CS 2300 Week 02 Lab

Install Java JDK and IDE (JetBrains Intellij IDEA). Do a few brief tasks.

0. Lab Assignments.

How to find the lab assignment. Each week's lab can be found at **Blackboard / Course Content / Lab Assignments**.

Lab Sections. In addition to the **lecture** (in IRVINE 194) there are **lab sections** (in Stocker 190). Students are required to attend the computer labs unless excused. The grade from the labs is a major part of the class grade. Labs are two hours long, but you can leave as soon as you submit your lab. It often takes less than one hour to finish the lab. If you are not done, you can submit your lab **within one week**. Labs will not be accepted later than after one week unless advance arrangements are made.

PC's and Macs. The computer labs have PC's with everything installed that you will need to write programs. However, you can also bring your own PC, Mac, or whatever, to the lab and use it. That is perhaps a better idea, because that way you can work on your assignments anywhere.

Today's lab will show you how to set up your computer for programming Java.

All detailed instructions are given for a PC. The equivalent Mac operating system commands are similar, but more like what a Mac user is used to doing. All Java code is exactly the same for a PC, Mac, or anything else.

1. Java Background.

Java is one of the most widely used programming languages for all **devises** (desktop computers, mobile computers, phones, game consoles, pretty much everything else).

1.1. **JVM**. All Java programs works by communicating to a **Java Virtual Machine (JVM)**, which in turn communicates with the actual computer chip and associated hardware. For any Java program to work, a Java **JRE** must be installed on the devise used. It is very likely that the Java JRE is already installed on your computer, so don't worry about this.

1.2. **Java Code.** The actual program you write is called **Java code**. The Java code interacts with the JRE to make your program do some work. The Java language has evolved over the last 25 years or so, going from version 1 to version 19. Currently, a new version comes out twice a year. Each new version adds additional features. Each version is **backwards compatible**, which means that older programs continue to work with newer versions of Java.

1.3. **JDK**. In order to write Java code, a **Java Development Kit** must be installed on your computer. For Java 1 through Java 8, you needed to install both the JDK and the **JRE** (Java Runtime Environment), which is basically the JVM. In more recent versions, the JVM is included in the JDK, so you no longer have to install both.

1.4. **IDE**. Java code can be written by any text editor, such as WordPad, NotePad, or even MS Word. However, using an ordinary text editor is **not a good idea**. It is vastly better to use an **Integrated Development Environment**. An IDE takes care of many details, checks for mistakes, offers suggestions on how to fix errors, suggests things to use, has keyboard shortcuts for everything, and lots of other good stuff. **It is the way to go**, and we will use one in this class.

In recent years, the **JetBrains** company's IDE's have become extremely popular. The IDE for Java is called **Intellij IDEA**, and there are other ones for other computer languages. Now that they offer a free community version, we will use it.

3. Install Java.

3.1. What is already installed?

On a PC, click WIN + E. I.e., hold down the Windows key (with the Window symbol) and click E. This brings up the Windows Explorer. Navigate to the **Windows (C)** link and click it. If you can't find it, keep clicking the up arrow, then **This PC**, and you can find it there. Then click **Program Files**, then **Java**. If there is no Java folder, you don't have any Java installed. In the Java folder on my computer, I see folders labeled:

jdk 1.8.0_311
jdk-11.0.13
jdk-16.0.1
jdk-17.0.1
jre 1.8.0_311

This means I have four different versions of Java JDK installed, and one JRE. I have uninstalled many earlier versions, and will soon get rid of some of the older versions still installed.

If you want to, you can browse around in these folders and check out the stuff in there. Don't change anything or you might have to reinstall.

3.2. **Install Java**. At some point, you might have to register a free user account from Oracle. Do it.

3.2.1. Browse to **Java.com**.

3.2.2. Click on **Developers**. This is in the small print on the second last line, near the bottom of the page.

3.2.3. Click on **JDK Downloads**. Choose **Java 17** if there is a choice.

3.2.4. Click on **Linux, macOS, or Windows**, depending on what kind of computer you have. If you have a PC, click on x64 installer. This will download a .exe that you click to install Java. You might have to deal with "Microsoft Store" crap. If this happens, choose to not have to install from MS Store. You will have to click **next** a few times. Choose all defaults.

3.3. Either click **close** or **next steps**. The latter will present some tutorial info that you can read if you have time.

