

¿Como configurar mi ambiente de javafx en Visual Studio Code (vscode) con MacOS?

Paso 1:

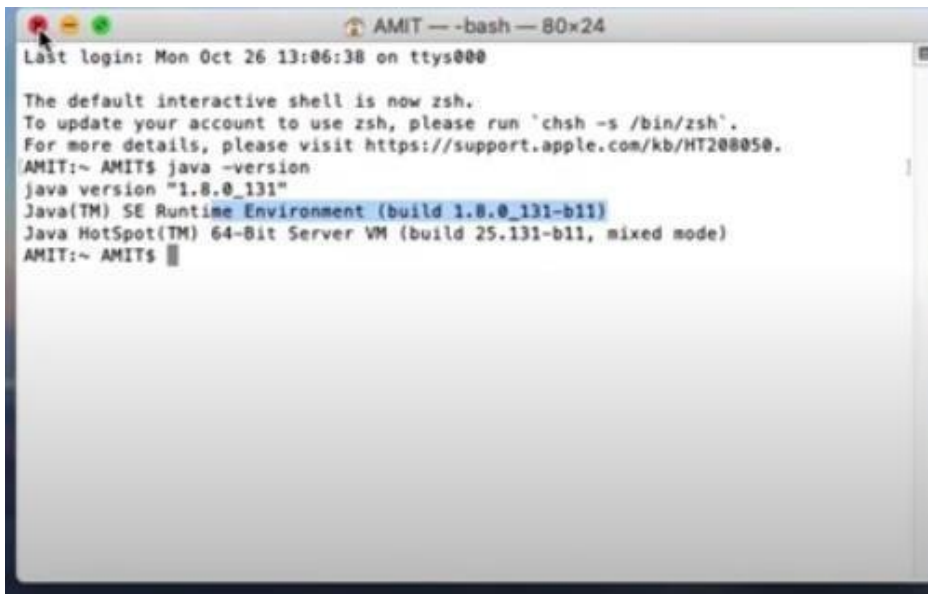
Actualizar o descargar la última versión de VS code

Paso 2:

Instalar Oracle u open Jdk 11 para correr proyectos java en vscode con sistema operativo MacOS, link de descarga: <https://adoptopenjdk.net/> seguir los pasos de instalación indicados.

Paso 3:

Verificar la instalación exitosa de Java, ingresando desde la terminal 'java -version' deberá verse algo así:



```
AMIT ~ -bash — 80x24
Last login: Mon Oct 26 13:06:38 on ttys000

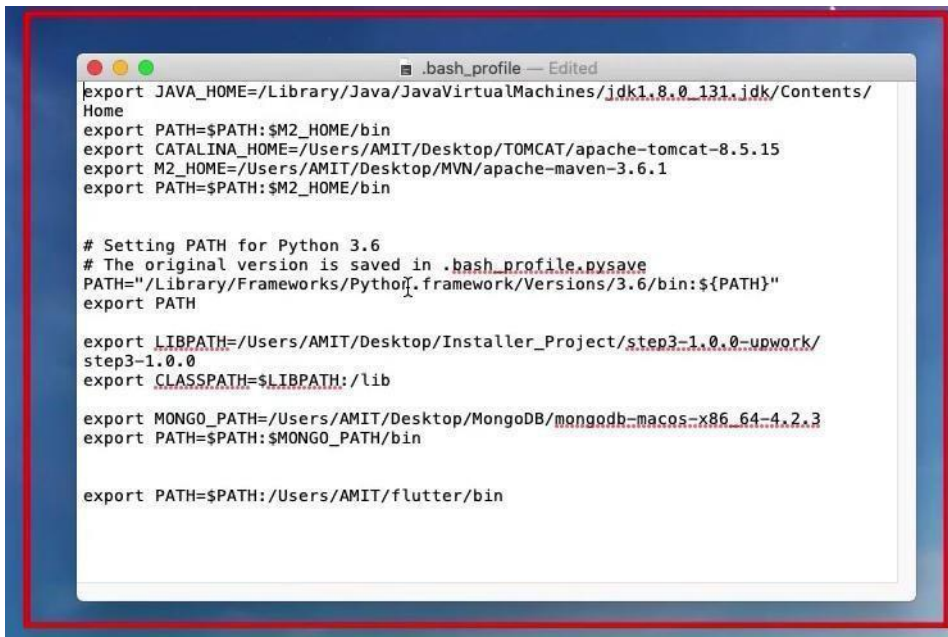
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
AMIT:~ AMIT$ java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
AMIT:~ AMIT$
```

