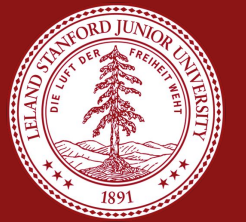
How to Start Your Own Java Programs

Lecture 25

CS106A, Summer 2019
Sarai Gould && Laura Cruz-Albrecht

With inspiration from slides created by Keith Schwarz, Mehran Sahami, Eric Roberts, Stuart Reges, Chris Piech, Brahm Capoor, & others.



Announcements

- Download blank starter code from the website!

Learning Goals for Today

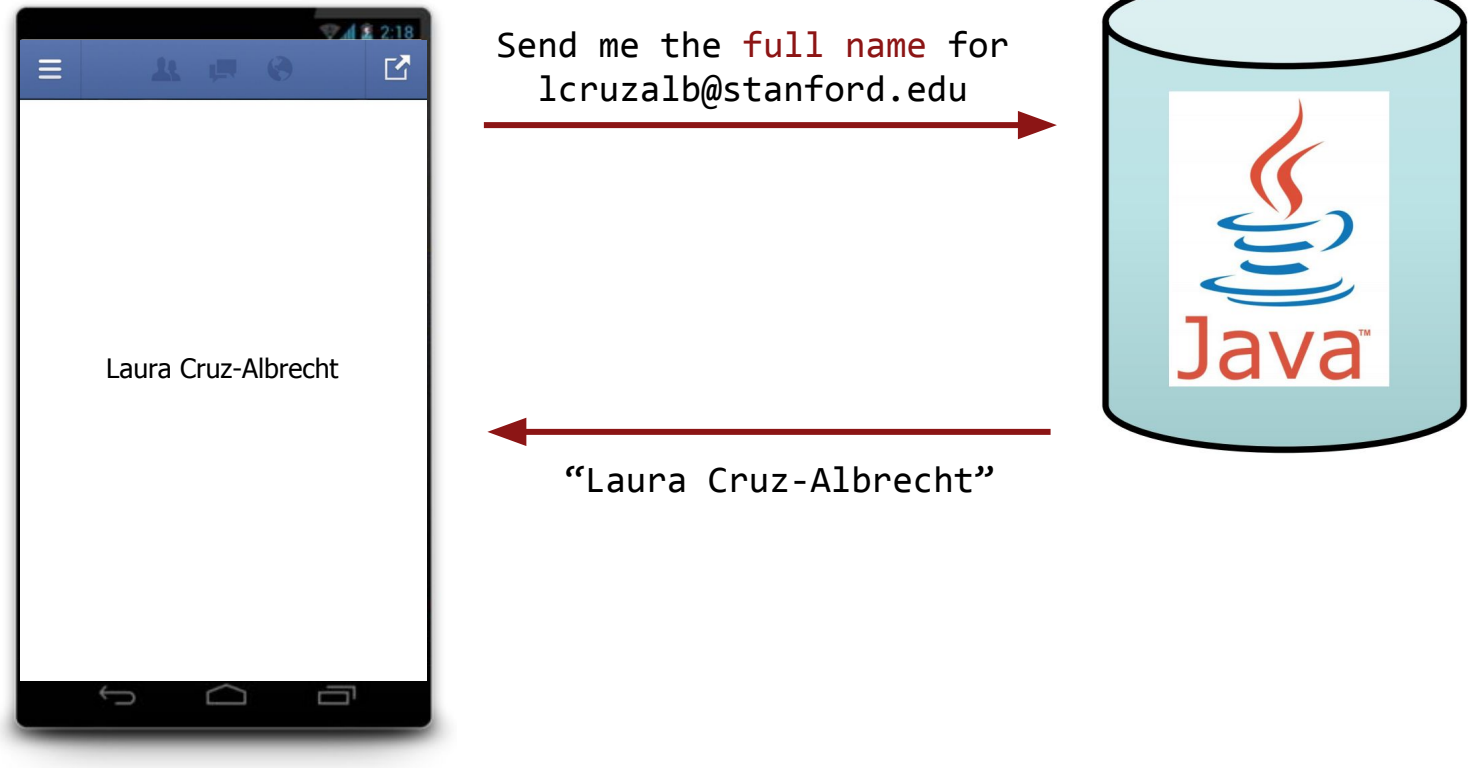
Learn About Writing Java code in the Real World!

Plan for Today

- Review: Server & Client
- “Real” Java
- Console Program
- Basic Gui
- Creating Our Own Project!
- Example: Reading HTML with JSoup

There are two types of internet programs: **servers** and **clients**.

Clients *send* ***requests*** to servers. ***Servers respond*** to those requests.


Facebook Server
Laura Cruz-Albrecht
Send me the full name for
lcruzalb@stanford.edu
Java
“Laura Cruz-Albrecht”

URL

In PollClient, change the *HOST* constant to:

http://cbc0026a.ngrok.io

Real Java?

Does this mean we didn't learn REAL Java?

Of Course We Did!

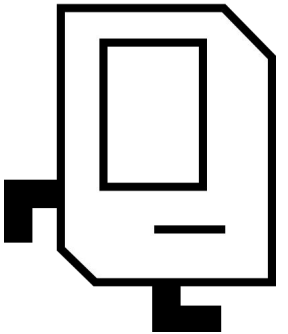
While we did use some Stanford specific Java libraries at times, you did learn REAL Java!

All of the data structures, primitives, classes, loops, etc are all part of real programming and real Java!

