by a Java virtual machine (JVM).**

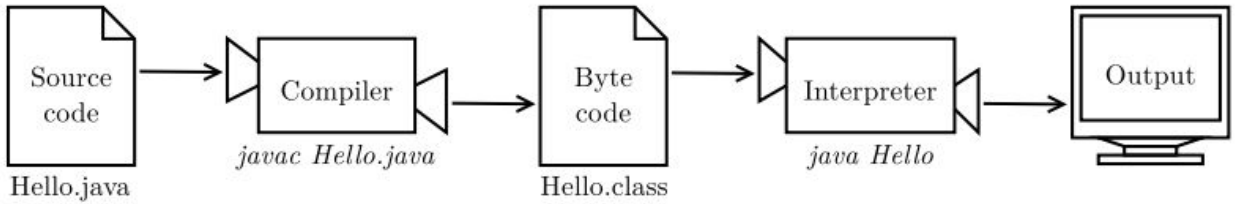


Figure 1.2: The process of compiling and running a Java program.

Writing Our First Program

- Once you have verified that Java is properly installed on your computer, **we can get a simple program running with just a text editor and a command-line interface (CLI) – On a Mac, it's called Terminal, and on Windows, it's Command Prompt**
- We break the process of programming in Java into three steps:
  a. **Create the program by typing it into a text editor and saving it to a file named, HelloWorld.java.**
  b. **Compile it by typing "javac HelloWorld.java" in the command-line window.**
  c. **Execute (or run) it by typing "java HelloWorld" in the command-line window.**
- The first step creates the program; the second translates it into a language more suitable for machine execution (and puts the result in a file named HelloWorld.class); the third actually runs the program.

Writing Our First Program

- **Creating a Java program** – A program is nothing more than a sequence of characters, like a sentence, a paragraph, or a poem.
- To create one, we need only define that sequence characters using a text editor in the same way as we do for email.
- Type the following into your text editor and save it into a file named HelloWorld.java:

```
public class HelloWorld {
   public static void main(String[] args) {
      // Prints "Hello, World" in the terminal window.
      System.out.println("Hello, World");
   }
}
```

Writing Our First Program

- **Compiling a Java program** – A compiler is an application that translates programs from the Java language to a language more suitable for executing on the computer.
- It **takes a text file with the .java extension as input (your program) and produces a file with a .class extension (the computer-language version).**
- To compile HelloWorld.java type the following text below into the command-line:

```
javac HelloWorld.java
```

- **If you typed in the program correctly, you should see no error messages.** Otherwise, go back and make sure you typed in the program exactly as it appears above.

Writing Our First Program

- **Executing (or running) a Java program –** Once you compile your program, you can execute it.
- This is the exciting part, where the computer follows your instructions.
- To run the HelloWorld program, type the following in the command-line window:

```
java HelloWorld
```

- If all goes well, you should see the following response:

```
Hello, World
```

How Does the "Hello, World!" Program Work?

- In Java, **every application begins with a class definition.**
- In this particular program, HelloWorld is the name of the class and **the name of the class should match the filename in Java.**

- Next is the main method.
- **Every application in Java must contain the main method.**
- The Java compiler starts executing the code from the main method, and **it's mandatory in each of our executable Java programs.**

- The signature of the main method in Java is:

    - **public static void main(String [] args) { ... }**

How Does the "Hello, World!" Program Work?

- After that, we have a comment.
- In Java, **any line starting with // is a single-line comment.**
- Comments are intended for users reading the code to understand the intent and functionality of the program.
- **It is completely ignored by the Java compiler (an application that translates Java program to Java bytecode that computer can execute).**

- Lastly, we have **a print statement**.
- It **prints the text “Hello, World!” to standard output (your screen)**.
- The text inside the quotation marks is called a String literal in Java.
- Notice the print statement is inside the main function, which is inside the class definition.

Creating Your Own Java Program

- For the time being, all of our programs will be just like HelloWorld.java, except with a different sequence of statements in main().

- The easiest way to write such a program is to:
  - **Copy HelloWorld.java into a new file whose name is the program name followed by .java**
  - **Replace HelloWorld with the program name everywhere**
  - **Replace the print statement by a sequence of statements**

- Let's make a new program called **Greeting.java**, that prints two lines:
  - First: **Hello, my name is [YOUR NAME HERE].**
  - Second: **How are you?**

Types of Errors

- There are three types of errors can occur in a program:
  - **Compile-time errors**. These **errors are caught by the system when we compile the program, because they prevent the compiler from doing the translation** (so it issues an error message that tries to explain why).
  - **Run-time errors**. These errors are **caught by the system when we execute the program, because the program tries to perform an invalid operation** (e.g., division by zero).
  - **Logical errors**. These **errors are (hopefully) caught by the programmer when we execute the program and it produces the wrong answer**. Bugs are the bane of a programmer's existence. They can be subtle and very hard to find.
- One of the very first skills that you will learn is to identify errors; one of the next will be to be sufficiently careful when coding to avoid many of them.