Paso 4:

Nuevamente desde la terminal, escribir el siguiente comando para abrir el documento bash_profile:

```
open ~/.bash_profile
```

Al presionar enter luego de escribir el comando debe abrirse una nueva ventana como la siguiente:

A screenshot of a text editor window titled ".bash_profile — Edited". The window contains a series of export statements for environment variables. The first section sets JAVA_HOME to "/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home", then sets PATH to include \$M2_HOME/bin. The second section sets CATALINA_HOME to "/Users/AMIT/Desktop/TOMCAT/apache-tomcat-8.5.15" and M2_HOME to "/Users/AMIT/Desktop/MVN/apache-maven-3.6.1", then updates PATH to include \$M2_HOME/bin. The third section is a comment about setting PATH for Python 3.6, followed by a comment about saving the original version in .bash_profile.py, and then sets PATH to include "/Library/Frameworks/Python.framework/Versions/3.6/bin:\${PATH}". The fourth section sets LIBPATH to "/Users/AMIT/Desktop/Installer_Project/step3-1.0.0-upwork/step3-1.0.0" and CLASSPATH to \$LIBPATH:/lib. The fifth section sets MONGO_PATH to "/Users/AMIT/Desktop/MongoDB/mongodb-macos-x86_64-4.2.3" and updates PATH to include \$MONGO_PATH/bin. The final line updates PATH to include "/Users/AMIT/flutter/bin".

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home
export PATH=$PATH:$M2_HOME/bin
export CATALINA_HOME=/Users/AMIT/Desktop/TOMCAT/apache-tomcat-8.5.15
export M2_HOME=/Users/AMIT/Desktop/MVN/apache-maven-3.6.1
export PATH=$PATH:$M2_HOME/bin

# Setting PATH for Python 3.6
# The original version is saved in .bash_profile.py
PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"
export PATH

export LIBPATH=/Users/AMIT/Desktop/Installer_Project/step3-1.0.0-upwork/step3-1.0.0
export CLASSPATH=$LIBPATH:/lib

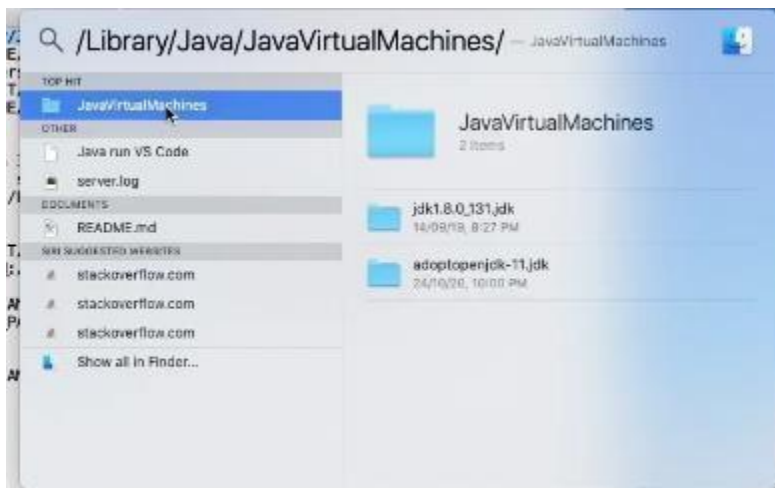
export MONGO_PATH=/Users/AMIT/Desktop/MongoDB/mongodb-macos-x86_64-4.2.3
export PATH=$PATH:$MONGO_PATH/bin