ACM Libraries

All quarter, we have used the Java **ACM Libraries**.

- Karel, ConsoleProgram, RandomGenerator…
- GraphicsProgram, GOval, GRect, GImage…

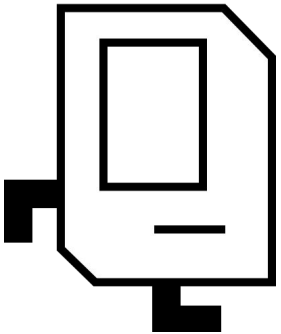


Can you imagine Java without me?!

I can't...

ACM Libraries

Today, we'll look at how **standard Java** programs are written!



Sounds fun!

Can't wait! 😄

Hello World with ACM

This was the first **ConsoleProgram** we ever wrote! Notice:

- it **import**s acm.program.*
- it **extends** a ConsoleProgram
- It uses **public void** run()
- It uses println()

```
import acm.program.*;

public class HelloWorld extends ConsoleProgram {
    public void run() {
        println("Hello, world!");
    }
}
```

Standard Hello World

This is Hello World in Standard Java! Notice:

- no **imports** or **extends** needed for this program!
- It uses **public static void** main(String[] args)
- It uses System.*out*.println()

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

main

Why `public static void main(String[] args)`?

`main` is the true entry point for a program in Java.

- It must be written exactly as seen above!
- `String[] args` is an array of String arguments given to the program.

main

Why `public static void main(String[] args)`?

`main` is the true entry point for a program in Java.

- It must be written exactly as seen above!
- `String[] args` is an array of String arguments given to the program.

`System.out.println()` is the true `println()`

Standard Java methods are `static` unless part of a class of objects.

Why did we use `ConsoleProgram`?

Creates a new window.

Puts a new scrollable area in the window.

Provides **`print`** and **`println`** commands to send text output to the window.

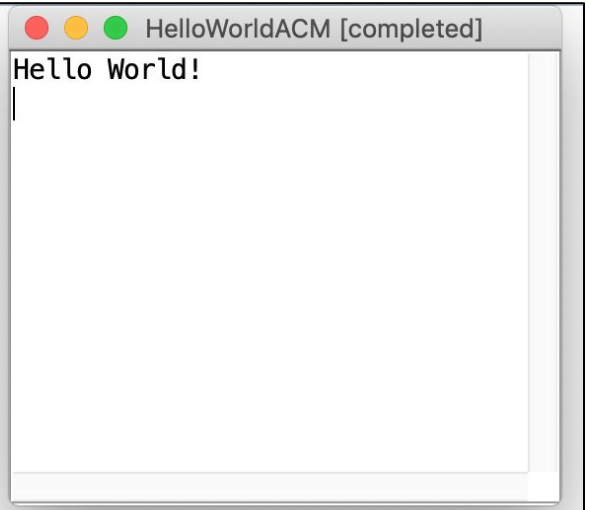
Contains a **`main`** method that calls your **`run`** method.

Why did we use ConsoleProgram?

```
/*
 * File: HelloWordACM.java
 * -----------------------
 * This implements Hello World using the ACM Libraries!
 */

import acm.program.*;

public class HelloWorldACM extends ConsoleProgram{
    public void run() {
        println("Hello World!");
    }
}
```



Standard Java Version

HelloWorldStandard.java | HelloWorldACM.java

```
1  /*
2   * File: HelloWordStandard.java
3   * ----------------------
4   * This implements Hello World in standard Java!
5   */
6
7
8  public class HelloWorldStandard{
9      public static void main(String[] args) {
10         System.out.println("Hello World!");
11     }
12 }
13
```

Console | Debug

<terminated> HelloWorldStan

```
Hello World!
```

Reading Input with ACM

This is a short program that read in input! Notice:

- It uses `readLine` and `readInt` to read input

```
import acm.program.*;

public class SayingHello extends ConsoleProgram {
    public void run() {
        String name = readLine("What's your name?");
        int age = readInt("How old are you?");

        println("Hello" + name + "!");
        println(name + " is " + age + " years old.");
    }
}
```

Reading Input with Standard Java

This is another short program that read in input!

```
import java.util.Scanner;

public class SayingHello{
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.println("What's your name?");
        String name = console.nextLine();
        System.out.println("What's your name?");
        int age = console.nextInt();

        System.out.println("Hello" + name + "!");
        System.out.println(name + " is " + age + " years old.");
    }
}
```

Reading Input with Standard Java

This is another short program that read in input!

```
import java.util.Scanner;

public class SayingHello{
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.println("What's your name?");
        String name = console.nextLine();
        System.out.println("What's your name?");
        int age = console.nextInt();

        System.out.println("Hello" + name + "!");
        System.out.println(name + " is " + age + " years old.");
    }
}
```

We have to create a Scanner to read input!

Reading Input with Standard Java

This is another short program that read in input!

```
import java.util.Scanner;

public class SayingHello{
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.println("What's your name?");
        String name = console.nextLine();
        System.out.println("What's your name?");
        int age = console.nextInt();

        System.out.println("Hello" + name + "!");
        System.out.println(name + " is " + age + " years old.");
    }
}
```

`System.in` is another name for the console here.

