

Chapter 1: How to run the Program

Chapter 2: UML Diagram of Classes and Class/Methods explanations

Chapter 1:

How to run the program

This version of the code is an incomplete version of the program. You will have to implement the code as if you are using it as a project. This project is made using Java.

This will require you to install the latest version of JavaFX and be able to use the code using GitHub and providing it to the IDE of your choice. We will be using Visual Studio Code to perform our implementation.

These are the resources that you will need:

Java JDK: <https://www.oracle.com/in/java/technologies/downloads/>

Visual Studio Code: <https://code.visualstudio.com>

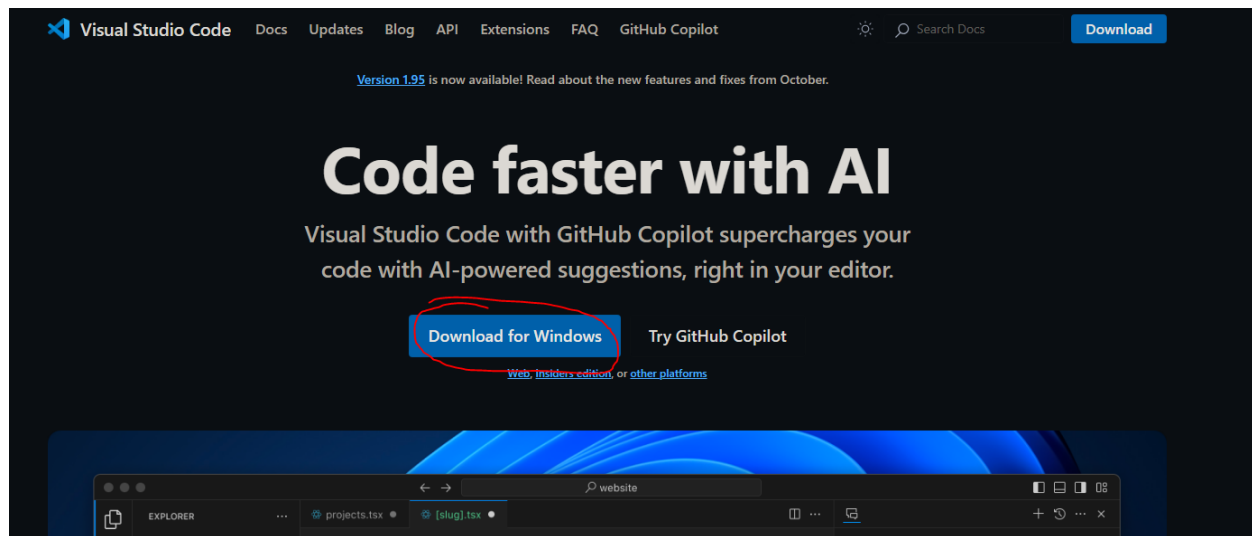
JavaFX: <https://openjfx.io>

GitHub Desktop: <https://desktop.github.com/download/>

If you have already installed everything needed, skip these steps.

Installing Visual Studio Code:

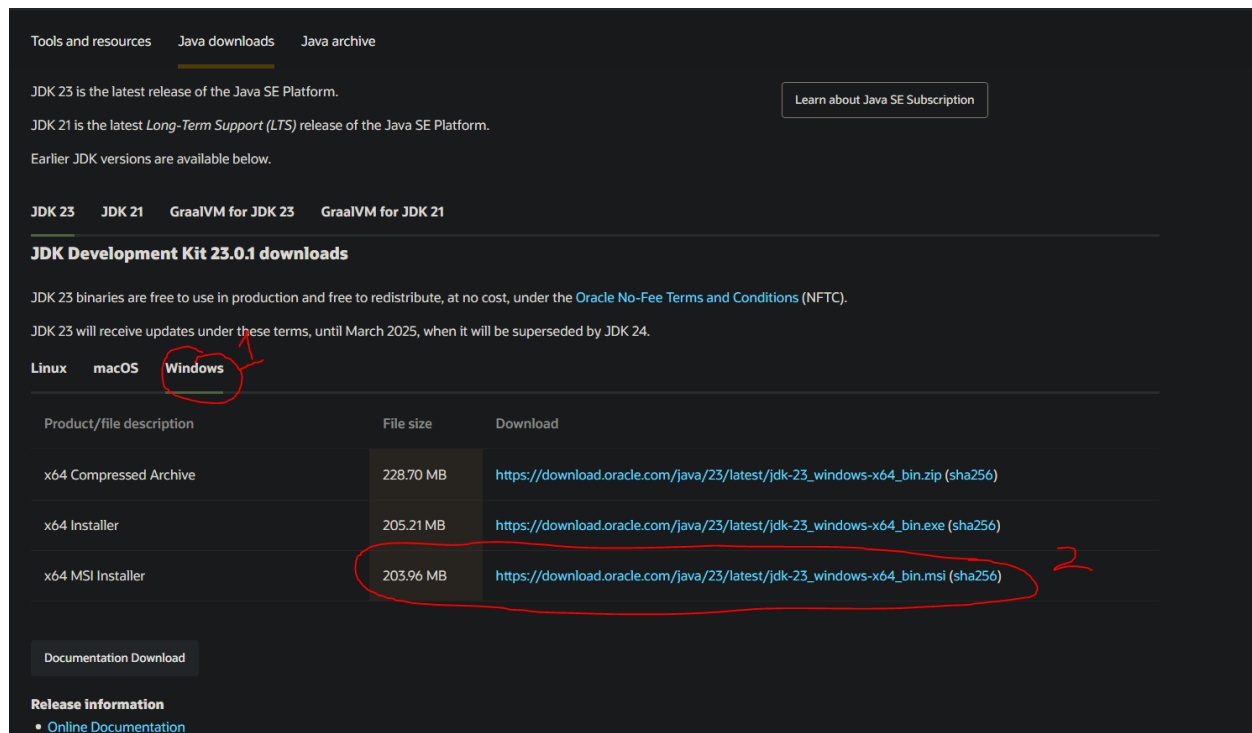
Click the link to [Visual Studio Code](#) above and click this button:



It will automatically start installing the program. When it's done, run the program and initialize your preferred settings. When the installation is done, we will now download Java SDK.

Downloading the Java JDK to be used in Visual Studio Code:

Click the link to [Java JDK](#) and press your current platform of your device. For windows users you will follow this:



Only 64-bit processors are supported. If you are running a 32-bit processor, you will not be able to use Visual Studio Code.

If you are unsure what processor you have, you can type “System Information” in the taskbar and find this line:

Item	Value
OS Name	Microsoft Windows 10 Home
Version	10.0.19045 Build 19045
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	DESKTOP-6SP4LVR
System Manufacturer	ASUS
System Model	System Product Name
System Type	x64-based PC
System SKU	SKU
Processor	AMD Ryzen 9 5900X 12-Core Processor, 3701 Mhz, 12 Core(s), 24 Logical Pro...

This line will tell you what type of processor you have.

For Apple/Mac users you will follow this:

Tools and resources Java downloads Java archive

JDK Development Kit 23.0.1 downloads

JDK 23 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 23 will receive updates under these terms, until March 2025, when it will be superseded by JDK 24.

Linux **macOS** Windows

Product/file description	File size	Download
ARM64 Compressed Archive	226.30 MB	https://download.oracle.com/java/23/latest/jdk-23_macos-aarch64_bin.tar.gz (sha256)
ARM64 DMG Installer	225.79 MB	https://download.oracle.com/java/23/latest/jdk-23_macos-aarch64_bin.dmg (sha256)
x64 Compressed Archive	228.87 MB	https://download.oracle.com/java/23/latest/jdk-23_macos-x64_bin.tar.gz (sha256)
x64 DMG Installer	228.39 MB	https://download.oracle.com/java/23/latest/jdk-23_macos-x64_bin.dmg (sha256)

Documentation Download

Release information

- Online Documentation

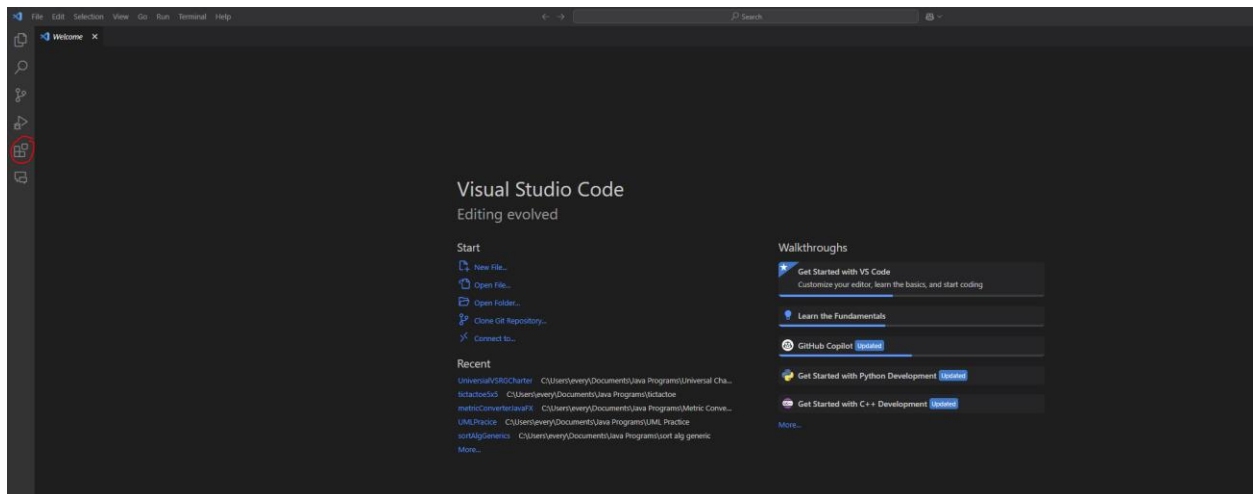
Those with ARM64 processors will use the ARM64 DMG Installer. Those with 64-bit Mac processors will use the x64 DMG Installer.

