C212 Lab 1

Objectives:

- Identify how to setup a development environment of your choice. (**IntelliJ IDEA**, Eclipse, Netbeans, VS Code, etc/)
- Identify the purpose and utility of text editors and IDEs in Java development (vs scripting or dynamic programming languages)
- Apply what is listed in the lab to create and compile a basic java program.
- Understand where to find (and how to use) Oracle's Java API documentation and Oracle Java documentation tutorials
- Start getting a general understanding of Object-Oriented Concepts from reading – this will immensely help when beginning to learn the Java programming language

Development Environment:

Your Java development environment will consist of two parts: your installed **JDK** (Java Development Kit), and your **code editor**. You may already have installed Java as a consumer (or have it pre-installed on your computer), but there is a difference between the **JRE** (Java Runtime Environment) and JDK. The JDK provides access to the **javac** program, which lets you input .java source code and output .class bytecode, which is runnable using the JRE's **java** program.

- You must install a version of the JDK (at least JDK 8) on your computer. We recommend using Oracle's JDK 16 (oracle.com/java/technologies/javase-jdk16-downloads.html), but OpenJDK or any other distribution is alright as well. All JDK implementations implement the same Java language specification*, which is both why all major distributions will work for this class and why the Oracle Java **javadocs** are a good goto reference whenever you have a question about something in the **Java standard library**

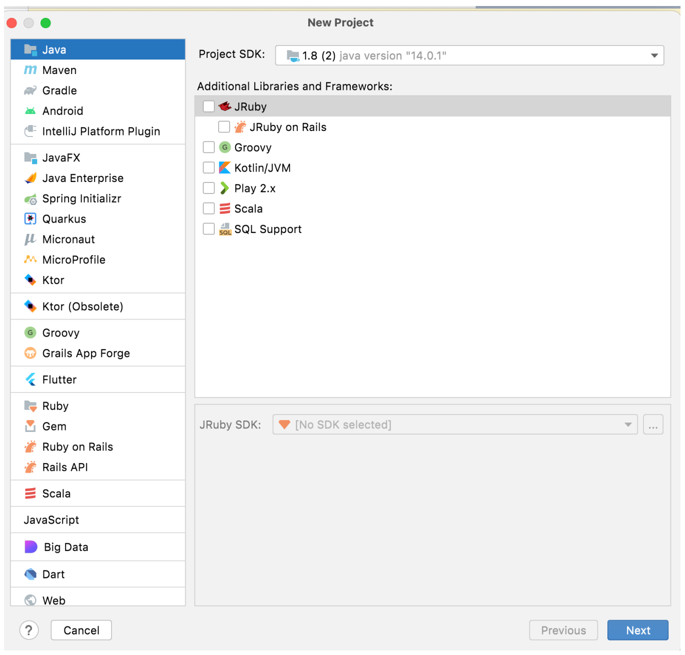
*except for a few quirks

- After this, download an IDE or text editor of your choice.
- We recommend IntelliJ IDEA, as its code completion, static analysis, compilation, and syntax highlighting and shortcuts are user-friendly and helpful, especially for new Java developers. However, Eclipse, Netbeans, Atom, VS Code, Dr. Java, or even textedit can all be used.
- Download links:
    - IntelliJ IDEA: https://www.jetbrains.com/idea/download/ (Community Edition)
    - Netbeans: https://netbeans.apache.org/download/index.html
    - Eclipse: https://www.eclipse.org/downloads/

Task 1: Creating a Project.

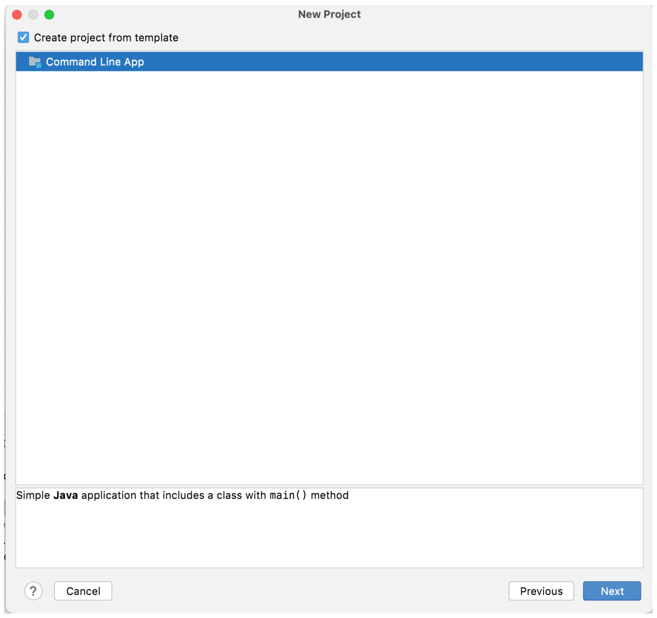
First, create a new Java project if you're working in an IDE (IntelliJ IDEA, Eclipse, etc..). For those of you who prefer a text editor, just create a new file.