Reading Input with Standard Java

This is another short program that read in input!

```
import java.util.Scanner;

public class SayingHello{
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.println("What's your name?");
        String name = console.nextLine();
        System.out.println("What's your name?");
        int age = console.nextInt();

        System.out.println("Hello" + name + "!");
        System.out.println(name + " is " + age + " years old.");
    }
}
```

We have to print out the question using: `System.out.println`

Reading Input with Standard Java

This is another short program that read in input!

```
import java.util.Scanner;

public class SayingHello{
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.println("What's your name?");
        String name = console.nextLine();
        System.out.println("What's your name?");
        int age = console.nextInt();

        System.out.println("Hello" + name + "!");
        System.out.println(name + " is " + age + " years old.");
    }
}
```

We read in the input using:
`console.readLine()`

Reading Input with Standard Java

This is another short program that read in input!

```
import java.util.Scanner;

public class SayingHello{
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.println("What's your name?");
        String name = console.nextLine();
        System.out.println("What's your name?");
        int age = console.nextInt();

        System.out.println("Hello" + name + "!");
        System.out.println(name + " is " + age + " years old.");
    }
}
```

GraphicsPrograms?

Creates a new window.

Creates a drawing canvas in the center of the window.

Provides convenient methods to detect Mouse Events.

Contains a `main` method that calls your `run` method.

Basic GUI with ACM

```
public class ColorFun extends Program {
    public void init() {
        JButton redButton = new JButton("Red!");
        JButton blueButton = new JButton("Blue!");
        add(redButton, SOUTH);
        add(blueButton, SOUTH);
        addActionListeners();
    }

    public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equals("Blue!"){
            setBackground(Color.BLUE);
        } else {
            setBackground(Color.RED);
        }
    }
}
```

Basic GUI with Standard Java

```
public class ColorFun implements ActionListener {
      public static void main(String[] args) {
            new ColorFun().init();
      }
      private JFrame frame;
      public void init() {
         frame = new JFrame("ColorFun");
         frame.setSize(500, 300);
         JButton redButton = new JButton("Red!");
         JButton blueButton = new JButton("Blue!");
         redButton.addActionListener(this);
         blueButton.addActionListener(this);
         frame.add(redButton, "South");
         frame.add(blueButton, "North");
         frame.setVisible(true);
      }
```

```
   public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equals("Blue!"){
            frame.setBackground(Color.BLUE);
        } else {
            frame.setBackground(Color.RED);
        }
    }
}
```

Basic GUI with Standard Java

```
public class ColorFun implements ActionListener {
      public static void main(String[] args) {
            new ColorFun().init();
      }
      private JFrame frame;
      public void init() {
         frame = new JFrame("ColorFun");
         frame.setSize(500, 300);
         JButton redButton = new JButton("Red!");
         JButton blueButton = new JButton("Blue!");
         redButton.addActionListener(this);
         blueButton.addActionListener(this);
         frame.add(redButton, "South");
         frame.add(blueButton, "North");
         frame.setVisible(true);
      }
```

```
   public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equals("Blue!"){
            frame.setBackground(Color.BLUE);
        } else {
            frame.setBackground(Color.RED);
        }
    }
}
```

Basic GUI with Standard Java

```
public class ColorFun implements ActionListener {
    public static void main(String[] args) {
        new ColorFun().init();
    }
    private JFrame frame;
    public void init() {
        frame = new JFrame("ColorFun");
        frame.setSize(500, 300);
        JButton redButton = new JButton("Red!");
        JButton blueButton = new JButton("Blue!");
        redButton.addActionListener(this);
        blueButton.addActionListener(this);
        frame.add(redButton, "South");
        frame.add(blueButton, "North");
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equals("Blue!")){
            frame.setBackground(Color.BLUE);
        } else {
            frame.setBackground(Color.RED);
        }
    }
}
```

Basic GUI with Standard Java

```
public class ColorFun implements ActionListener {
      public static void main(String[] args) {
            new ColorFun().init();
      }
      private JFrame frame;
      public void init() {
         frame = new JFrame("ColorFun");
         frame.setSize(500, 300);
         JButton redButton = new JButton("Red!");
         JButton blueButton = new JButton("Blue!");
         redButton.addActionListener(this);
         blueButton.addActionListener(this);
         frame.add(redButton, "South");
         frame.add(blueButton, "North");
         frame.setVisible(true);
      }
```

```
      public void actionPerformed(ActionEvent e) {
            if(e.getActionCommand().equals("Blue!")){
                  frame.setBackground(Color.BLUE);
            } else {
                  frame.setBackground(Color.RED);
            }
      }
}
```

Basic GUI with Standard Java

```
public class ColorFun implements ActionListener {
    public static void main(String[] args) {
        new ColorFun().init();
    }
    private JFrame frame;
    public void init() {
        frame = new JFrame("ColorFun");
        frame.setSize(500, 300);
        JButton redButton = new JButton("Red!");
        JButton blueButton = new JButton("Blue!");
        redButton.addActionListener(this);
        blueButton.addActionListener(this);
        frame.add(redButton, "South");
        frame.add(blueButton, "North");
        frame.setVisible(true);
    }
```

```
    public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().equals("Blue!")){
            frame.setBackground(Color.BLUE);
        } else {
            frame.setBackground(Color.RED);
        }
    }
}
```

Basic GUI with Standard Java

We had to create a frame.

Adding actionListeners was a lot more involved!

Why Not Libraries?

Why don't we just use ACM all the time then?