3.4. Repeat the steps in 3.1 and see that Java 17 has been installed.

4. Install IDEA.

4.1. Browse to **jetbrains.com**. Click on **Developer Tools / Choose your tool**.

4.2. Click on **Intellij IDEA / download**.

4.3. Click on **Community / download**. This is the free version. Don't click on the 30 day trial of Ultimate version, or you will have to pay some $$$ in 30 days.

4.4. A rather large .exe file will be downloaded. This take a minute or so.

4.5. Click on the downloaded .exe file. This will cause a setup program to run. Accept all the defaults (unless you have a reason not to, and know what you are doing). Check all the boxes on Installation options (unless you have a reason not to). Click Finish. This might call for a reboot.

4.6. Check that there is now an **Intellij IDEA icon** that you can click on.

5. Check out IDEA.

Click the IDEA icon. The first time you have to click a couple things.

5.1. You should see "Welcome to Intellij IDEA".

5.2. On the left it should say Intellij Idea and have four choices: Projects, Customize, Plugins, Learn Intellij IDEA.

5.2.1. Click **Customize** and take a look at the options. If you want to, you can change the color scheme or font size, or whatever. You can also make the shortcut keys from other applications be used rather than the JetBrains ones. It is probably a good idea to choose the **NetBeans** shortcuts, because they are easy to use.

5.2.2. Click **Plugins**. Check out all the other things you can add. **Resist the temptation to add a bunch!!!**. Adding a bunch of other things clutters things up and is confusing for beginners. Add something else when you need it, and not beforef.

5.2.3. Click **Intellij IDEA**.

Click **Help**. Here are tons of resources.

Click on **Create your first Java Application**. This brings up a tutorial by Trisha Gee. Trisha is a **Java Champion**, kind of a
spokesperson for Java. She is also the primary spokesperson for JetBrains. Check her out on Twitter or blog: https://blog.jetbrains.com/author/trishagee/ Trisha is a star in the Java world.

6. Start your first IDEA program.

6.1. Close IDEA so that we get a clean start.

6.2. Click the IDEA icon to **launch IDEA**.

6.3. If the Welcome screen appears, click **New Project**. Otherwise, from the main menu at the top left, select **File** and then **New Project**.

6.4. On the New Project wizard, select **Java** (which is probably the top one, and preselected). At the top middle is a box to select the **project SDK** (software development kit). It probably says **17**. If you need to, you can change this to use another version.

6.5. We are not going to use any additional stuff, so click **next**.

6.6 We are not going to use a template, so click **next**.

6.7. Type in a name for the project, such as **HelloWorld**. Then hit the **Enter** key.

6.8. If you want to, you can change the default location where the project will be stored. Don't do this unless you have a reason to. Click **Finish**.

7. Create a package and a class.

Java programs consist of one or more **package**s. Each package contains one or more **classes** (or other class-like things).

The **Project Window** is on the upper left. Click on the project name. Notice the word **src**, which stands for the **source code** folder. This is where stuff that you write goes.

7.1. Right click on the **src folder**, select **New**, select **Java Class**.

7.2. In the **Name** field, type: **lab01.HelloWorld** and hit the enter key.

This will create a **Java class** named HelloWorld inside a **Java package** named lab01.   The name of the class must be one word (no blanks) and it should always start with an **uppercase letter**. The name of the package is usually all lowercase letters and perhaps digits. The package name can contain extra dots inside it, but avoid doing this unless you have a reason to.

7.3. The last step will cause an **edit window** to open up with this in it:

```
package lab01;

public class HelloWorld {

}
```

This is actually a very small legal Java program, but it does not do anything.

8. Getting a program to work.

Here is the secret to getting programs to work. Read this often!!

Start small. I.e., start with a program that does nothing but say “Hello World”, or something.

Do one little thing at a time. Add one little thing at a time to the existing program. Every time you add a few lines of code, enough that it is a legal program, run it and see if it works. If not, fix it. Never add more stuff until the previous part works.

That’s it. This is the only way to get programs to work. A common mistake for beginners is to write a big program full of errors and then trying to fix it. You never will fix it, it will never work, no one can help you with it, and you will have to delete it and start over.

8.1. **Comments**. Some information in a program is intended for the human reading the program, as opposed to the compiler that will turn the code into an **executable program**. Such information is called a **comment**. There are three ways to put a comment into a Java program. Here we will use a **Javadoc comment**. Add several lines so that the code looks like this:

```
package lab01;

/**
 * @author Ralph Kelsey
 * Lab 01
 * 2022 08 24
 */


public class HelloWorld {

}
```

Use your name and the current date. Note the colors used by IDEA, which survived when I copied and pasted the above code into this MS Word document.