If you are not sure what processor you have, you can follow this article to find the processor:
<https://vectorlinux.com/is-my-mac-arm64-or-x64/>

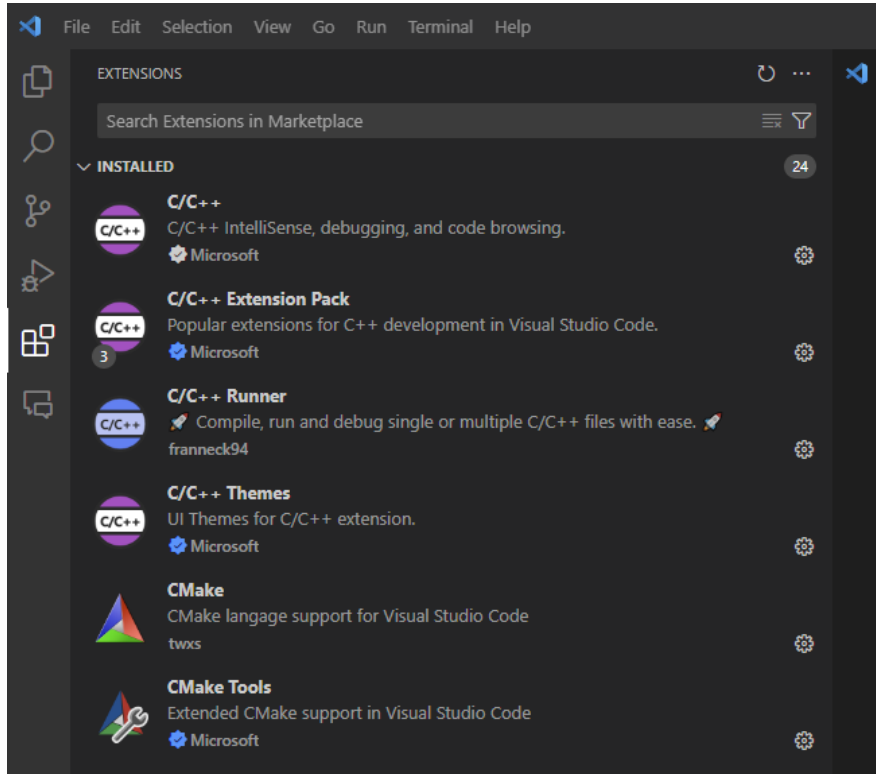
With this out of the way, run the program and proceed to execute the program. Where you define the folder will define the location of the JDK. Visual Studio Code should recognize the location of the JDK if you did the default location.

Installing Java Extensions in Visual Studio Code

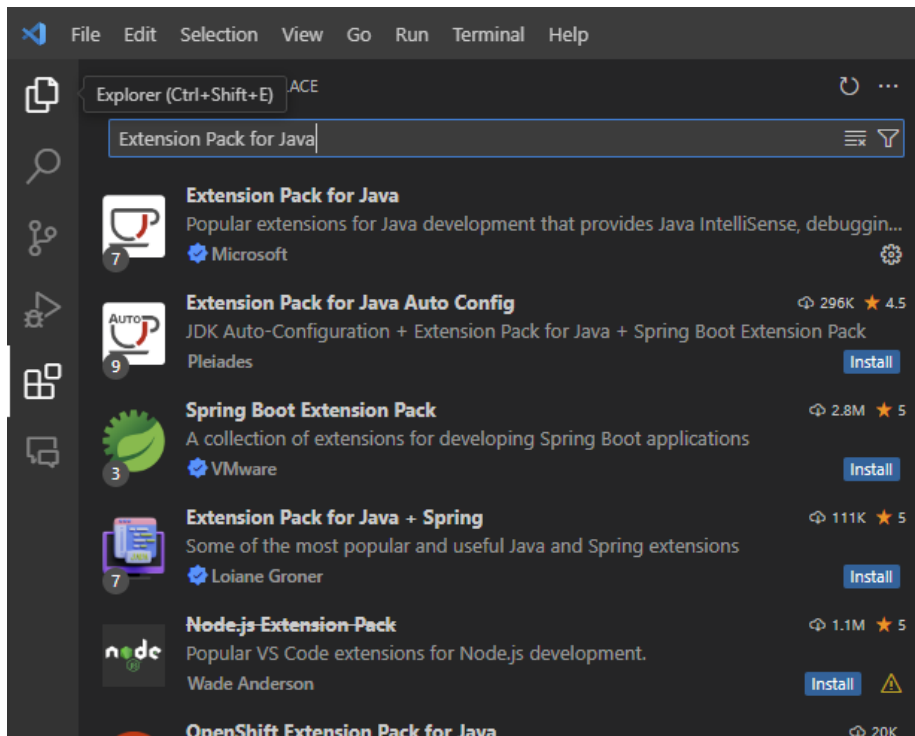
Installing extensions will make your life easier when handling code. Open Visual Studio Code and look at the left-hand bar:



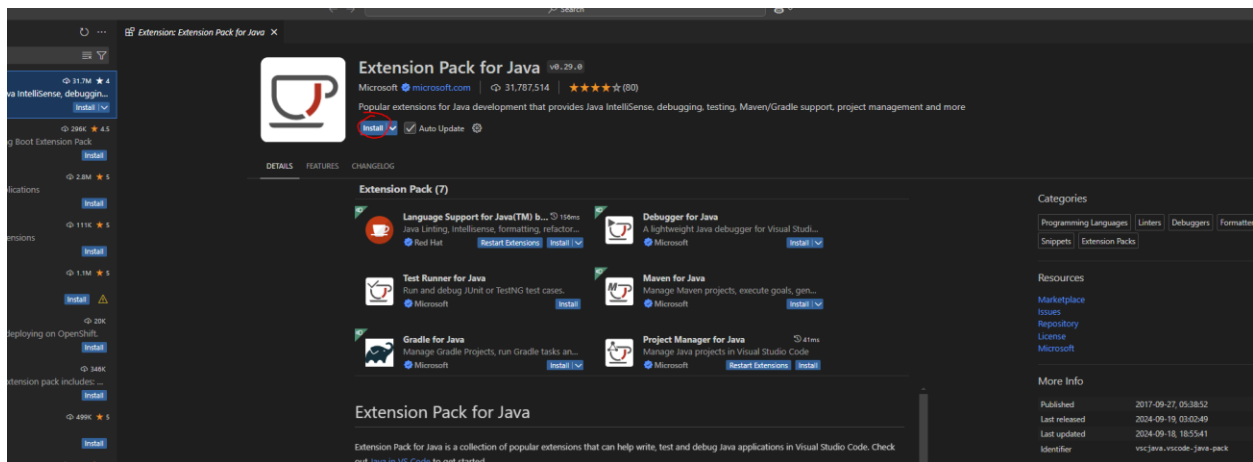
The icon that is circled is the extensions tab. Click on this tab and this will pop up:



Type “Extension Pack for Java” in the search bar:

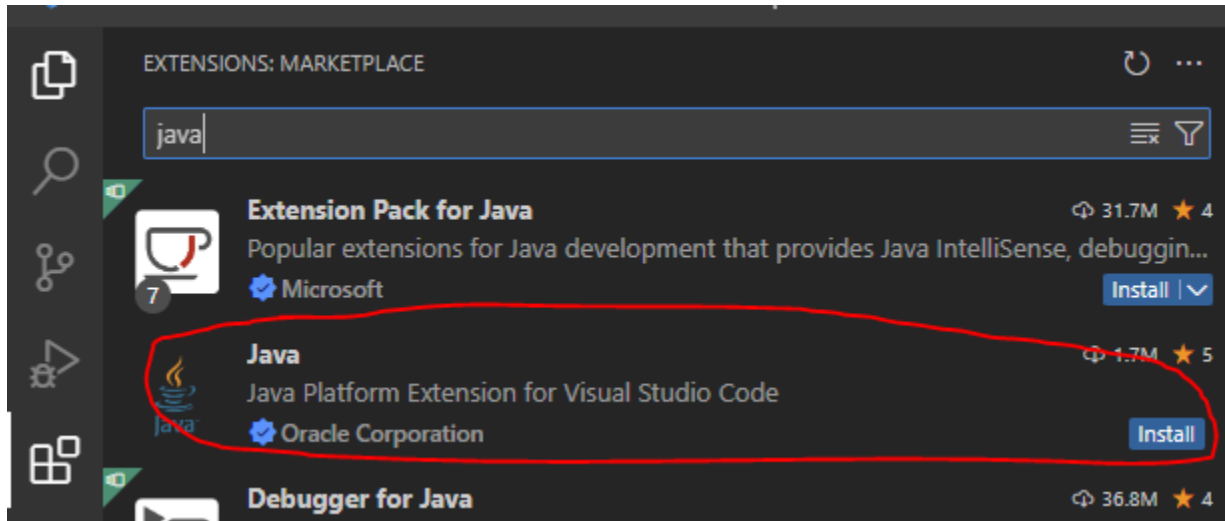


Click on the first result and install this extension:



Important Note:

Be careful **NOT** to install the Java extension:



An issue will arise when the JDK and this extension conflict with each other. When using JavaFX, everything under JavaFX in code will be underlined red. It will act like an error; however, your program will run. The issue with this is that you can have potential errors in your program and will not be able to identify them because the symbol of JavaFX “does not exist”. Avoid this problem by not installing this extension. If you do have it installed, uninstall it and use only the JDK one, which Visual Studio code already can detect.

Installing/Extracting JavaFX

JavaFX is given as a compressed zip folder. Windows already has a built in unzipper so be prepared to do this process.

Click the link to JavaFX and click on this button:

JavaFX

JavaFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java. It is a collaborative effort by many individuals and companies with the goal of producing a modern, efficient, and fully featured toolkit for developing rich client applications.



Download

JavaFX runtime is available as a platform-specific SDK, as a number of jmods, and as a set of artifacts in Maven Central.

DOWNLOAD



Develop

JavaFX, also known as OpenJFX, is free software; licensed under the GPL with the class path exception, just like the OpenJDK.

LET'S DO IT!