Basic GUI with ACM

- Benefits of libraries:
  - simplify syntax/rough edges of language/API
  - avoid re-writing the same code over and over
  - possible to make advanced programs quickly
  - leverage work of others

Basic GUI with ACM

- Benefits of libraries:
  - simplify syntax/rough edges of language/API
  - avoid re-writing the same code over and over
  - possible to make advanced programs quickly
  - leverage work of others

- Drawbacks of libraries:
  - limitations on usage; e.g. **ACM library cannot be distributed for commercial purposes.**

Creating Our Own Java Project

What other sorts of cool Java libraries are out there?

How do we create our own Java projects using them?

Reading from the Web

We're going to write a program that will grab HTML from a webpage from the internet and print out important information from that page!

Before we pick the webpage, let's set up our new Java Project!

Let's Create an Empty Project First!

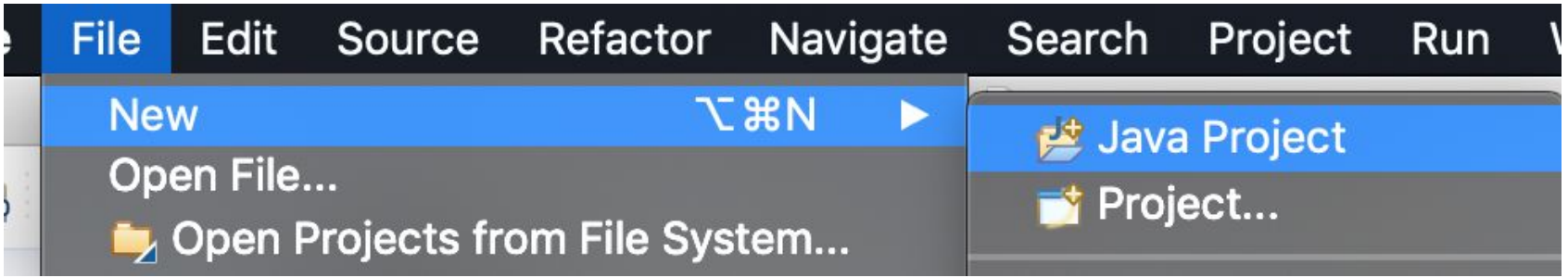
To create an empty project on your own, you don't need the starter code!

You will need the starter code for some of the cool code we'll use to read the HTML from our web page.

Let's Create an Empty Project First!

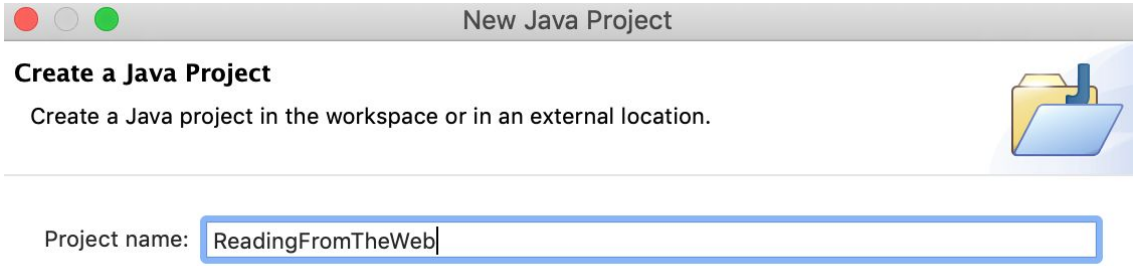
In your Eclipse menu, click on:

File -> **New** -> **Java Project**

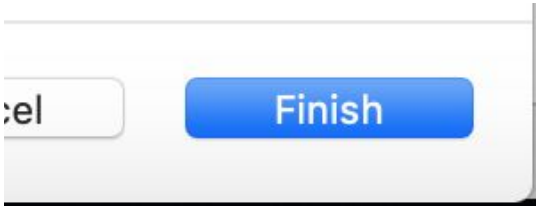


Let's Create an Empty Project First!

Give your project a name:



Click **Finish**:



Let's Create an Empty Project First!

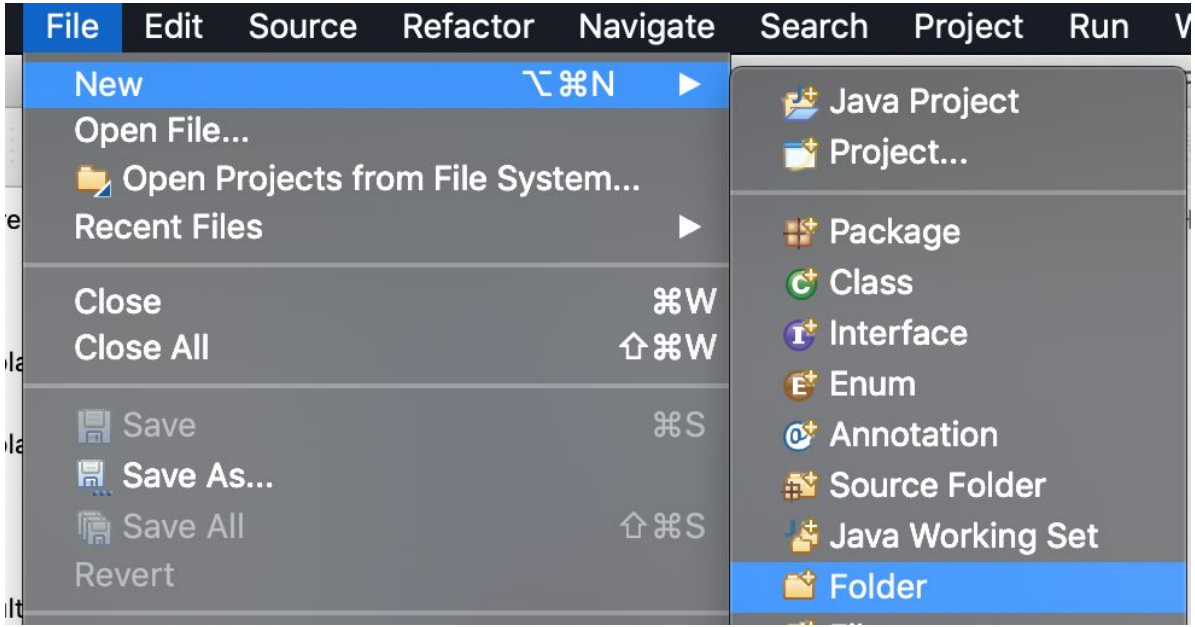
Eclipse will ask if you want to create a module.