New project creation steps for IntelliJ IDEA are shown below: For now you will not need to select any additional libraries or frameworks.



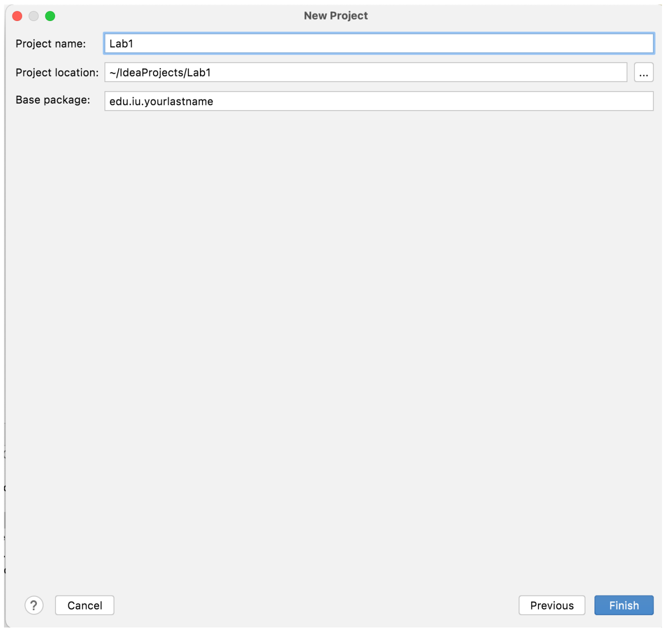
Project SDK: 1.8 (2) java version "14.0.1"

JRuby SDK: [No SDK selected]