You will be followed to:

The screenshot shows the JavaFX website with a blue header containing navigation links: Products, Developers, Pricing, Services, Insights, and Contact. Below the header, the page title is "JavaFX". The "Roadmap" section features a table with the following data:

Release	GA Date	Latest version	Minimum JDK	Long Term Support	Extended or custom support	Details
24	March 2025	early access	21	no		
23	September 2024	23.0.1 (October 2024)	21	no	upon request	details
21	September 2023	21.0.5 (October 2024)	17	yes	upon request	details
17	September 2021	17.0.13 (October 2024)	11	until September 2026	upon request	details

Below the table is a link: [\[show older versions...\]](#). A list of bullet points follows:

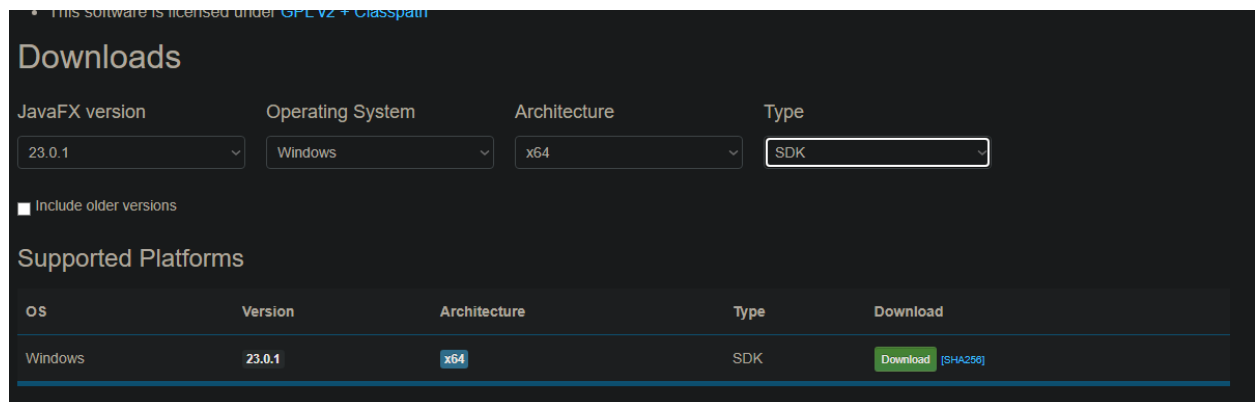
- All dates on this page are informative. [Contact us](#) for a personalised quote for our JavaFX Long Term Support (LTS) service.
- Releases in grey don't receive updates anymore. We strongly encourage all our users to use either the latest version (currently 22) or the latest version of one of the LTS releases (currently 17 and 21).
- The JavaFX runtime is available as a platform-specific SDK, as a number of jmods, and as a set of artifacts in [maven central](#).
- The OpenJFX page at [openjfx.io](#) is a great starting place to learn more about JavaFX.
- This software is licensed under [GPL v2 + Classpath](#).

The "Downloads" section has four dropdown menus: "JavaFX version" (set to 23.0.1), "Operating System" (set to [any]), "Architecture" (set to [any]), and "Type" (set to [any]).

Scroll down to here and select the platform you are on and select the version:



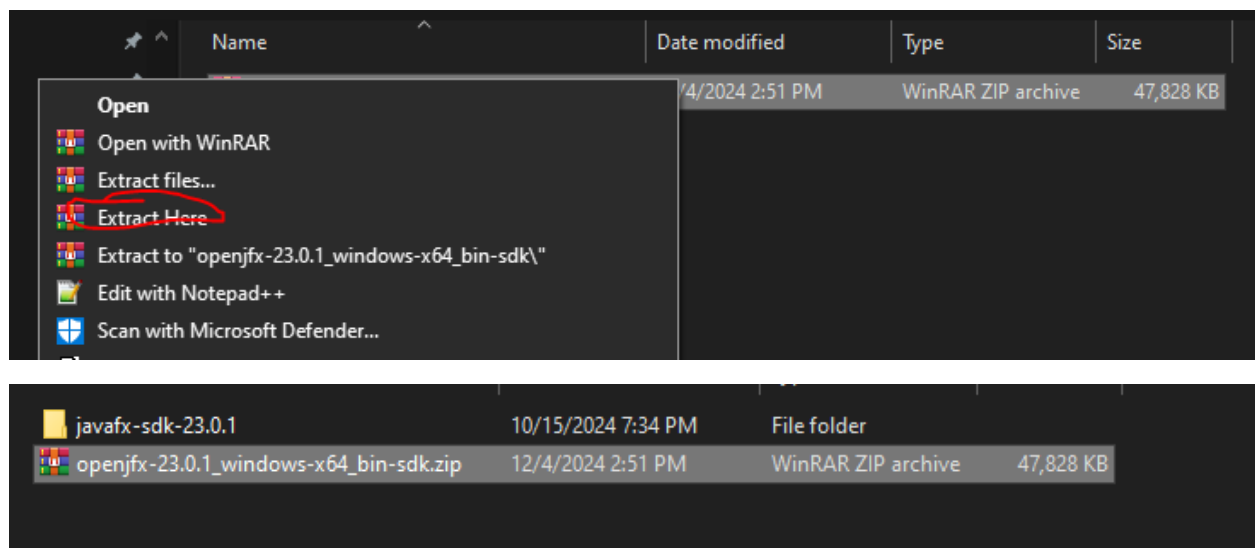
Since I am using windows, mine looks like this:



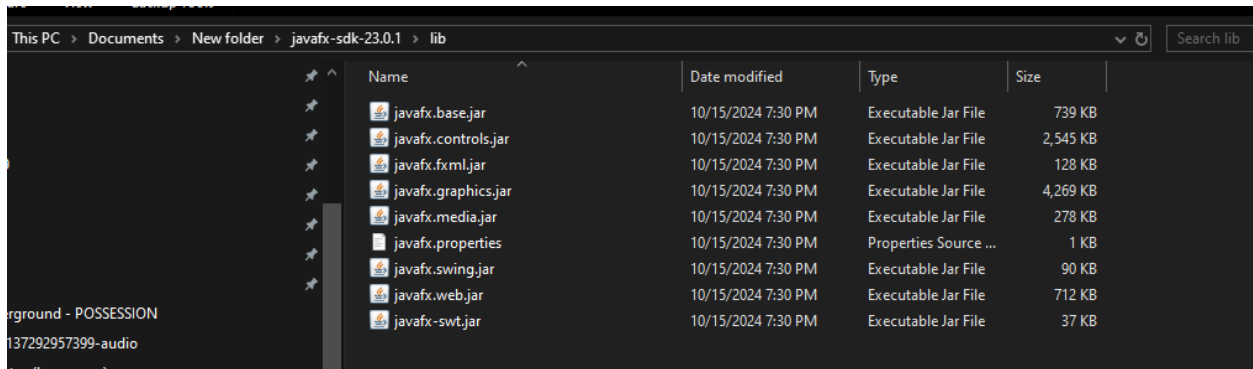
Press the green download button to download the SDK.

Once you have downloaded the file, we will extract this zip file to wherever we want:

I am using winrar, but you should have an equivalent based on what windows has given you.



Navigate inside this folder and open the lib folder:

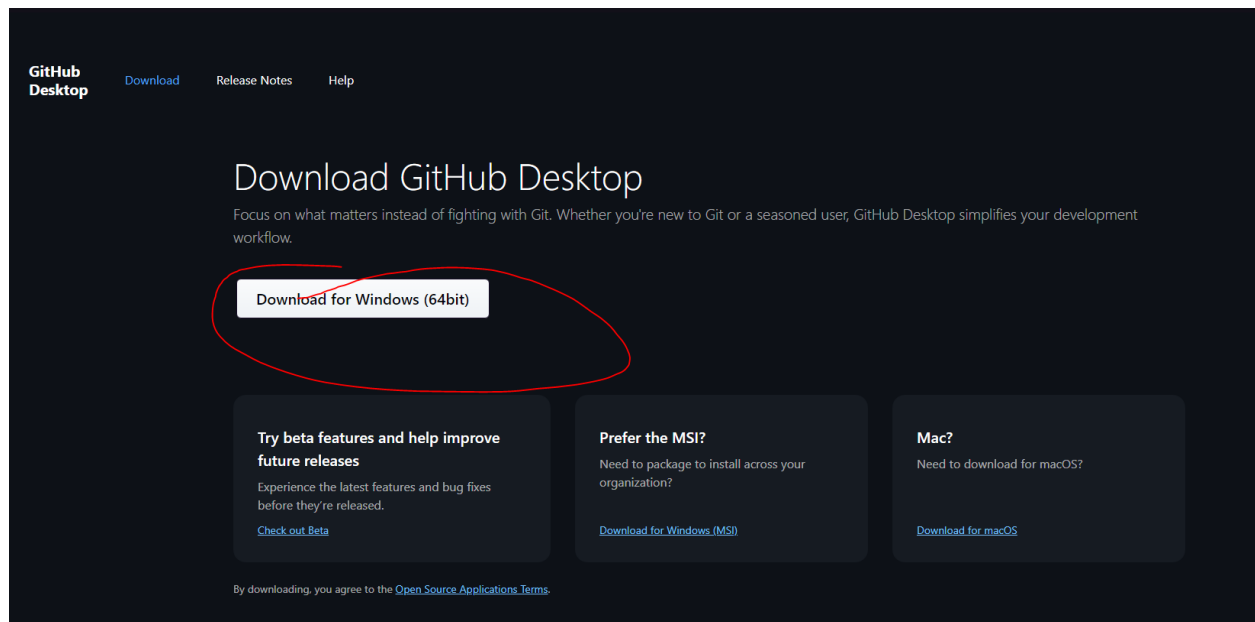


This is the path to our JavaFX folder. If we need to find the library of JavaFX we can refer to this location.

Using GitHub Desktop

The simplest way to use code from GitHub is through this program. You don't even have to open Visual Studio Code and run a command in the terminal. This makes it so that you do not need to do that.

Click the link to GitHub Desktop and click the Download button:



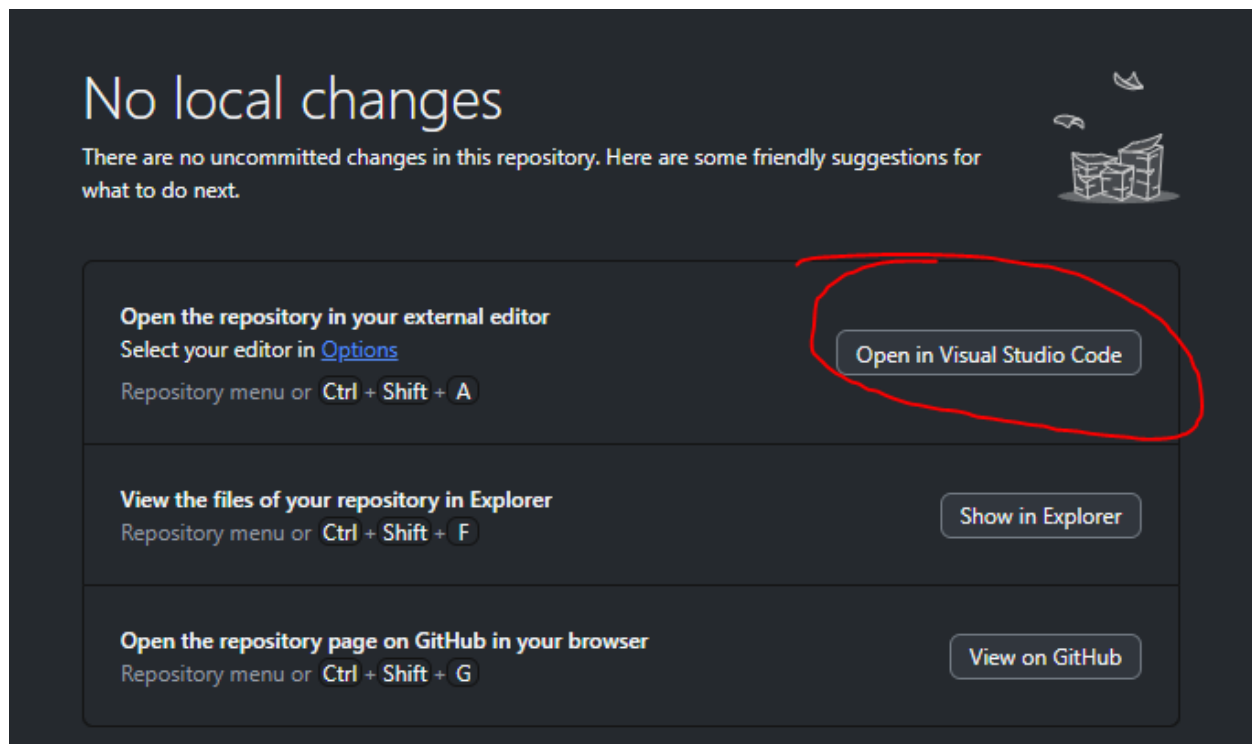
Run and set up this program.