Click **Don't Create**:

Let's Create a lib Folder

In your Eclipse menu, click on:

File -> **New** -> **Folder**

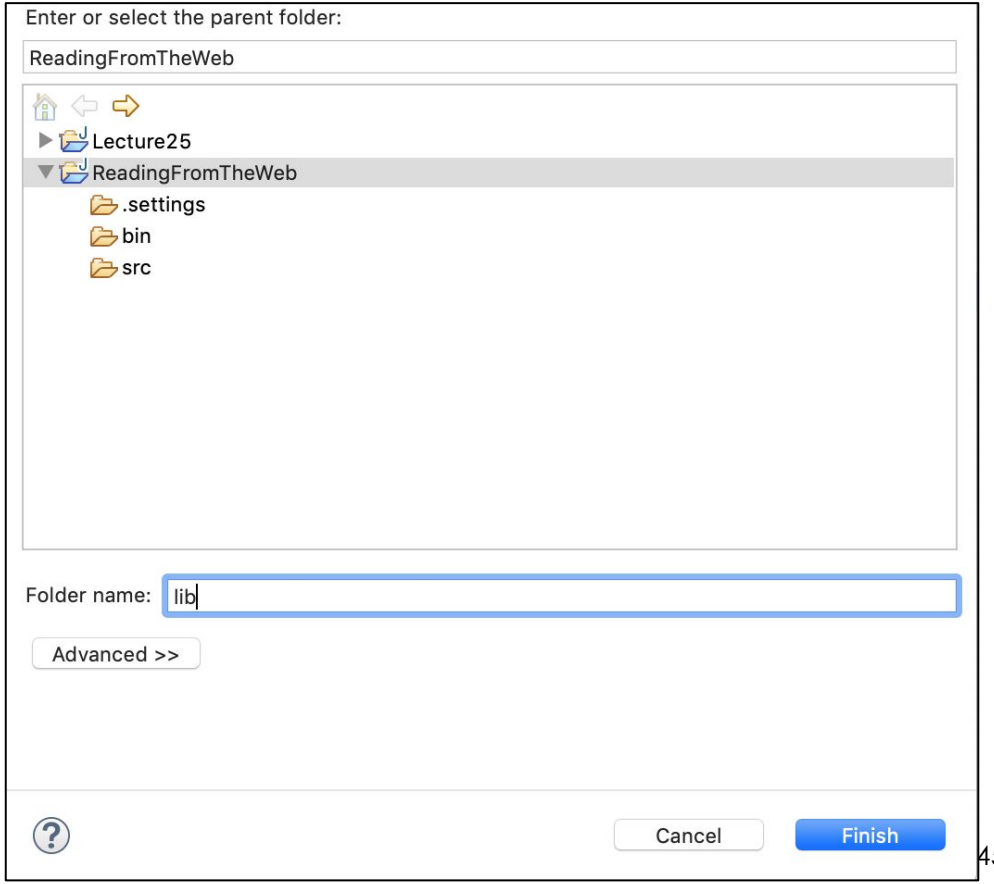


Let's Create a lib Folder

Select your ReadingFromTheWeb project folder.

Create a **lib** folder to hold any .JAR files we may want!

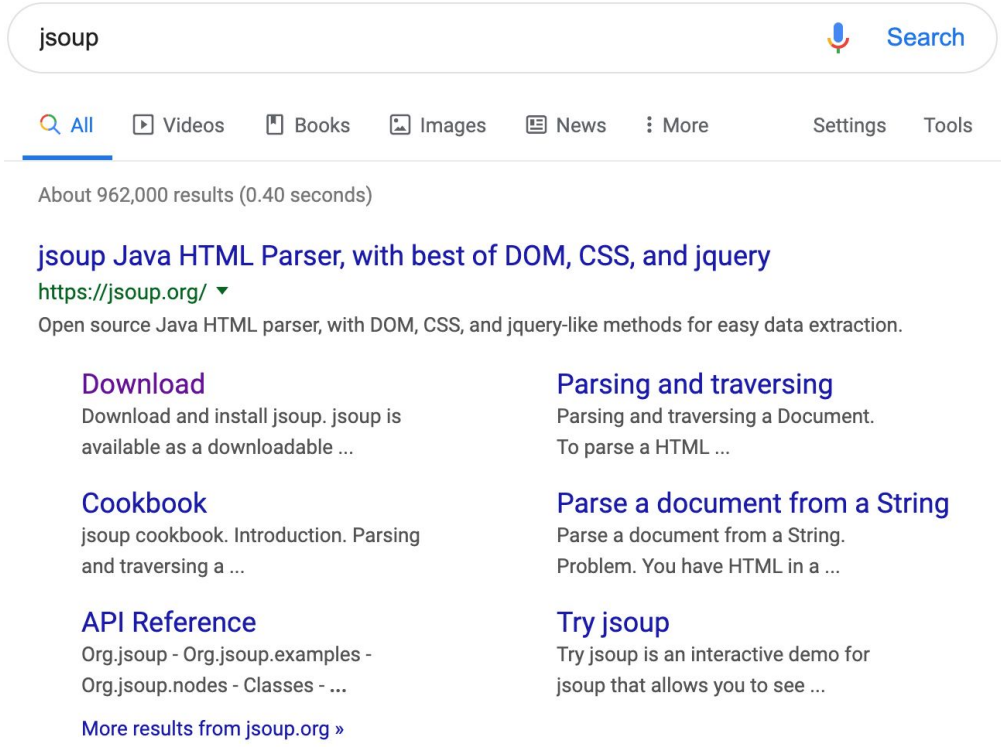
Press **Finish** when done!