☑ Create project from template

Simple **Java** application that includes a class with `main()` method

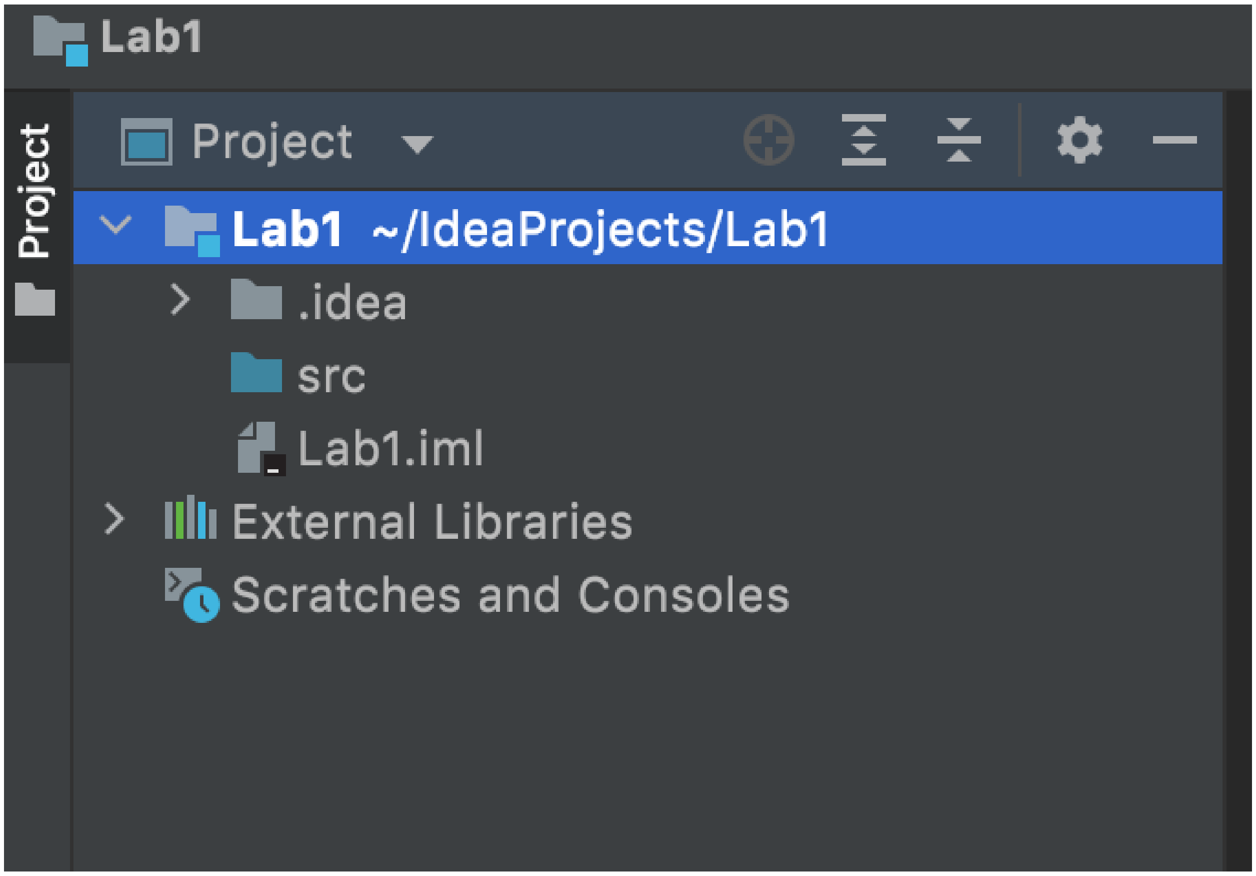


Project name: Lab1

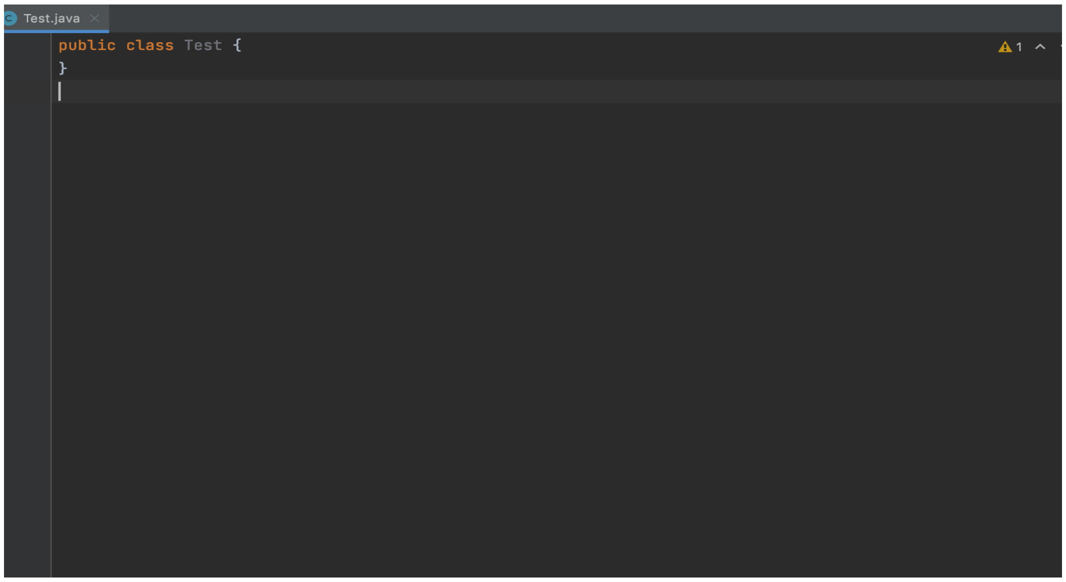
Project location: ~/IdeaProjects/Lab1

Base package: edu.iu.yourlastname

After creating the project your file directory should look something like this:



Right click the src folder and select New > Java Class.
You can then name your class and click next and it will create your file, it should look like this:



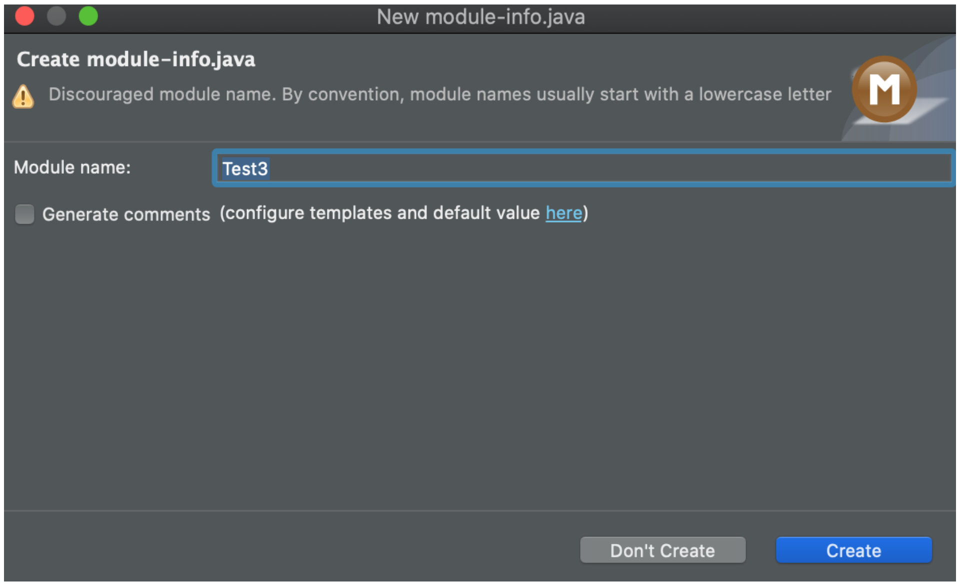
```
public class Test {
}
```

Now you are ready to start writing your java program. Don't forget to create your main method.

A main method in java starts the execution of the code, it is written as
public static void main(String[] argos){
}

Eclipse:

Press File > New > New Java project.
Name your project Lab1.
After pressing next you will see the following screen:


New module-info.java

Create module-info.java

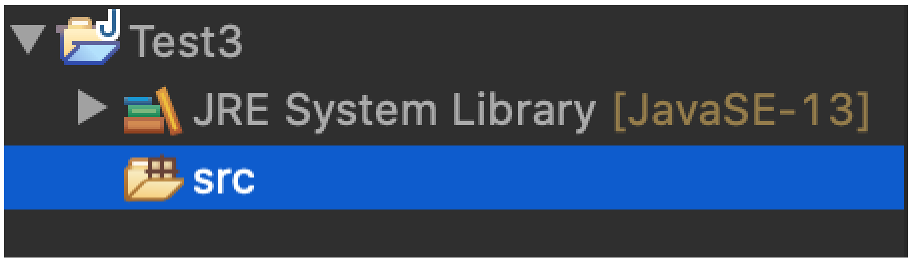
Discouraged module name. By convention, module names usually start with a lowercase letter

Module name: Test3

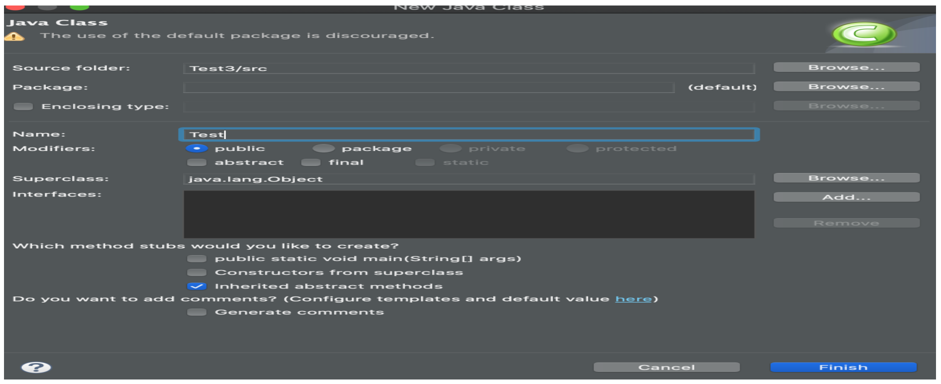
Generate comments (configure templates and default value here)

You do not need to worry about creating a module. It can be helpful later in the semester for larger projects, but at the time being it is not necessary to use.

After creating a project you should have a file directory that looks like this:


Test3
JRE System Library [JavaSE-13]
src

Right click on the src folder and click New > Class.
Doing so will bring you to a screen showing this:



New Java Class

Java Class

The use of the default package is discouraged.