You can log into your GitHub account if you need or want to.

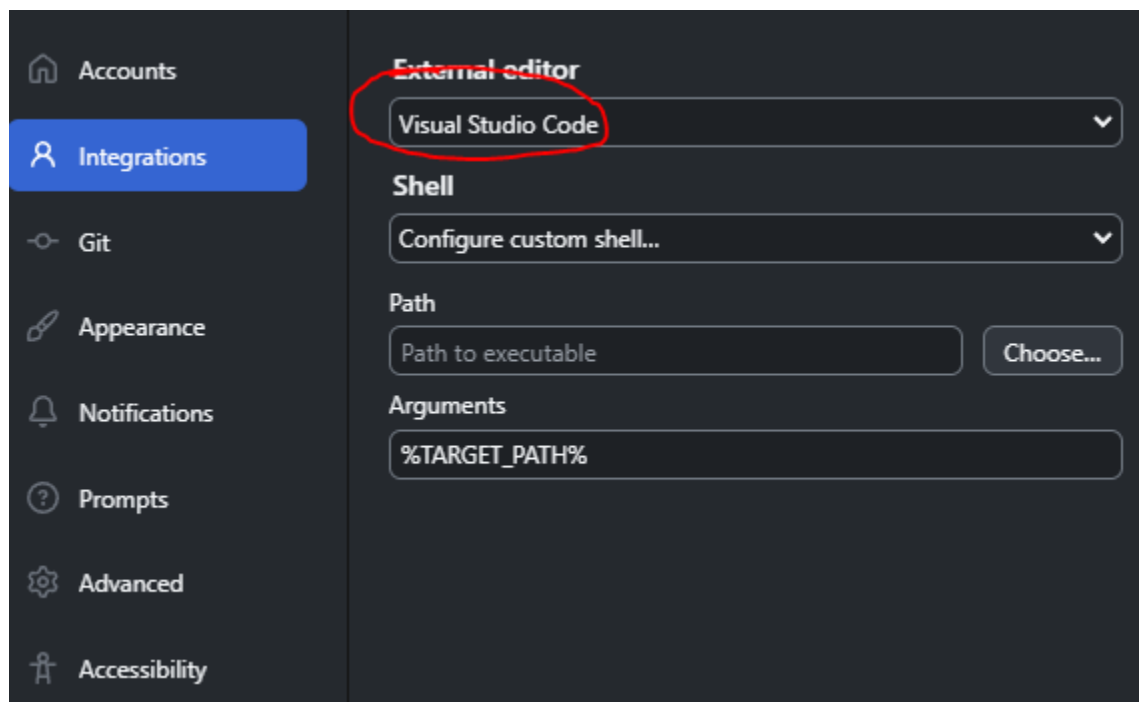
How to open Universal VSRG Charter code using GitHub Desktop:

From the GitHub repository, open the code drop down and select “Open with GitHub Desktop”

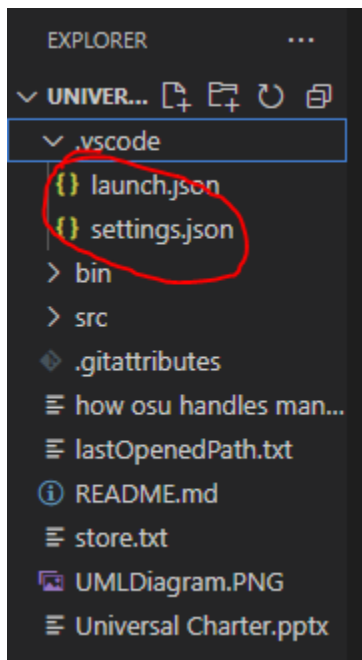
You will be prompted to clone the code. Press “Clone” and then you can open this with Visual Studio code:



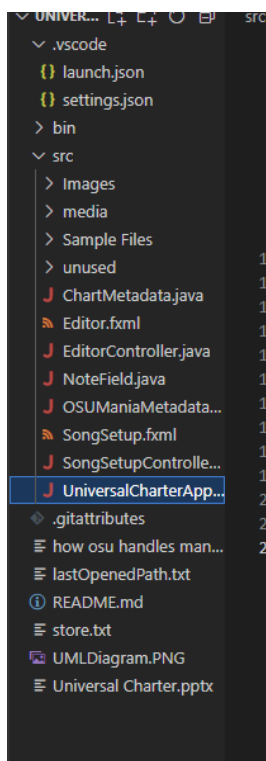
If it does not prompt you, you can press “Select your editor in Options” where Options is underlined and hit the first drop down and choose Visual Studio Code:



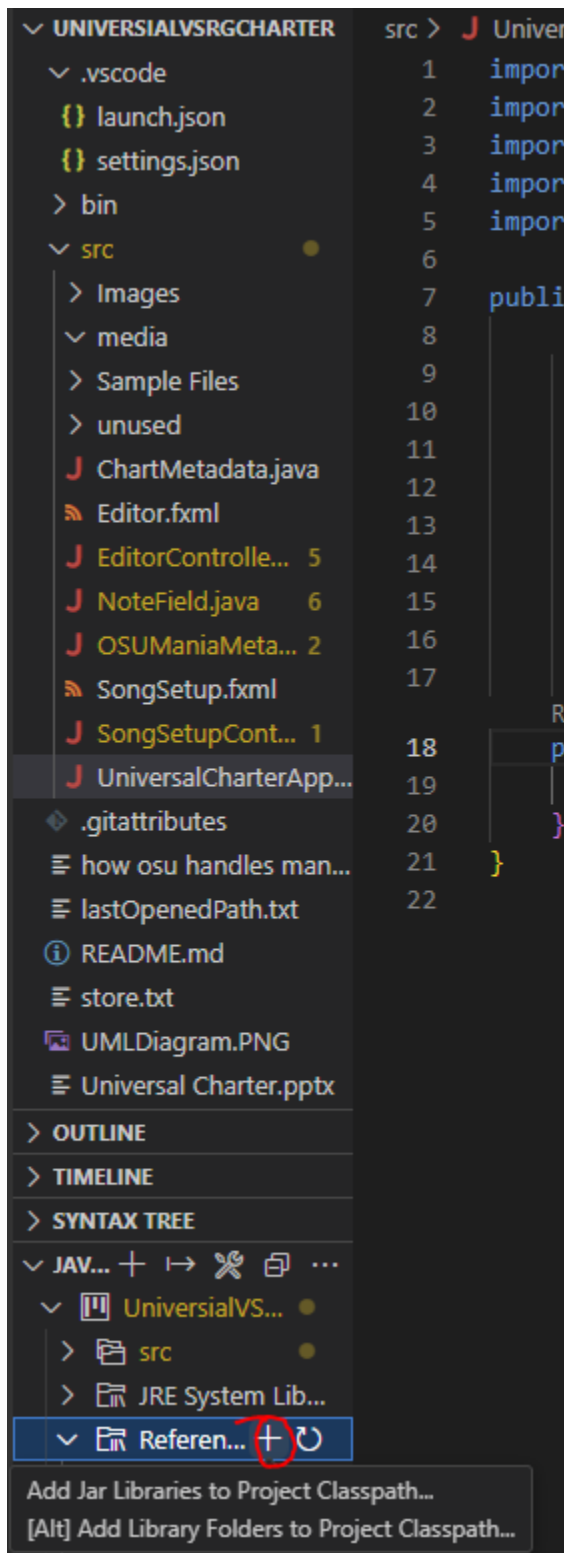
When you are able to open the code, navigate to .vscode in the built in file explorer and find the launch.json file and the settings.json file:



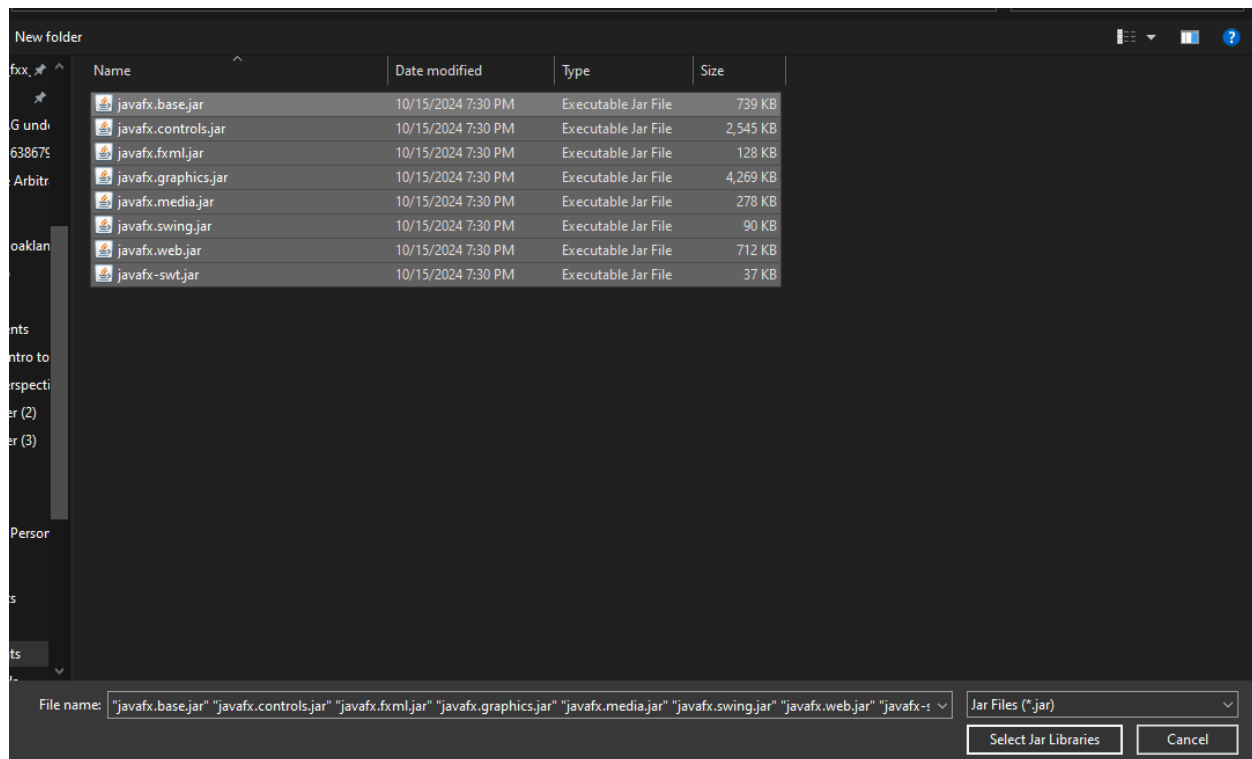
In your Visual Studio Code, open the explorer and open the “src” folder and select “UniversalCharterApp.java”