Let's Get a .JAR

We're going to use the jsoup library to process HTML in a webpage. Let's go get it!

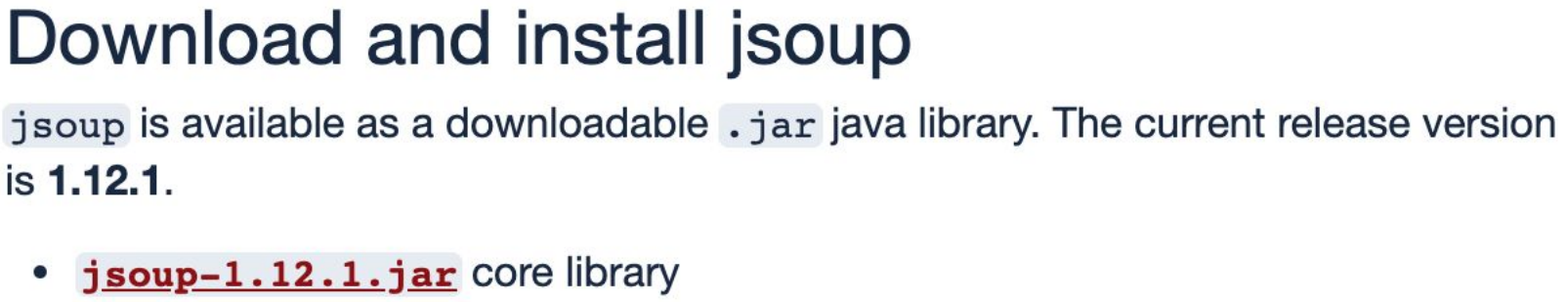
If you search "jsoup" online, you should find it! Click on **Download**.



Let's Get a .JAR

After you've clicked **Download**, you will see some links.

Click on **`jsoup-1.12.1.jar`** to download the .JAR file that contains jsoup!

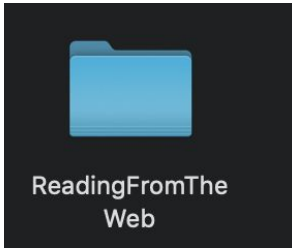


Let's Get a .JAR

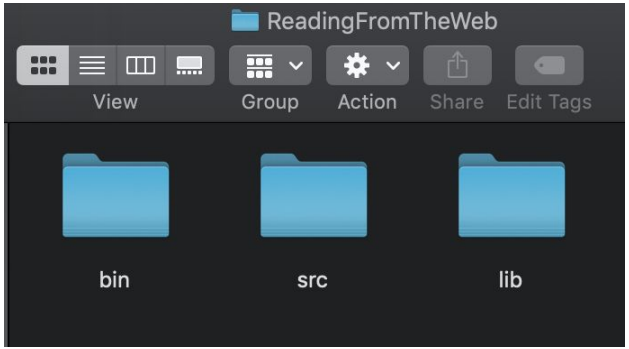
Let's put the .JAR file we just downloaded into our lib folder we just created!

We can drag it into our folder using Finder or Windows Explorer!

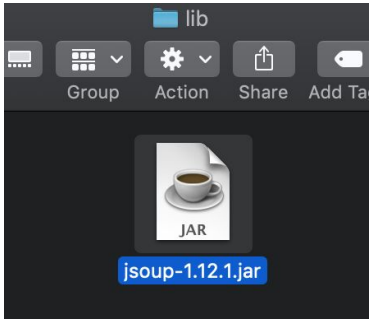
Step 1: Open your Folder



Step 2: Open lib



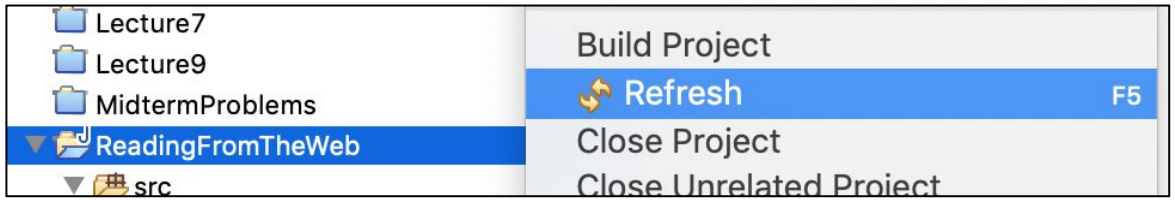
Step 3: Drag the .jar file into lib



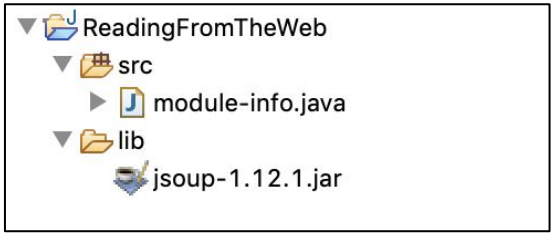
Refresh Our Project

Now, we can return to Eclipse and refresh our project!

