CSC-133 :: CN1 Installation and Assignment 0

Codename One Installation

CN1 can be installed to one of the following IDEs: Eclipse, NetBeans, or IntelliJ IDEA which run on various operating systems. You can use whatever you want, however, it is not my responsibility to debug your installation problems, resolve why your jar doesn't work on the command line, or any other numerous issues that arise from this. Generally speaking, the primary issue with submissions not working is students not following directions closely. I suggest that you read this document multiple times and ask questions in the **appropriate discussion forum**to make sure that you are conforming to the requirements.

While you are free to use the IDE of your choice, or even no IDE, you must have at least JDK 11 installed.

Java JDK

Before you begin with the IDE installation you will want to make sure that you have an appropriate JDK. If you already have a Java **JDK** installed that is compatible then you may use it and won't need to install a different JDK.

So what does "compatible" mean?

1. JDK 11 or Later
2. Distributed with JavaFX
   - JavaFX is needed for media in the simulator. You will be playing sounds, so you will need JavaFX

I'm not going to get into the nitty gritty details of the matter, but, JavaFX was distributed differently after Java 8 and may present problems on Linux even with Java 8. I expect that your submitted code will run, compiled with a Java 11 or later distribution.

I will not manually dig into your code to see why it doesn't work with a Java 11 or later JDK!

So, now you know that you need Java 11, which one do you get. There are multiple distributions of Java 11. Here are some of the main distributions. Note that Java 11 is recommended over Java 14 as it is tagged as LTS, meaning *Long Term Support.* The value to you as a student is that it will probably meet your needs over several semesters without needing to be reinstalled. All bets are off with the latest release. In any case, you may choose Java 14 if you wish. I will run any grading scripts with Java 14 so you will be compatible with anything from Java 11 to Java 14. Now, on to the distributions:

1. **The official Oracle distribution**
   - **NOT RECOMMENDED** : This is the official distribution from Oracle. The link is to JDK14, the current release. You may download the current release without an Oracle account, a pleasant change from earlier releases, however, you will have to agree to their technology licensing agreement. Because of the restrictive license, I cannot recommend the standard distribution. In addition to having a restrictive license, in the past Oracle has installed very difficult to remove updating software. The one thing that it has in it's favor is that it is among the easiest to

install.

2. **The official Oracle open distribution**
   - **NOT RECOMMENDED :** This does not have the restrictive license of the standard JDK distribution. However, AFAIK, it does not properly bundle JavaFX. I believe that the Ubuntu package installed distributions are based on this.

3. **Adopt Open JDK**
   - **NOT RECOMMENDED :** This does not have the restrictive license of the standard JDK distribution. However, AFAIK, it does not properly bundle JavaFX. You can download versions with an install and different java VM implementations. On the upside, it does come with proper installers.

4. **Azul Zulu Open JDK**
   - **RECOMMENDED :** This does not have the restrictive license of the standard JDK distribution. However, you can download a variant that does properly bundle JavaFX. The only downside is that it is bundled in a zip file and you will have to set the path to the JDK yourself. Make sure that under *Java Package* that you select JDK-FX.

I put my Java files underneath a JDK directory somewhere on your system. There are issues with putting it under "Program Files" on Windows if you want to use the Java 11 single file execution feature, but we aren't likely to need that in this class. I'm going to assume that Linux users are savvy enough with IT issues to know where to install a JDK. Post a message on **the discussion forum**if you want to chat about it. If you aren't sure then I would just install them under a JDK directory in your user directory. This will also work on windows and you don't have to worry about the Java 11 single file execution. The only downside to this is that it will only work for your user.