Then at the bottom of the window, go to “Java Project”, expand until you see “Referenced Libraries”, then click the “+” symbol:



Find the folder containing JavaFX and select all of the jar and then press “Select Jar Libraries”:

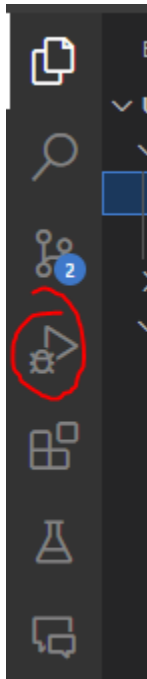


Now in .vscode, in the settings.json, your file should look like this:

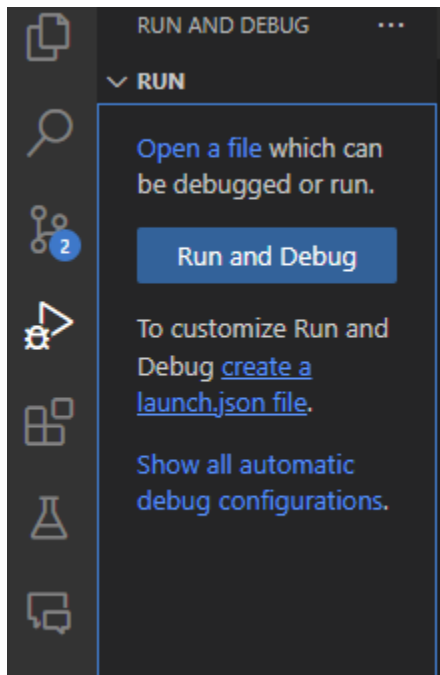


Remember that the path of the files will be different from mine to yours. If the last part contains the jar file, you are fine.

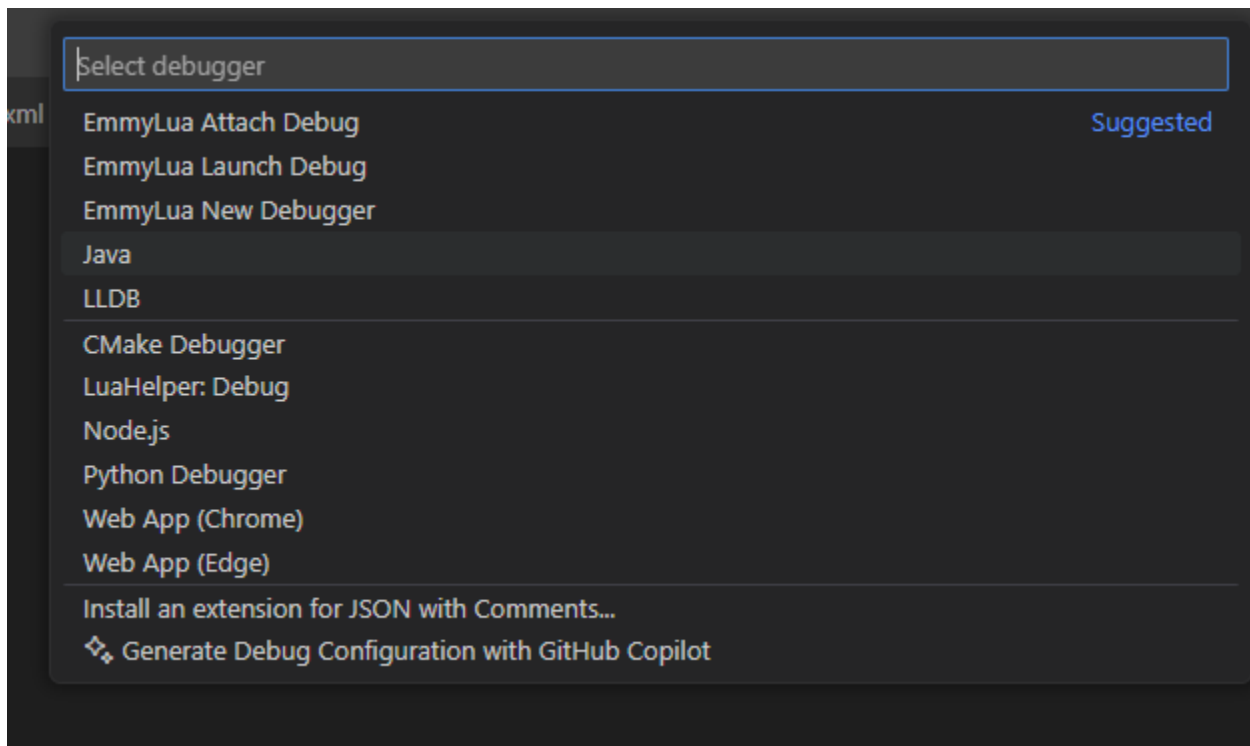
Now we click on this icon:



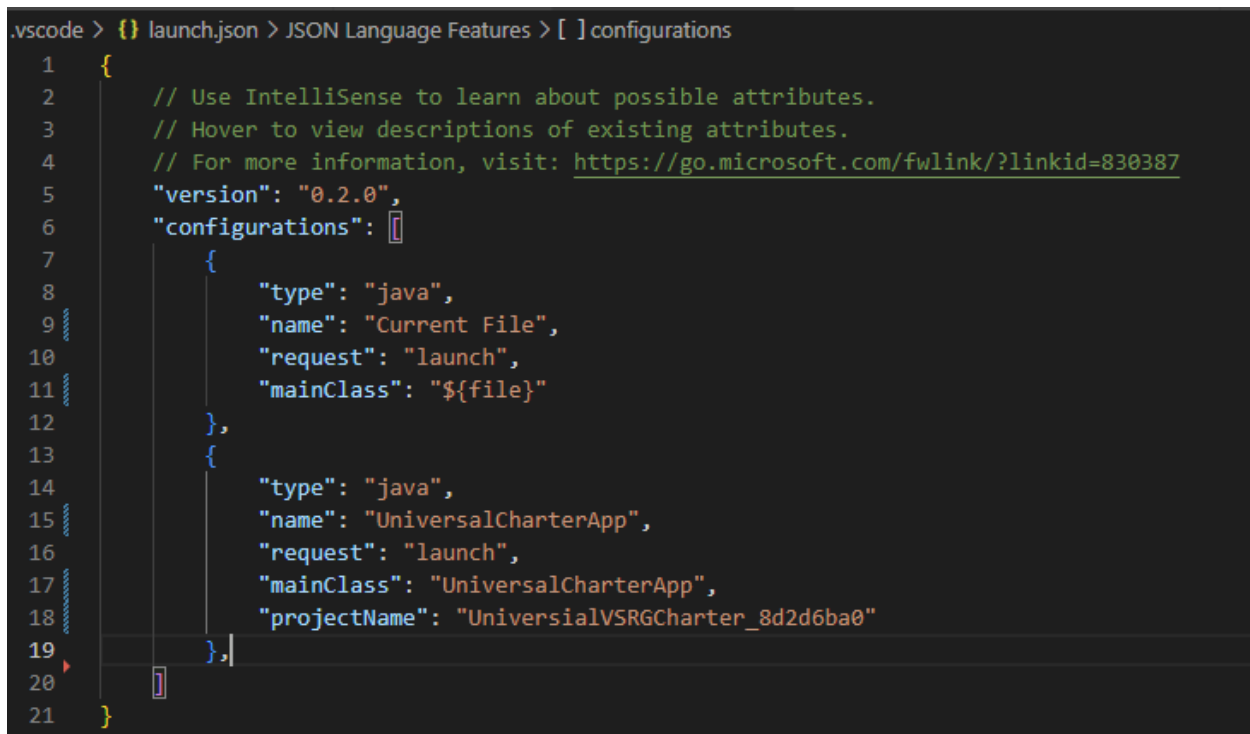
And now click on “create a launch.json file.”



If this pops up, select “Java”



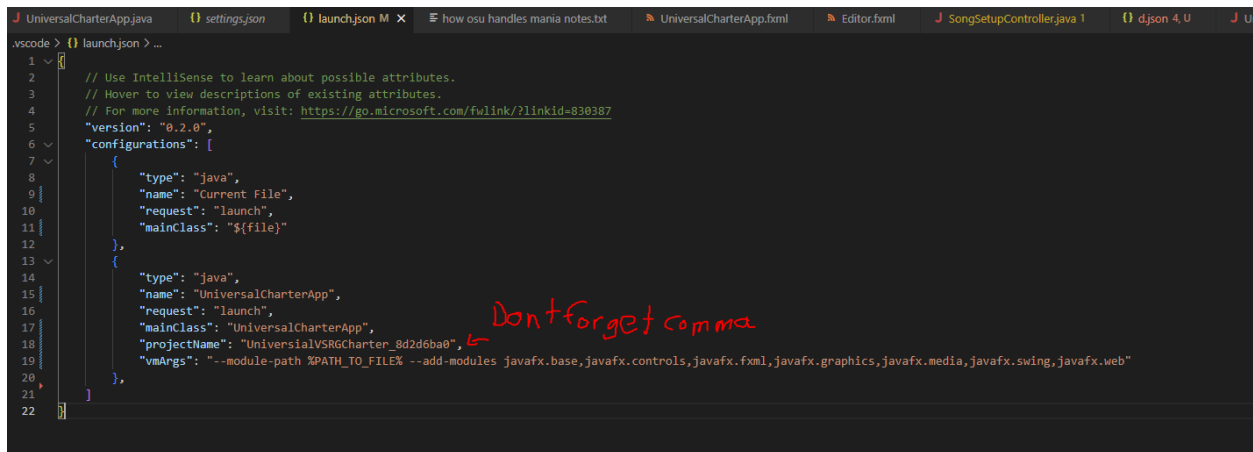
You should have this:



Next to “projectName: UniversalVSRGCharter_8d2d6ba0”, after the “, place a “,” next to it and press enter and paste this line:

```
"vmArgs": "--module-path %PATH_TO_FILE% --add-modules  
javafx.base,javafx.controls,javafx.fxml,javafx.graphics,javafx.media,javafx.swing,javafx.web"
```

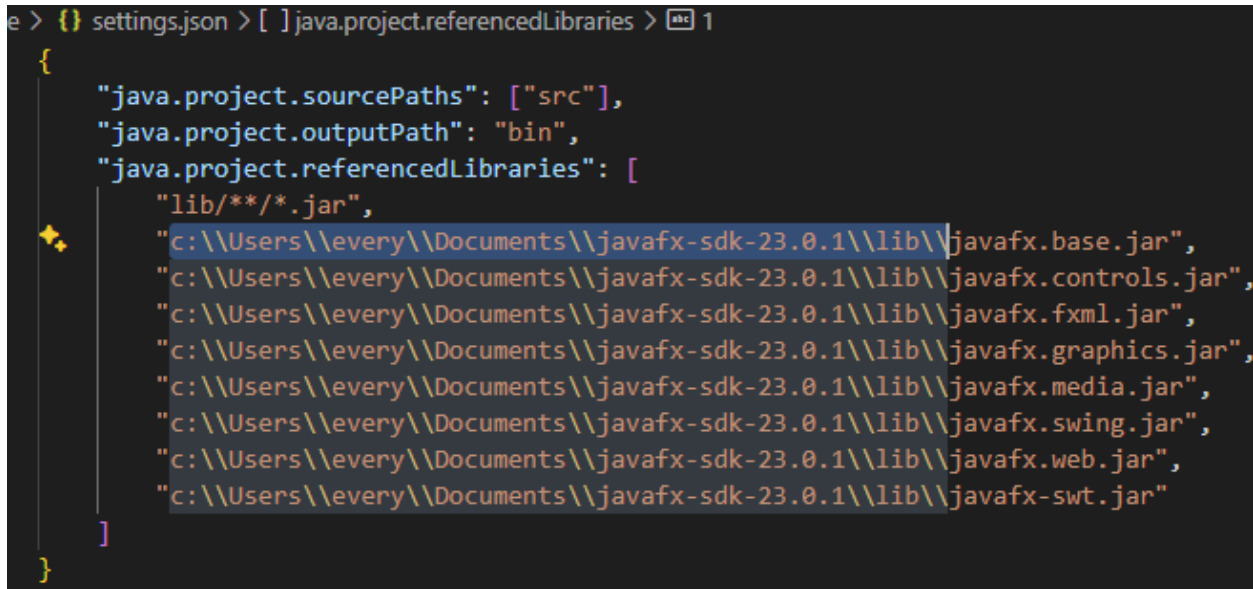
It will now look like:



```
.vscode > {} launch.json > ...  
1 // Use IntelliSense to learn about possible attributes.  
2 // Hover to view descriptions of existing attributes.  
3 // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387  
4  
5 "version": "0.2.0",  
6 "configurations": [  
7   {  
8     "type": "java",  
9     "name": "Current File",  
10    "request": "launch",  
11    "mainClass": "${file}"  
12  },  
13  {  
14    "type": "java",  
15    "name": "UniversalCharterApp",  
16    "request": "launch",  
17    "mainClass": "UniversalCharterApp",  
18    "projectName": "UniversalVSRGCharter_8d2d6ba0",  
19    "vmArgs": "--module-path %PATH_TO_FILE% --add-modules javafx.base,javafx.controls,javafx.fxml,javafx.graphics,javafx.media,javafx.swing,javafx.web"  
20  },  
21 ],  
22 }
```

Don't forget comma

Go back into launch.json and copy your path until the second to the last set of “\\”



```
e > {} settings.json > [ ] java.project.referencedLibraries > 1  
{  
  "java.project.sourcePaths": ["src"],  
  "java.project.outputPath": "bin",  
  "java.project.referencedLibraries": [  
    "lib/**/*.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx.base.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx.controls.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx.fxml.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx.graphics.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx.media.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx.swing.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx.web.jar",  
    "c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\javafx-swt.jar"  
  ]  
}
```

Replace “%PATH_TO_FILE%” with your path:

```
"type": "java",
"name": "UniversalCharterApp",
"request": "launch",
"mainClass": "UniversalCharterApp",
"projectName": "UniversalVSRGCharter_8d2d6ba0",
"vmArgs": "--module-path %PATH_TO_FILE% --add-modules javafx.base,javafx.controls,javafx.fxml,javafx.graphics,javafx.media,javafx.swing,javafx.web"
```

It will now look something like this:

```
"type": "java",
"name": "UniversalCharterApp",
"request": "launch",
"mainClass": "UniversalCharterApp",
"projectName": "UniversalVSRGCharter_8d2d6ba0",
"vmArgs": "--module-path c:\\Users\\every\\Documents\\javafx-sdk-23.0.1\\lib\\ --add-modules javafx.base,javafx.controls,javafx.fxml,javafx.graphics,javafx.media,javafx.swing,javafx.web"
```

We have to replace the “\\” with “/”, so now it will look like this:

```
},
{
  "type": "java",
  "name": "UniversalCharterApp",
  "request": "launch",
  "mainClass": "UniversalCharterApp",
  "projectName": "UniversalVSRGCharter_8d2d6ba0",
  "vmArgs": "--module-path c:/Users/every/Documents/javafx-sdk-23.0.1/lib --add-modules javafx.base,javafx.controls,javafx.fxml,javafx.graphics,javafx.media,javafx.swing,javafx.web"
},
]
```

If your path contains spaces, your entire path must be contain in single quotation marks (‘):

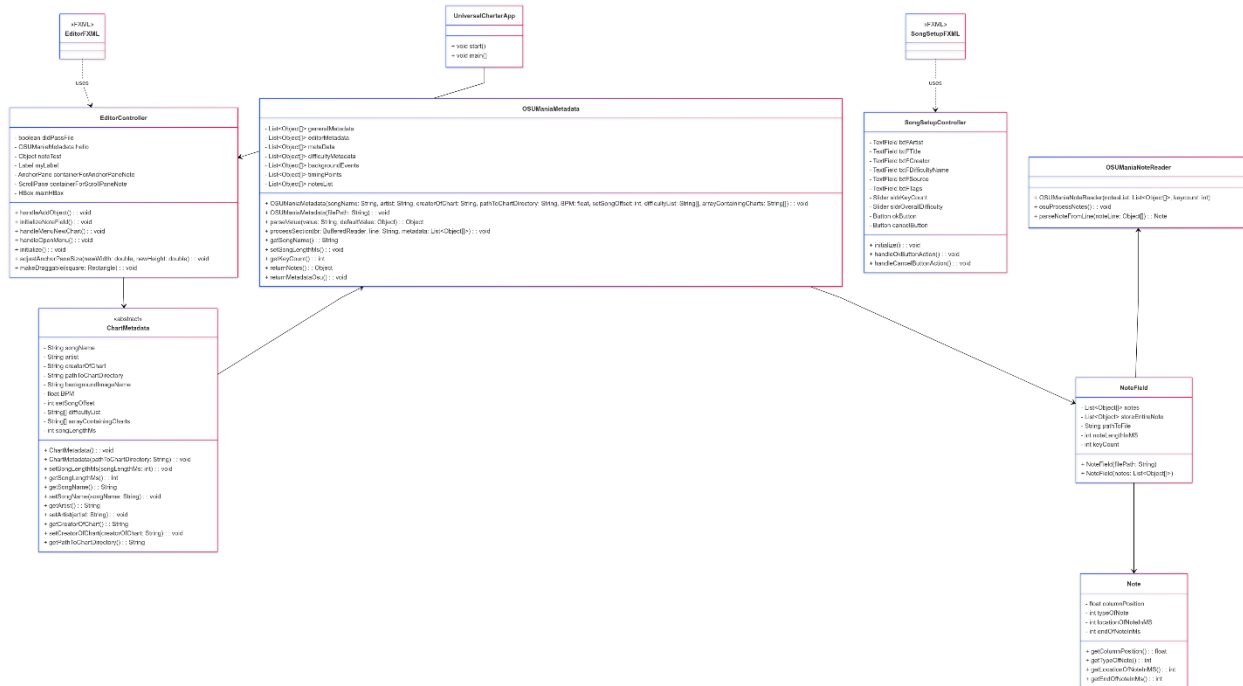
```
{
  "type": "java",
  "name": "UniversalCharterApp",
  "request": "launch",
  "mainClass": "UniversalCharterApp",
  "projectName": "UniversalVSRGCharter_8d2d6ba0",
  "vmArgs": "--module-path 'c:/Users/every/Documents/javafx-sdk-23.0.1/lib' --add-modules javafx.base,javafx.controls,javafx.fxml,javafx.graphics,javafx.media,javafx.swing,javafx.web"
},
]
```

With this, you should successfully be able to run the program. Try running UniversalCharterApp.java.

Chapter 2:

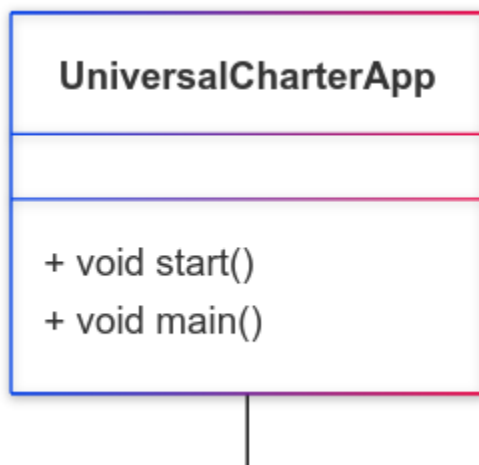
UML Diagram of Classes

The UML Diagram represents a program's classes through a visual diagram. It can tell us what classes are related to each other:



To make this easier, I will explain what class contains what and how they are being accessed:

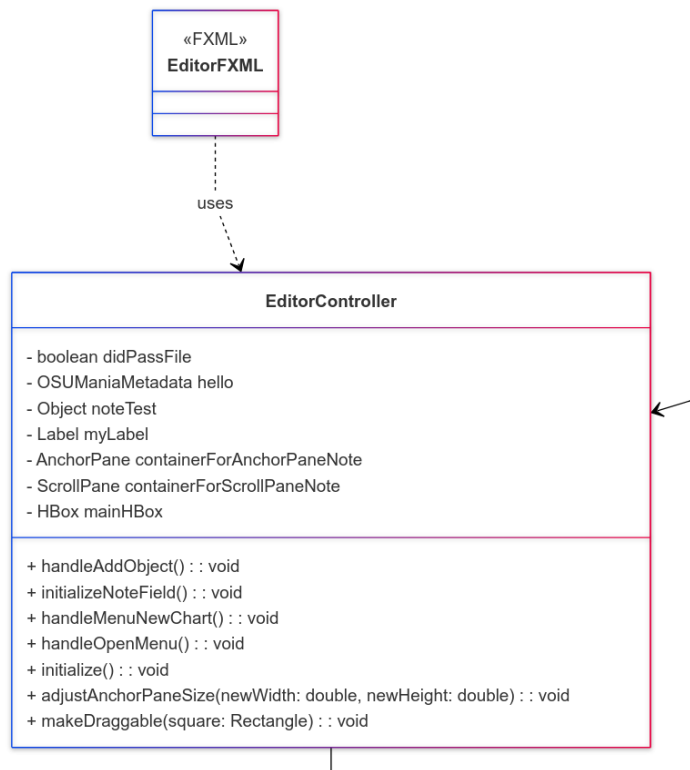
UniversalCharterApp class:



This class extends the abstract class Application and forces you to implement the abstract method:

- void Start():
 - This Sets up an object called primaryStage
 - This will call an FXML file “Editor.fxml” and uses it as a blueprint to output a window. This is stored as the variable “root”.
 - This will set the title to “Universal Charter App”.
 - primaryStage method sets the scene to root and will show this window
- void main()
 - Calls a method called launch() and passes in args. This will initialize Application class and proceed to run everything in the start method.