Source folder: Test3/src

Package: (default)

Enclosing type:

Name: Test

Modifiers: [✓ public] package private protected

abstract final static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

- [ ] public static void main(String[] args)
- [ ] Constructors from superclass
- [✓] Inherited abstract methods

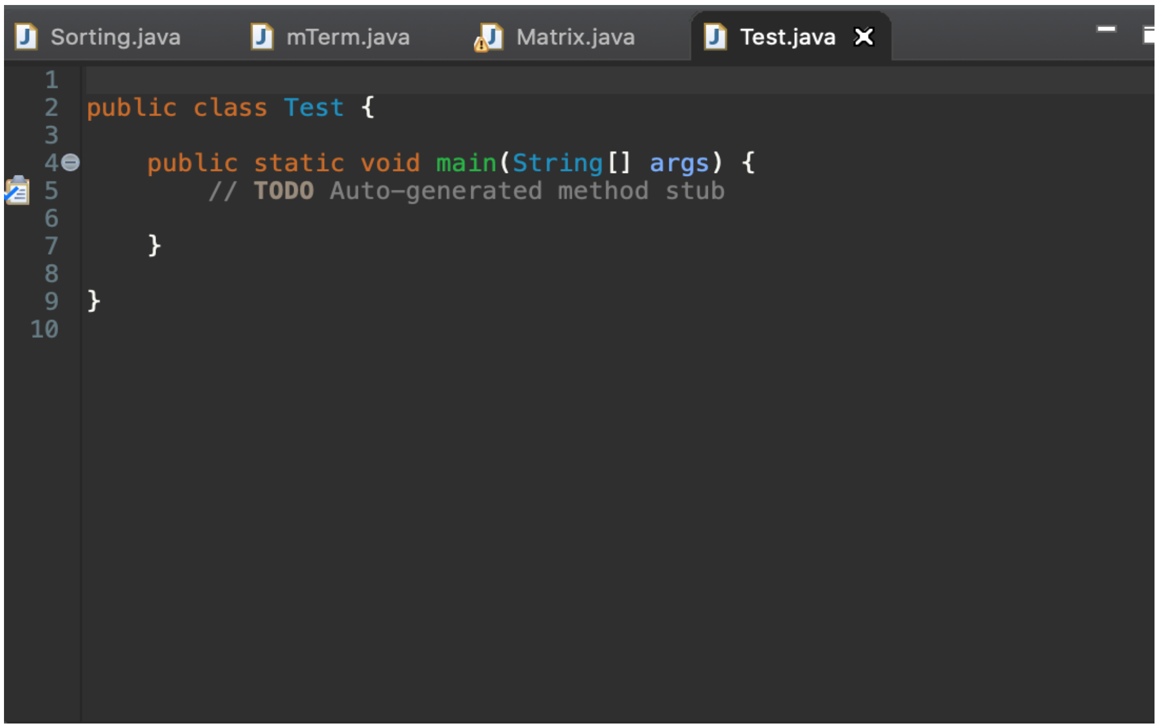
Do you want to add comments? (Configure templates and default value here)

- [ ] Generate comments

Give your class a name, if you would like you can check the box that says
Public static void main(String[] args)

Checking this box will create a main function for you, although you may want to practice writing them yourselves.

When done click create and your new java file in eclipse should look like this:



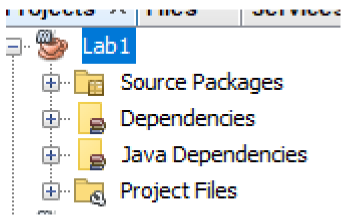
```

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}

```

Now you are ready to start writing your java program.

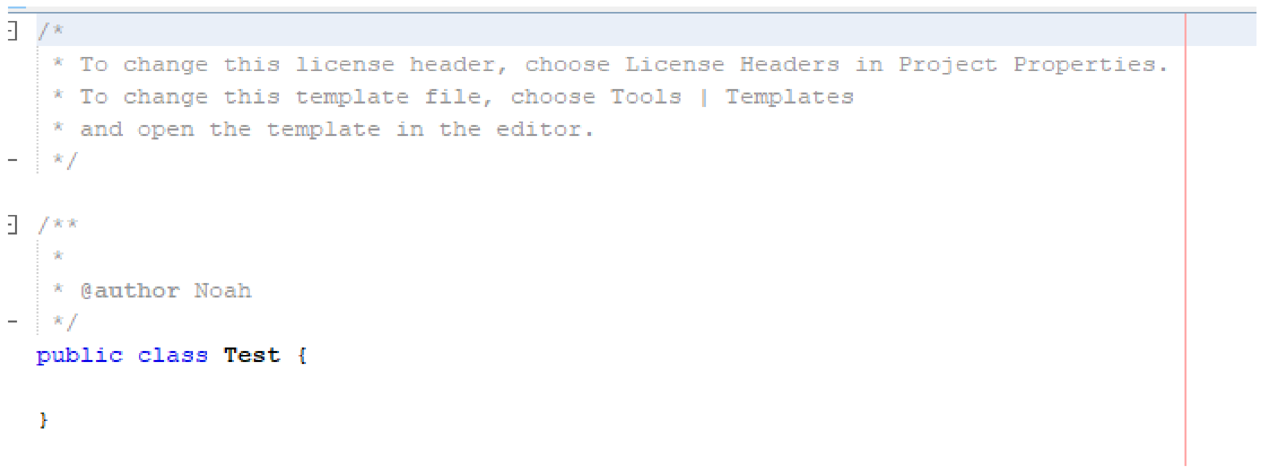
NetBeans IDE:

This IDE is similar to the other two.

- Start by opening the IDE and clicking File > New Project. There are a few options, but for now just select Java with Maven. It may ask you to activate some other modules, go ahead and do so.
- Create a project name, something like Lab1 should be fine.



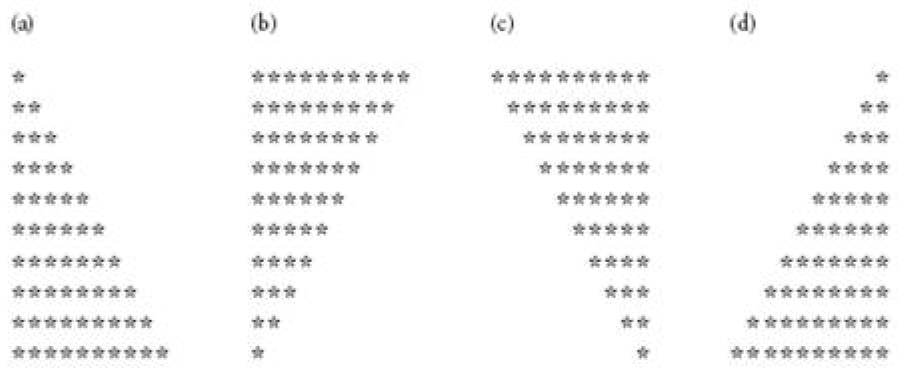
After those steps your file directory should look like this.
You can then right click on the Source Packages folder and create a new Java Class. Give your class a name and press enter, after that your file should open and look like this:



```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Noah
 */
public class Test {

}
```

Now your file is created and you are ready to start writing your main method and the remainder of your java program.

Task 2: Creating patterns through print statements



```
(a)          (b)          (c)          (d)

*            **********   **********            *
**           *********     *********           **
***          ********       ********          ***
****         *******         *******         ****
*****        ******           ******        *****
******       *****             *****       ******
*******      ****               ****      *******
********     ***                 ***     ********
*********    **                   **    *********
**********   *                     *   **********
```

TODO:

- Print the following 4 asterisks patterns in your project you have just created. Use java's System.out.print or System.out.println methods. All patterns can be made by just using print statements, and can all be done inside of your main function.
- Properly comment your code, example // Printing pattern (a):

(public static void main(String[] args))

- After you have written the respective print statements, compile and run your project, and make sure the patterns match the image shown above.

Useful Links:

- Getting started: The Java Technology Phenomenon and the A Closer Look at “Hello World” Application - https://docs.oracle.com/javase/tutorial/getStarted/index.html
- OOP Concepts: https://docs.oracle.com/javase/tutorial/java/concepts/index.html
- Java 8 API: https://docs.oracle.com/javase/8/docs/api/index.htm

Glossary:

JDK: Java Development Kit
JRE: Java Runtime Environment
Code Editor: Either a text editor or an IDE that provides smart tools to developers such as type hints and static analysis
javac: The Java language compiler program, compiling Java source files (.java) to Java bytecode (.class)
java: The Java language runtime program, allowing you to run compiled Java bytecode
javadoc: The method, class, and package documentation generated from a Java program
Java standard library (stdlib): The set of classes that come implemented inside the JDK. Other libraries can be used, but no additional work besides importing is required to use any class within the standard library.
Scanner: A class that can read values from an input string, file, or the console. It is used to get dynamic input.

Submission:

- Please turn in your java file with your created patterns into the appropriate location on canvas.