Secondary click (two-finger or right click) on the project name and press **Refresh**.



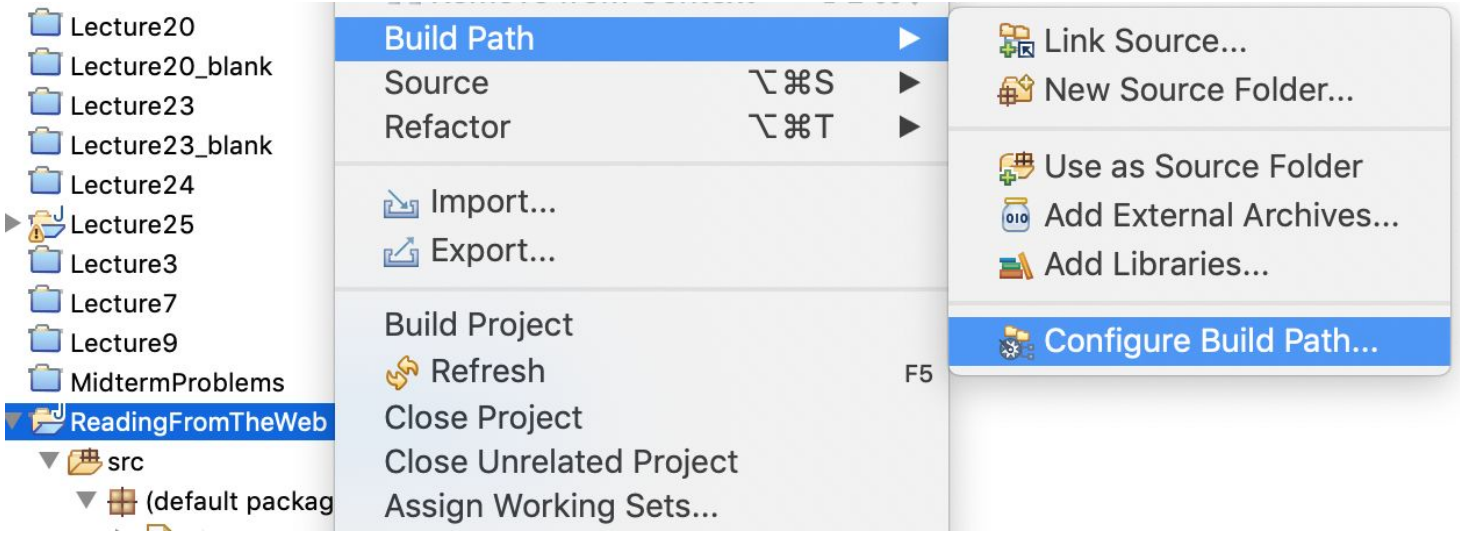
There's your .jar file!



Configuring Our Build Path

Even though we've added our .jar to our folder, we need to add the .jar to our Build Path.