After you have Java installed and the path pointing to the bin directory, then you should be able to open a command prompt or terminal window and type

```
java -version
```

On my Windows machine it looks something like this. I'm currently using Zulu 11 with Java FX. If you aren't able to get the version to show up, then it's almost certainly just your path. Don't go any further until you get this working.

```
C:\>java -version
openjdk version "11.0.8" 2020-07-14 LTS
OpenJDK Runtime Environment Zulu11.41+23-CA (build 11.0.8+10-LTS)
OpenJDK 64-Bit Server VM Zulu11.41+23-CA (build 11.0.8+10-LTS, mixed mode)
```

Finally, do not muck with your JAVA_HOME environment variable if it is set for your JRE. You don't need to change that to use the *java* command from the command line or to use a different JDK inside of an IDE.

Now that you have a JDK working properly, you need to install an IDE of your choice. I use IntelliJ IDEA and I will not be installing either Eclipse or NetBeans to verify issues. I will do my best to help you based on my past experience with these other IDEs, but if you have no preference and you are not strong with IT issues, then I highly recommend that you install IntelliJ.

InteliJ IDEA Installation

Download and install the Community Edition from:

https://www.jetbrains.com/idea/download/#section=windows

Install the CodeName 1 plugin from the Pluigin Center or from the following URL.

https://plugins.jetbrains.com/plugin/7357-codename-one

From within the IDE, you will need to go to file->Project Structure to change the JDK for your project. Go to Project:Project SDK and click on the down arrow to select **+ Add SDK.** Choose JDK and navigate to your installed JDK.

Eclipse Installation

Download and install Eclipse from:

https://www.eclipse.org/downloads/packages/installer

Install the codename one plugin from within the Eclipse Marketplace.

Add your newly installed JDK to Eclipse,

In this course your programs must be contained in a CN1 project with a specified name for each project. For example, for example, for Assignment 0, the project name will be A0Prj, for Assignment 1 the name will be A1Prj, for Assignment 2, A2Prj, etc.

Create a new project as described in the videos above and call it A0Prj. You will put your project in the package **org.csc133.a0**, adjust the assignment number for each assignment, and you will use the starting class **AppMain** for all assignments.

*This naming scheme is a **REQUIREMENT**. Don't waste your time trying to ask for an exception when you didn't follow this requirement. If your code doesn't run because you didn't pay attention to the details of this document, then my grading software will give it a **zero**.*

Compile and run the Hello Codename One default *"Hello World(Bare Bones)" template* and a *"native"*theme. You should be able to get it to run in the simulator. Follow the instructions in the above videos, but, don't bother editing it to add the dialog box. We will be adding some code below.

Now, verify that your program works properly from the command prompt before moving on to the next step.

For Eclipse, first make sure that the A0Prj.jar file is up-to-date. If not, under eclipse, right click on the dist directory and say “Refresh”.

For IntelliJ IDEA, you will have to complete some additional steps to create a jar. From File->Project Structure, choose *"artifacts"* and click on the "+" to add an artifact. An artifact is something that is made as part of the build process. Most often we want to create a jar that executes, but not in this case. The reason is that the main class that executes is in the JavaSE.jar that is provided by CN1. So, counterintuitively, you want to choose JAR->Empty. Give the jar the same Name as the project, so for this project **A0Prj**. It will give a default output directory just leave that alone. Select "include in project build" so that the jar is automatically updated whenever you build the project. Finally, you will need to tell IDEA what to include in the Jar. In the right hand pane under *"Available Elements"* you should see *"A0Prj compile output"* this represents all of the class files and resources that are a part of your project. Click on it and it should move to the left indicating that it will now be included in the jar file.

For this project you only need the jar to contain class files. Since there is only one source file, you will submit your source as a separate file. More on that below. Now you will want to bring your command terminal or shell up and navigate to your project directory.

For Intellij IDEA the jar file is placed in your artifacts directory , in Eclipse it's placed in the dist directory. The command line(s) given below should execute the application.

Note that the only difference between Windows and Linux is the colon vs the semicolon and forward slash vs back slash in the path names. As far as I know, the command lines for Linux will also work on a Mac. I don't currently have access to a Mac that I can run this on, so, I'm not able to personally verify the command lines. These have been tested on Windows and Linux. Mac users, do not send me a panicked email telling me that you think that they're wrong. Post in the **discussion forum.**While there may be some mistakes here, it's unlikely and it's far more likely that any issues on a Mac have to with something other than this command line syntax.