Types of Errors

- Example: Let's say I typed in the following program:

```
public class Hello {
    public static void main() {
        System.out.println("Doesn't execute");
    }
}
```

- It compiles fine, but **when I execute it, I get the error java.lang.NoSuchMethodError: main**. What am I doing wrong? What type of error is this?

String [] args and User Interactivity

- Typically, we want to provide input to our programs: data that they can process to produce a result.
- The simplest way to provide input data is by using the “String [] args” parameter as illustrated in UseArgument.java below:

```
public class UseArgument {

    public static void main(String[] args) {
        System.out.print("Hi, ");
        System.out.print(args[0]);
        System.out.println(". How are you?");
    }

}
```

String [] args and User Interactivity

- **The “String[] args” parameter allows command-line arguments to be passed to the program.** (Through the use of an array which you will learn more about late)

- These arguments can be accessed within the main method using the “args” parameter.
- args[0] is the first (and in our case only) command-line argument that we will pass to the program

- Whenever **this program is executed**, it **reads the command-line argument that you type after the program name and prints it back out to the terminal as part of the message.**
  - Compilation: **javac UseArgument.java**
  - Execution: **java UseArgument [your name]**

New Java Programming (As of JDK 21 and onward)

- For the last thirty years, ever since Java 1.0, the “Hello, World” program in Java looked like our first example.
- However, in modern Java (starting with Java 21 in preview mode, and finalized in Java 25), **the main method no longer needs to be public static, and the String[] argument is optional**
- This means that we can actually rewrite our HelloWorld.java program to look like this now:

```
void main() {
    IO.println("Hello, World!");
}
```

- **You don't even need a class.** Just put void main() { ... } in a Java file.

New Java Programming (As of JDK 21 and onward)

- To use the command line, **invoking a Java program has become simpler. There is no need to compile the program.** Simply call:

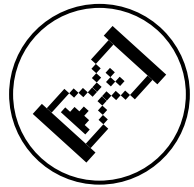
```
java MyProgram.java
```

- In our case, we'll say:
  - **java --enable-preview HelloWorld.java**
- As of now, we will have to enable preview mode until Java 25 releases in September.
- The program automatically compiles and runs, just like with Python.
- This also works for programs that consist of multiple source files. Just put them in the same folder and launch the file with the main method.

04

Introduction to IDEs

(Integrated Development Environments)



What is an IDE?

- Although as we just saw, we can write, compile, and run our programs with a text editor and command-line interface, it may be simpler to use an IDE.
- **A Java IDE (integrated development environment) is a software program that developers use to write and debug code more easily.**
- You are encouraged to develop programs with an IDE (Eclipse, Netbeans, Visual Studio, etc.) because **IDEs have many built-in features that make it easier to work with large programs!**
- Most Java IDEs have a:
  - **code editor,**
  - **a set of tools for automating the building process,**
  - **and a debugger**
- Java IDEs can **increase productivity by combining capabilities such as editing, building and testing within a single application.**

What is an IDE?

- You are free to choose one (or more!) IDE(s) that you like best but as a suggestion some of the most frequently used IDEs include:
  - Eclipse
  - NetBeans
  - IntelliJ IDEA Community Edition
  - DrJava
  - Visual Studio Code (with extensions for Java!)

- Since **NetBeans is slightly easier to use than Eclipse, it's also a good choice for beginner developers.**
- One of the advantages of NetBeans is that it's part of the Apache ecosystem, meaning built in Apache Maven functionality for users.

Installing Platform-Independent NetBeans on Windows/macOS

- Since we have already downloaded the JDK, we can skip to **downloading the application from https://netbeans.apache.org/front/main/index.html**
- There are many "bundles" available. **I suggest you choose the platform independent ZIP version (e.g.,"netbeans-26-bin.zip").**
- NetBeans is written in Java, hence, it is platform independent.
- Once downloaded, unzip the downloaded file into a directory of your choice.

05

Writing Your First Java Program
(Part 2 - Using the IDE)

Writing Java Programs in NetBeans

- **Launch NetBeans by running netbeans.exe or netbeans64.exe under "bin".** If the "Start Page" appears, close it by clicking the "cross" button next to the "Start Page" title.

- **For each Java application, you need to create a "project" to keep all the source files, classes and relevant resources:**
  - From **"File"** menu ⇒ Choose **"New Project ..."**.
  - The **"Choose Project"** dialog pops up ⇒ Under **"Categories"**, choose **"Java with Maven"** ⇒ Under **"Projects"**, choose **"Java Application"** ⇒ **"Next"**.
  - The **"Name and Location"** dialog pops up ⇒ Under **"Project Name"**, enter **"FirstJavaProject"** ⇒ In **"Project Location"**, select a suitable directory to save your works ⇒ In Group Id: enter **"com.nowhere"** ⇒ **Finish**.
  - **A Hello-world program FirstJavaProject.java is automatically created.** You can right-click on the source file ⇒ **Run File**.