EditorController:

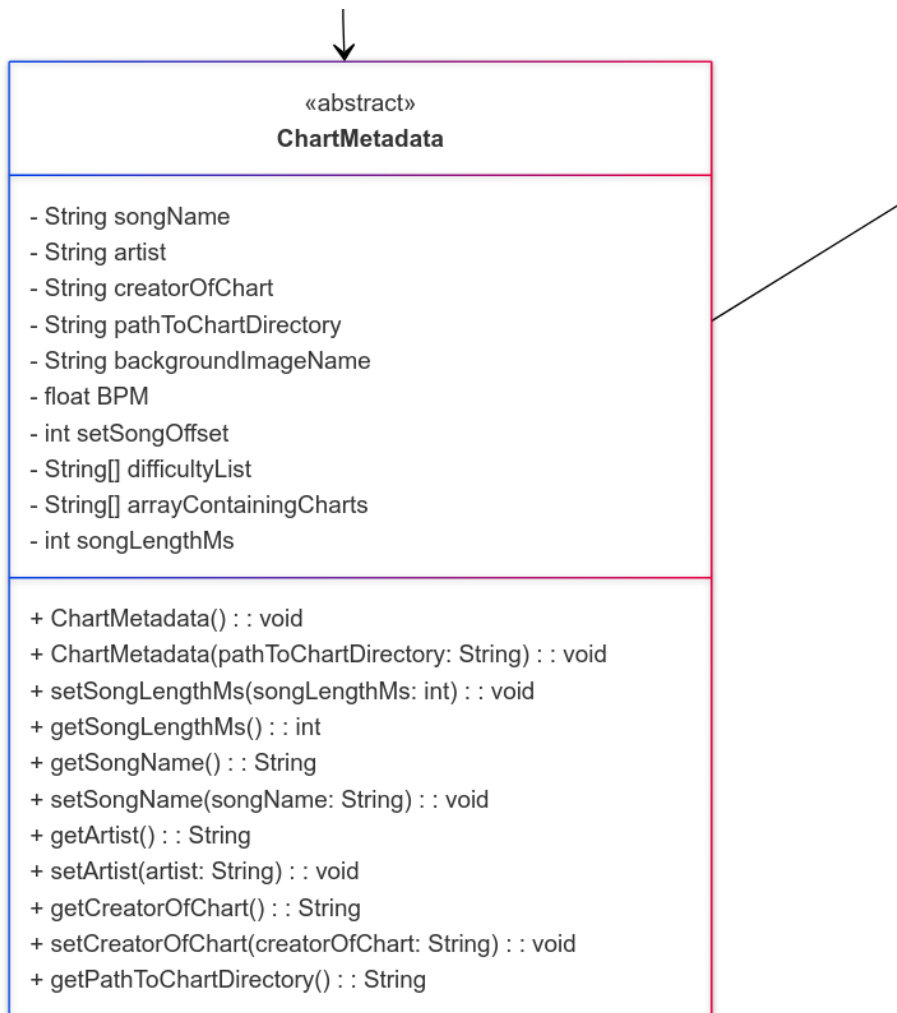


This class has a few variables and methods:

- Private Boolean didpassfile:
 - Determines if the user passes a file

- Private OSUManiaMetadata hello:
 - Creates an object called OSUManiaMetadata with the variable name hello
- Protected Object noteTest
 - This is an unfinished variable, but its purpose was to store an array of notes to then be parsed into displaying them onto the ScrollPane
- Private Label myLabel
 - Unused variable
- Private AnchorPane containerForAnchorPane
 - Defines the AnchorPane as a variable. This will be used to display our notes onto the pane
- Private ScrollPane containerForScrollPane
 - This just makes it so that the panes are scrollable as because the amount of object will be outside of the bounds.
- handleAddObject()
 - Used to place an object onto the pane. For debugging purposes
- initializeNoteField()
 - Used to place the AnchorPane and ScrollPane and place the notes onto them
- handleMenuNewChart()
 - This will execute when File>New is done. It currently opens a new window with parameters and textboxes to fill in. Currently does nothing.
- handleOpenMenu()
 - This will execute when File>Open is done. It will open a File Explorer for .osu files. Once we have found the file of that format we will parse them
- makeDraggable()
 - Will make an object on the pane draggable

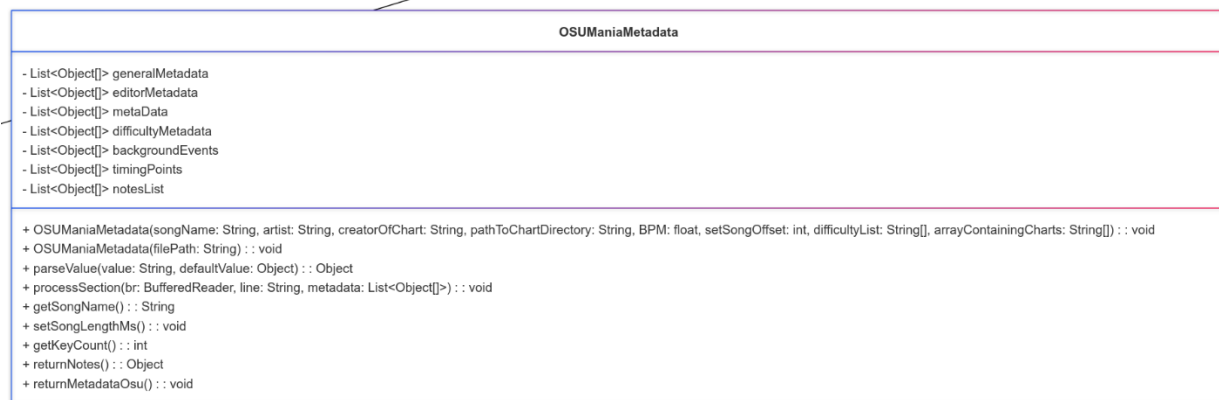
ChartMetaData (Abstract Class):



- Private String songName:
 - Determines the name of the song.
- Private String artist:
 - Contains the name of the artist of the song.
- Private String creatorOfChart:
 - Holds the name of the person that created the chart.
- Private String pathToChartDirectory:
 - Represents the file path to the directory containing the file.
- Private String backgroundImageName:
 - Stores the name of the background image used for the chart.
- Private float BPM:
 - Represents the beats per minute of the song.

- Private int setSongOffset:
 - Specifies the offset (time delay) for synchronizing the chart with the song in milliseconds.
- Private String[] difficultyList:
 - An array of strings representing the available difficulty levels.
- Private String[] arrayContainingCharts:
 - An array that stores references or filenames for different chart versions.
- Private int songLengthMs:
 - Stores the duration of the song in milliseconds.
- Constructor ChartMetadata(pathToChartDirectory: String):
 - Initializes the ChartMetadata object with the specified path to the chart directory.
- void setSongLengthMs(songLengthMs: int):
 - Sets the length of the song in milliseconds.
- int getSongLengthMs():
 - Retrieves the length of the song in milliseconds.
- void setSongName(songName: String):
 - Sets the name of the song.
- String getSongName():
 - Gets the name of the song.
- void setArtist(artist: String):
 - Sets the artist's name.
- String getArtist():
 - Retrieves the artist's name.
- void setCreatorOfChart(creatorOfChart: String):
 - Sets the name of the chart creator.
- String getCreatorOfChart():
 - Retrieves the name of the chart creator.
- String getPathToChartDirectory():
 - Returns the path to the chart directory.

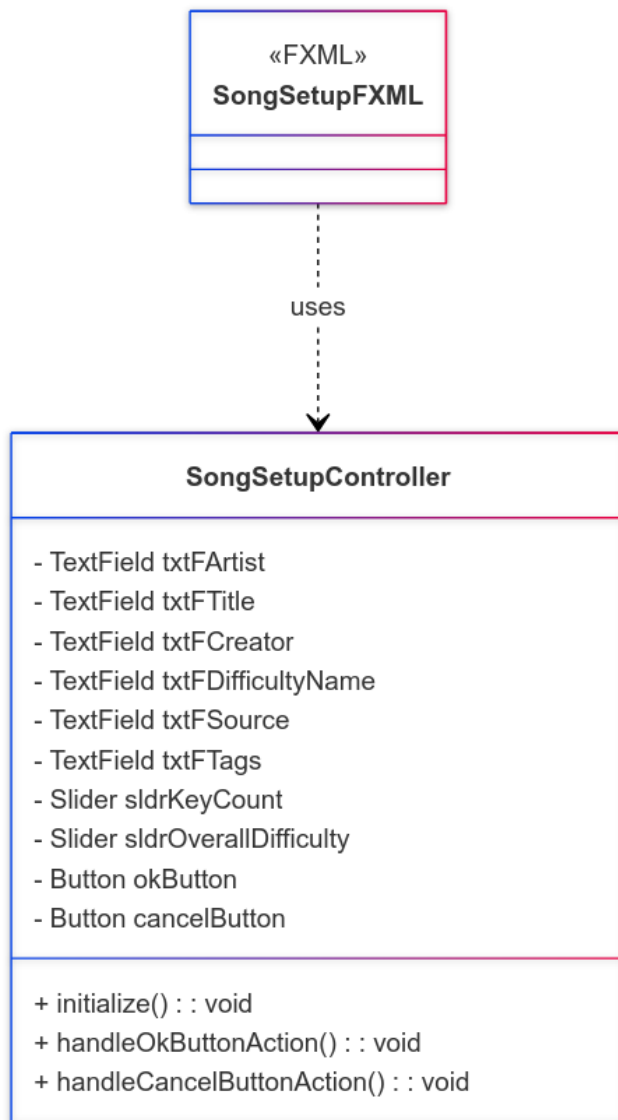
OSUManiaMetadata:



This class extends ChartMetadata. It will inherit all variables in ChartMetadata.