IDEA on Windows

```
java -cp  out\artifacts\A0Prj\A0Prj.jar;JavaSE.jar com.codename1.impl.javase.Simulator org.csc133.a0.AppMa
in
```

IDEA on Linux/Mac

```
java -cp  out/artifacts/A0Prj/A0Prj.jar:JavaSE.jar com.codename1.impl.javase.Simulator org.csc133.a0.AppMa
in
```

Eclipse on Windows

```
java -cp dist\A0Prj.jar;JavaSE.jar com.codename1.impl.javase.Simulator org.csc133.a0.AppMain
```

Eclipse on Linux/Mac

```
java -cp dist/A0Prj.jar:JavaSE.jar com.codename1.impl.javase.Simulator org.csc133.a0.AppMain
```

Substantial penalties will be applied to submissions which do not work properly from the command prompt! In short, you will earn a zero for the auto-graded portion of your score!

Assignment 0

First, there is very little programming in this assignment. This is mostly an assignment to get your toolchain setup and to make sure that everyone can write a few lines of Java.

You will want to open up your A0Prj that you created from above. If you made a mistake, don't be afraid to just recreate the project.

Directly beneath the generated imports add the following block of code.

```
import com.codename1.media.Media;
import com.codename1.media.MediaManager;

class Sound {
    private Media m;
    public Sound(String fileName) {
        try {   m = MediaManager.createMedia(Display.getInstance().getResourceAsStream(getClass(),
                    "/"+fileName), "audio/wav");
```

```
        } catch(Exception e) {}
    }
    public void play() { m.setVolume(50); m.setTime(0); m.play();}
}
```

This small block of code gives you the ability to play a sound. The important part of this assignment is that this actually runs with your chosen JDK. If you don't have JavaFX, the code that we will add below will exit with an error. Note: the code above has been stripped down to bare essentials. It's not good practice not properly respond to exceptions, in short, we won't be coding like this in this course.

In your AppMain class, modify the *start()* method so that it looks as follows:

```
public void start() {
    if(current != null){
        current.show();
        return;
    }
    new Sound("PD_DoorEntry.wav").play();
    Form hi = new Form("Hi World", BoxLayout.y());
    hi.add(new Label("Hi World"));
    hi.show();
}
```

This will play a sound when you run the application. Before you compile and run, download the **public domain sound called PD_DoorEntry.wav**and drop into the main source directory of your project.

Build and run you project, and execute the jar file from the command line as above. If it does not play the sound, don't progress any further until you have the sound working.

Some Actual Programming

Now that your environment is working. Add some interesting text to the project window. Try to create an interesting image from text. It could be a poem, some ascii-art, a short story, an ascii fractal, doesn't matter. Whatever you think represents you. I will say though, that effort counts here. I don't expect you to be an artist, but don't phone it in if you want full points! This should be a low stress programming exercise, where you are just going to add more of the lines like the following, perhaps wrapped in loops, whatever:

```
hi.add(new Label("Hi World"));
```

Don't spend too much time on this part of the assignment it's mostly meant to get you to go through a few compile cycles so that you get familiar with things.. If you want to, feel free to explore the CN1 API a bit, and use whatever you like to pretty up the page. If you want more screen real estate, then choose a different skin from the simulator. You can download new skins from the web from within the simulator. When you're happy with how it looks, build your jar and make sure that it runs from the command line. Take a clean screenshot of just the simulator. Take a minute and do a good job. Now, post that image to the **Lab 0 discussion topic**.

You're almost done. On the **main Lab assignment** you are going to submit two files. Submit your jar file and your AppMain.java file. Submit them separately. If I cannot read your .java file in Speedgrader you will not receive any points for that portion of the assignment.