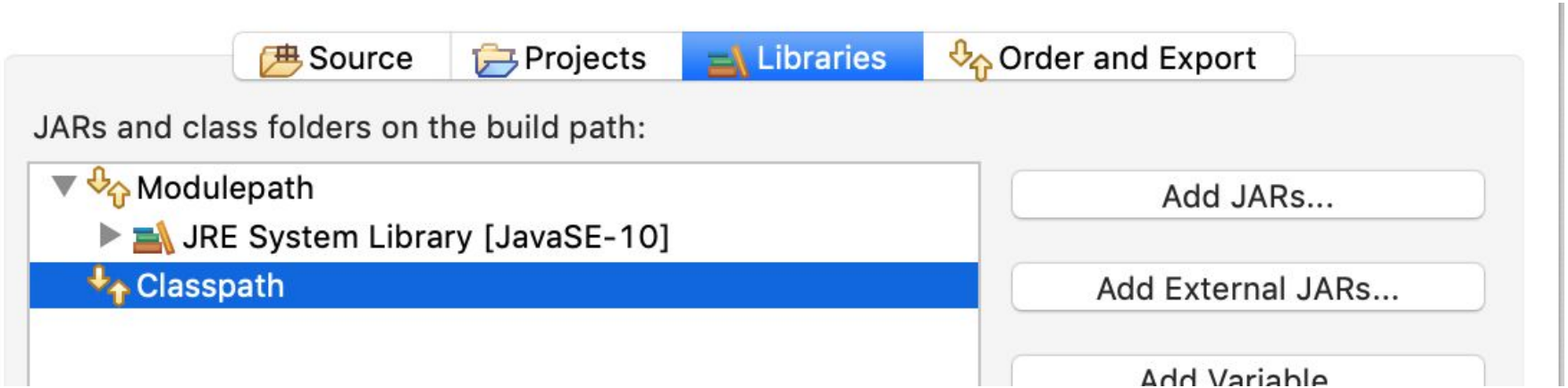
Let's **Configure Build Path...**



Configuring Our Build Path

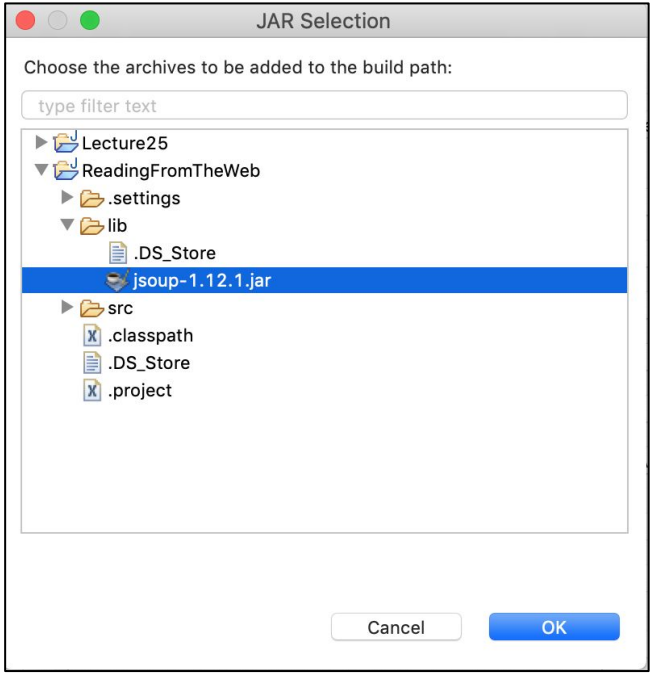
Click:

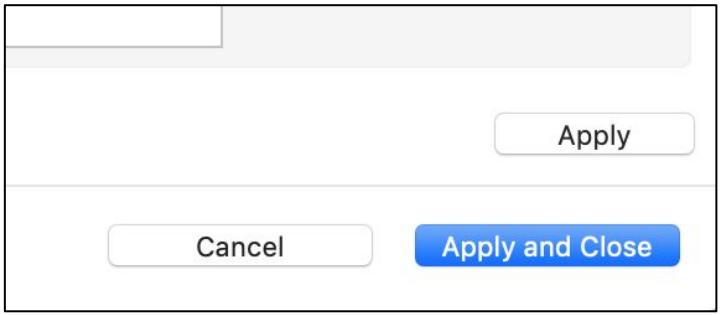
- **Libraries**
- **Classpath**
- **Add JARs...**



Configuring Our Build Path

Select our jsoup jar and click **OK**. Then, you can press **Apply and Close**.

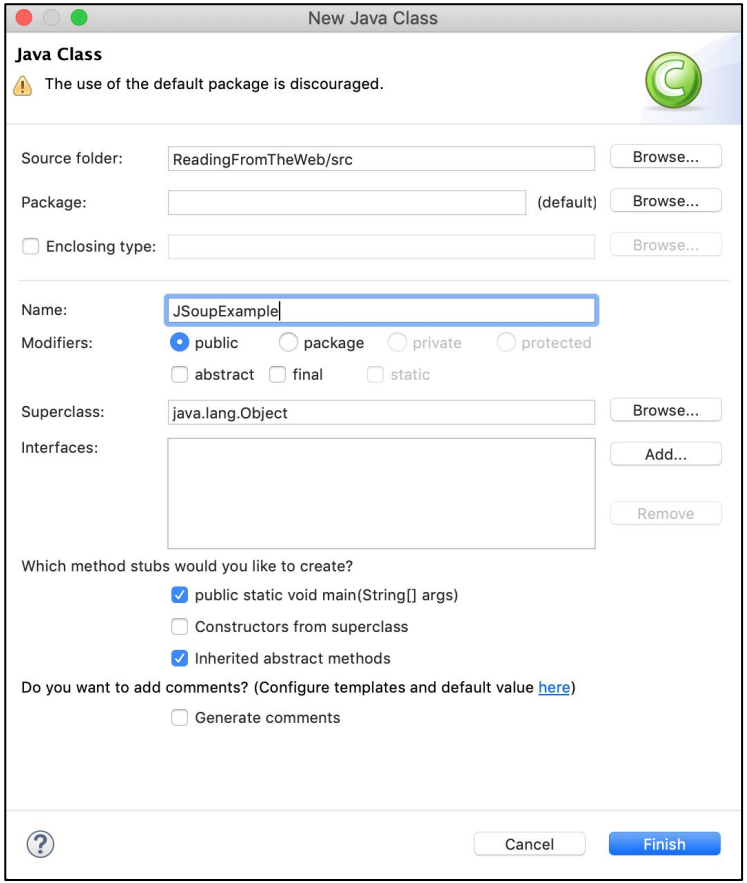




Creating Our Program Class

Now, we create our JSoupExample class so we can start coding!

We create this the same way we created a new project and a new folder!



Starter Code

While you could this program from Scratch there is a good starting point in the Starter Code.

You can code this portion in the Starter Code, or copy-paste JSoupExample from the Starter Code into your file!

Documentation/API

Using the Documentation from the JSoup website, we can create a program that will:

- Get HTML from an explore courses page about CS classes
- Print out all of the course titles for CS classes

Links to relevant documentation:

https://jsoup.org/cookbook/extracting-data/attributes-text-html

https://jsoup.org/cookbook/extracting-data/selector-syntax

Let’s Code It!

Plan for Today

- Review: Server & Client
- "Real" Java
- Console Program
- Basic Gui
- Creating Our Own Project!
- Example: Reading HTML with JSoup

Next Time: Life After 106A