- Private List<Object[]> generalMetadata:
 - Stores general section of the .osu file
- Private List<Object[]> editorMetadata:
 - Stores editor section of the .osu file
- Private List<Object[]> metaData:
 - Stores metadata section of the .osu file
- Private List<Object[]> difficultyMetadata:
 - Stores difficulty section of the .osu file
- Private List<Object[]> backgroundEvents:
 - Stores Events section of the .osu file
- Private List<Object[]> timingPoints:
 - Stores timing points section of the .osu file
- Private List<Object[]> notesList:
 - Stores HitNotes section of the .osu file
- Constructor OSUManiaMetadata(songName: String, artist: String, creatorOfChart: String, pathToChartDirectory: String, BPM: float, setSongOffset: int, difficultyList: String[], arrayContainingCharts: String[]):
 - This is used when you create a new chart.
- Constructor OSUManiaMetadata(filePath: String):
 - This is used when you pass in a file
- parseValue(value: String, defaultValue: Object): Object:
 - Parses a value and returns it as an object, or a default value if the parsing fails.
- processSection(br: BufferedReader, line: String, metadata: List<Object[]>): void:
 - Processes a section of the .osu file and populates the corresponding metadata list.
- getSongName(): String:
 - Retrieves the name of the song associated with the chart.
- setSongLengthMs(): void:
 - Sets the song length in milliseconds.
- getKeyCount(): int:
 - Retrieves the number of keys used in the chart.
- returnNotes(): Object:
 - Returns the HitNotes section. Used for passing the notes to be processed.
- returnMetadataOsu(): void:
 - Returns the metadata section. For finding information like the song name.

SongSetupController:

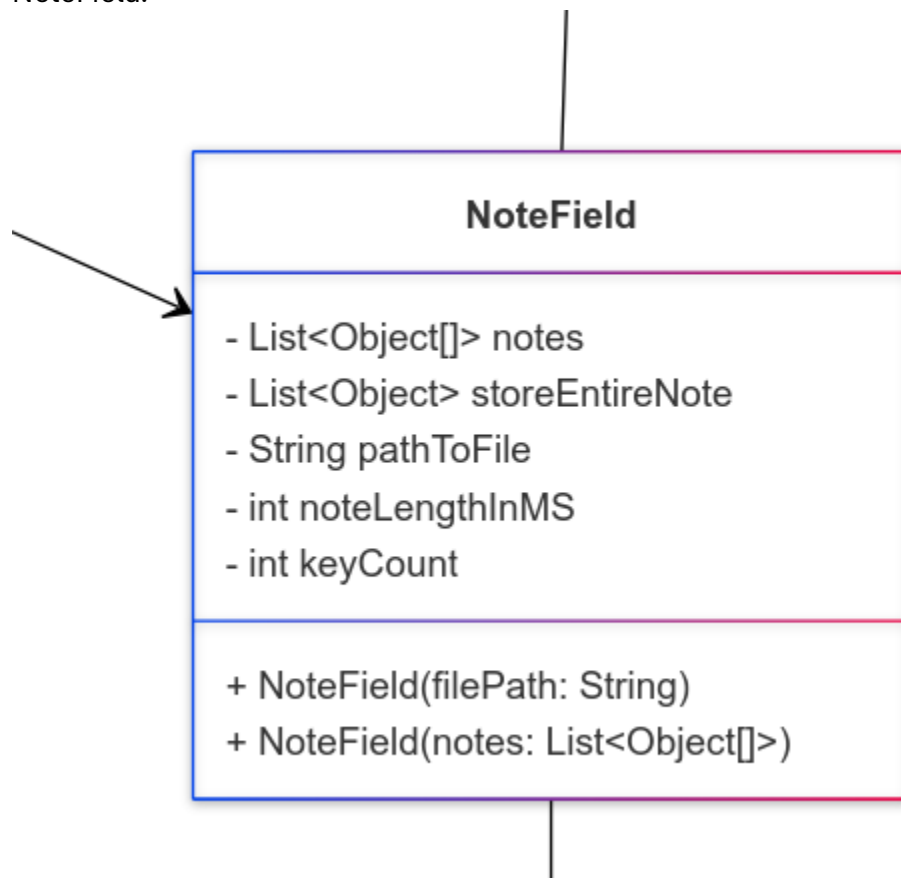


This opens after you do File>New. This create a new chart to be used in OSU.

- Private TextField txtFArtist
 - This stores the value of textfield of Artist
- Private TextField txtFTitle
 - This stores the value of textfield of Title
- Private TextField txtFCreator
 - This stores the value of textfield of Creator of the chart
- Private TextField txtFDifficultyName
 - This stores the value of textfield of Difficulty name
- Private TextField txtFSource
 - This stores the value of textfield of source of the song

- Private TextField txtFTags
 - This stores the value of textfield of tags of the chart
- Private Slider sldrKeyCount
 - This stores the value of the slider of keycount.
- Private Slider sldrOverallDifficulty
 - This stores the value of the slider for Overall Difficulty
- Private Button okButton
 - For referencing the ok button
- Private Button cancelButton
 - For referencing the cancel button
- Initialize()
 - Not coded
- handleOkButton()
 - Suppose to handle creating the chart. Does nothing as of now
- handleCancelButton()
 - Only clears the fields. Plan on closing the window.

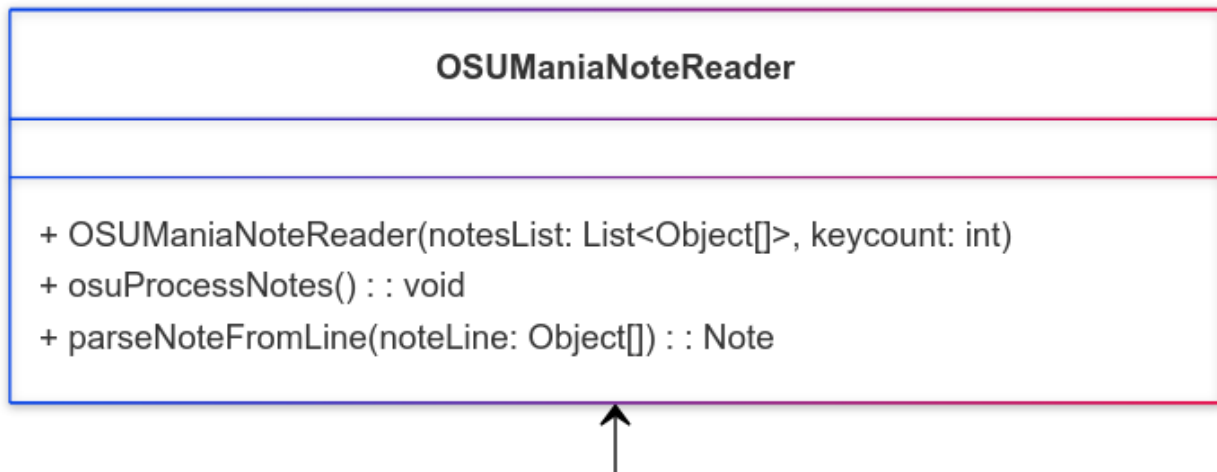
NoteField:



- protected List<Object[]> notes
 - Holds all of the notes and note data per line

- `protected List<Object> storeEntireNote`
 - Suppose to hold 1 note and its data.
- `private String pathToFile`
 - Stores the path to file
- `private int noteLengthInMS`
 - For long notes. Stores the length of time the note lasts in milliseconds
- `protected int keyCount`
 - Holds the key count.
- Constructor `NoteField(String filePath)`
 - This constructor is used when you pass a file. Sets the filepath
- Constructor `NoteField(List<Object[]> notes)`
 - This is used when you pass in a array of notes.

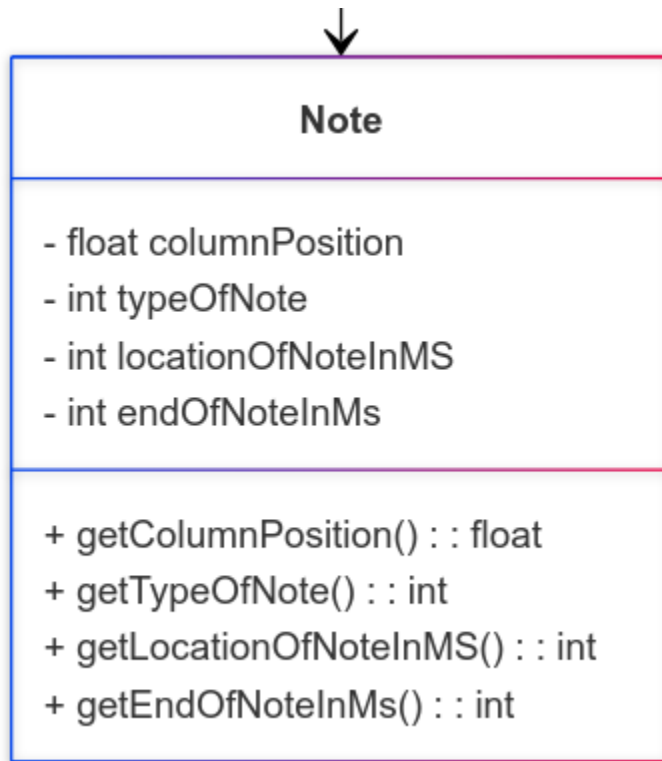
OSUManiaNoteReader:



This class extends NoteField.

- Constructor `OSUManiaNoteReader(List<Object[]> notelist: int keycount)`
 - For passing in the note data and the keycount
- `osuProcessNotes()`
 - This processed the passed in HitNotes section of the .osu file. This will separate break down the lines so that we can actually use them
- `parseNoteFromLine(Object[] noteLine)`
 - After processing the notes, we will set our results to an object called Note. A note will hold all data from 1 line.

Note:



- float columnPosition
 - Holding the columnPosition of the note
- int typeOfNote
 - holds what type the note is
- int locationOfNoteInMS
 - location of the note in milliseconds
- int endOfNoteInMs
 - This is used only for long notes. Signify the ending time of the note.
- getColumnPostition()
 - returns the variable column postiton
- getTypeOfNote()
 - returns the variable typeOfNote
- getLocationOfNoteInMs()
 - returns the variable LocationOfNoteInMS
- getEndOfNoteInMs()
 - returns the variable EndOfNoteInMs