The text starting at the /** and ending at the */ is a **Javadoc** comment. These will be discussed later.

This comment consists of a **name**, a **title**, and a **date**. It is a good idea to always start a program with at least this much information.

8.2. The **braces** { } define a **block**, which so far is empty. When you have a block within a block, the inner block should be indented 4 spaces to make it easy to read. IDEA will do this for you.

We will put code in the block. We can do this "by hand", or we can let IDEA help us out.

9. Add a main method.

Put the curser right after the opening brace {, and type **main**. At this point, IDEA will offer to write a **main method** for us. Double click on the suggestion, and now you will see this:

```
package lab01;

public class HelloWorld {
    public static void main(String[] args) {
    }
}
```

Read this a couple times. You might have to reproduce this **on the first quiz**!!

A **method** is a unit of code that does some work. The word **function** is used in some computer languages for the same idea.

A **block** is a something surrounded by a pair of braces, like this: **{ }.** Notice that when you type the left brace, **}**, and hit `enter`, the right brace **}** automatically appears in the correct location.

Notice the code above has a pair of **nested blocks**, i.e., one is inside the other. The inner block is indented four spaces.

9.1. The **main method** is "where the action starts" when a program is run. This program can be run as is. The main method will run, but there is nothing in it, so nothing will happen.

Put the curser inside the main method, and type:

```
System.out.println("Hello World");
```

Notice that when you typed the first ", the second one automatically appeared, and the curser is between the pair. This is a very good thing, because there is hardly ever a reason to only type one ". The same thing happens whenever you type a (, [, {, <, or '. These symbols always come in pairs, and it is a good thing to always put in both parts of the pair at the same time so you don't forget the closing symbol. Most IDE's do this for you, which saves a lot of mistakes.

The text between the two " signs is called a **string**, or sometimes a **c string**, to distinguish it from another type of string.

This statement should automatically be indented 8 = 4 + 4 spaces. Put the curser after the semicolon and hit enter. The curser should go to the next line and be under the first letter. Add another line:

```
System.out.println("from Ralph Kelsey");
```

Except put your own name instead of Ralph Kelsey.

The main method consists of two **print statements**. All statements must end with a **semicolon**. These two statements are **executable statements**, lines of code that **do something**.

10. Run the program.

Enter **ctrl + shft + f10**. This means hold down both the **Control** and **Shift** keys, and press the **F10** key. On some computers, you have to also hold down the **fn** key when you an f key, like f10. On a Mac, use **ctrl + R**.

In a moment an **output window** will open at the bottom of the monitor. It will print the two strings, like this:

```
Hello World
from Ralph Kelsey
```

11. Numbers

Skip a line after the `println` statement. Now add the following code:

```
int x = 4;
System.out.println("x = " + x);
x = x + 10;
System.out.println("x = " + x);
x += 20;
System.out.println("x = " + x);
```

Figure out what the + and also the += operators do.

12. Strings.

A very important class in Java is the **string class**.

After the previous code with numbers, skip a line, and type the following code into your program:

```
String a = "Hello";
String s = " ";
String b = "World";
String e = "!";
String message = a + s + b + e;
System.out.println(message);
```

Run the program and be sure it works.

Notice that four strings `a`, `s`, `b`, and `e`, were concatenated into the string named `message`, which is then printed out.

Go to Google on your browser, and google on **Java 17 String**. Click on the first thing that comes up. This will be the official **API (Application Programming Interface)** of the `String` class. Every built in Java class has a similar API, which is the standard way to document a class in Java. This one is probably by far the biggest such API. Take a quick glance at all the stuff in the API. We will use the API whenever we need information about a Java class.

13. Submit the program to Blackboard.

How to submit your lab. Be sure your name is in the comments at the top of the code.

Labs are usually given as **Blackboard assignments**, which includes a submit feature. On the Blackboard assignment, look for a label that says **attach file**. Click **browse computer** and find the file that holds your code.

Your code can be found several layers deep in the file system. For example, the code on my computer for this project is at:

```
D:\Documents\JetBrainsProjects\HelloWorld\scr\lab01\HelloWorld.java
```

I happen to keep data in a `D` drive. Yours is more likely to be in the `C` drive. You might be able to get to the code through the IDEA Project panel (on the left side of the IDEA screen).

Click on the .java file to attach it, then click the **submit** button. These will be graded online.