Writing Java Programs in NetBeans

- Now we can try rewriting the programs that we wrote using the text editor and compile and run them through our IDE.
- **Right-click on package "com.nowhere.firstjavaproject" ⇒ New ⇒ Java Class** ⇒ In **"Class Name"**, enter **"HelloWorld"** ⇒ **"Finish"**.
- The source file **"HelloWorld.java"** appears in the editor panel.
- You can now rewrite the HelloWorld.java file that we previously wrote.
- Once you have done that, there is no need to "compile" the source code in NetBeans explicitly, as NetBeans performs the so-called incremental compilation (i.e., the source statement is compiled as and when it is entered).
- **To run the program, right-click anywhere in the source (or from the "Run" menu) ⇒ Run File.**
- Observe the output on the output console.

Writing Java Programs in NetBeans

- Note: **You should create a NEW Java project for EACH of your Java applications.**

- Nonetheless, **NetBeans allows you to keep more than one programs in a project**, which is handy for writing toy programs (such as your tutorial exercises).

- To run a particular program, open and **right-click on the source file ⇒ Run File**.

- Let's try this by creating a new Java class for **Greeting.java**, rewriting our original Greeting.java code and running it in our IDE.

Writing Java Programs in NetBeans

- Let's create a new Project for UseArgument
- From **"File"** menu ⇒ Choose **"New Project ... "**.
- The **"Choose Project"** dialog pops up ⇒ Under **"Categories"**, choose **"Java with Maven"** ⇒ Under **"Projects"**, choose **"Java Application"** ⇒ **"Next"**.
- The **"Name and Location"** dialog pops up ⇒ Under **"Project Name"**, enter **"UseArgument"** ⇒ In "**Project Location"**, select a suitable directory to save your works ⇒ In Group Id: enter **"com.nowhere"** ⇒ **Finish**.
- Just like before, A Hello-world program UseArgument.java is automatically created.
- **Delete the code and rewrite our previous UseArgument.java code in its place**

Writing Java Programs in NetBeans

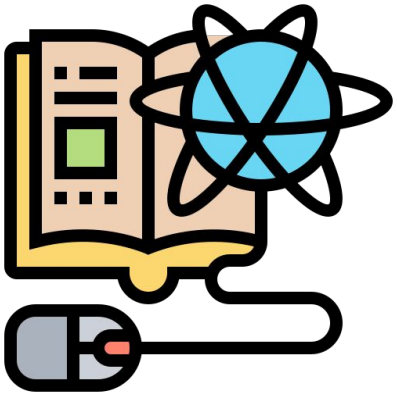
- If you try running the project by pressing the green triangle in the toolbar (which we can since we only have one file in this particular project), you should encounter the following error:

```
Hi, Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds for length 0
```

- This is because command-line arguments are normally used in the command-line interface (hence the name)
- However, we can adjust for this in our IDE as well
- **Right-click (on Windows) or Control-Click (on a Mac) the project branch in the NetBeans Projects pane.**
- In the resulting context menu, **select Properties.**

Writing Java Programs in NetBeans

- As a result, the Project Properties dialog box opens.
- On the left side of the Project Properties dialog box, **select Run.**
- In the main body of the Project Properties dialog box, **make sure that the Main Class field contains the**
- **name of the class containing the main method.**
- Then **type the command-line arguments in the Arguments field** (your name).
- **Click OK** to dismiss the Project Properties dialog box.
- **Run the program** in the usual way (by clicking the green arrow Run Project icon in the NetBeans toolbar) and **you should see the same outcome as we did in the CLI.**



06

Overview and Next Steps

Overview

- As of now, you should have/be:
  - Downloaded the Java Development Kit (JDK)
  - Briefly introduced to using the command-line Interface (CLI) to run programs by:
    - Writing simple Java programs in a text editor of your choice
    - Compiling the program using javac
    - Executing the program using java
  - Downloaded an Integrated Development Environment (IDE) – (we used NetBeans in this workshop but you are free to use whichever you would like)
  - Rewrote and ran our programs using the IDE for a brief introduction to the application

Next Steps

- This all just scratches the surface of all that you will learning in your upcoming CISC 1115 course, but it is definitely a good start!
- You are highly encouraged to read the following text (one of the best intro books for Java AND available as a free PDF online):

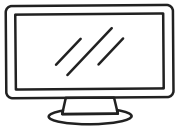
Allen Downey and Chris Mayfield, *Think Java: How to Think Like a Computer Scientist*, 2nd Edition, Version 7.1.0, Green Tea Press, 2020, Creative Commons License.

- In addition to whatever resources that will be required from your respective instructors

Next Steps

- I would encourage you to get a feel of **writing programs by hand (<u>your exams and final will all be handwritten</u>)** before running your programs on your own computers
- This, along with **tracing already written programs to determine the output**, will help you better retain information and an understanding of how exactly your programs work
- Furthermore, **tutors are available in the Learning Center – 1300 Boylan Hall** if you ever need any study assistance throughout the course

- Congratulations and best of luck on your CS journey! 🎉

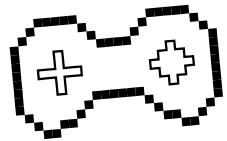


Thank You!

Presented By:

Amara Auguste

CS Tutor, Graduate Student,
and Adjunct Lecturer



Do you have any further questions?

auguste@sci.brooklyn.cuny.edu

amaraauguste.github.io