export PATH=$PATH:/Users/AMIT/flutter/bin
```

Como se puede observar este archivo exporta una versión de Java diferente a la instalada, por lo que es necesario realizar la corrección.

Paso 5:

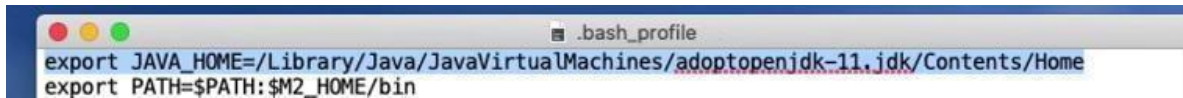
Desde el buscador de archivos de mac ingresaremos la siguiente ruta:



Y abrir la carpeta JavaVirtualMachines. En esta carpeta se podrá observar la versión anterior de java junto con la carpeta instalada en el paso 2. Acceder a: adoptopenjdk-11.jdk/Contents/Home y obtener la ruta de esta carpeta Home (<https://youtu.be/j4P4ENQxtRs?t=268>)

Paso 6:

Copiar la ruta del paso anterior y pegar en el archivo `bash_profile` ya abierto. Reemplazar la primera fila por la ruta copiada y guardar.



```
.bash_profile
export JAVA_HOME=/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home
export PATH=$PATH:$JAVA_HOME/bin
```

Paso 7:

Una vez actualizada la ruta, abrir nuevamente terminal y verificar la versión de java 'java -version'

Paso 8:

Ahora se podrá ver que la versión es la que necesita.

Paso 9:

Instalar en vscode la extensión Java Extension Pack

(<https://marketplace.visualstudio.com/items?vscjava.vscode-java-pack>).

Java Extension Pack es una colección de extensiones populares que pueden ayudar a escribir, probar y depurar aplicaciones Java en Visual Studio Code.

Paso 10:

Reiniciar VS code

Paso 11:

Para comenzar con el proyecto, acceder a paleta de herramientas de vscode (presionar CTRL + SHIFT + P) e ingresar 'Java: Create java project...' en modo de trabajo No build tools

Paso 12:

Seleccionar la ubicación en donde se quiere guardar el proyecto y asignar un nombre ejemplos: 'Hola_Mundo', 'Project1'

Paso 13:

Instalar las siguientes extensiones de javafx para VS code: **javafx Support**

(<https://marketplace.visualstudio.com/items?itemName=shrey150.javafx-support>)

y

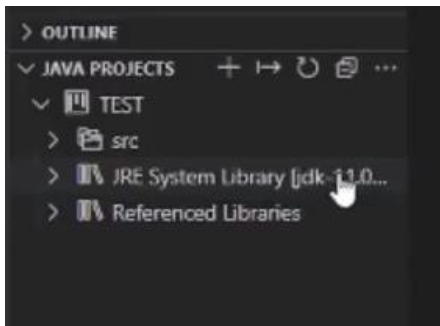
FXML Viewer (<https://marketplace.visualstudio.com/items?itemName=lzw-723.fxml-viewer>) estas herramientas ayudarán a crear interfaces gráficas de usuario.

Paso 14:

Instalar **javafx SDK** <https://gluonhq.com/products/javafx/> su última versión para javafx Mac OS X SDK. Descomprimir carpeta descargada y guardarla donde sea de preferencia. Se necesita configurar este sdk en vs code para correr los programas con javafx o cualquier otra aplicación javafx.

Paso 15:

Volver al proyecto Java creado, para configurar javafx abrir el archivo App.java y expandir todos los árboles de archivos que se encuentran del lado izquierdo en vscode.



Al abrir cualquier archivo Java, vscode, gracias a las extensiones instaladas, detectará esto y habilitará una nueva opción llamada 'JAVA PROJECTS' que contendrá una carpeta con el nombre de nuestro proyecto, desplegamos este menú y accedemos a la subcarpeta Referenced Libraries. A esa carpeta arrastraremos o cargaremos los ítems del sdk que descargamos en el paso anterior (las que se encuentran en el directorio 'lib')

Una vez cargados todos estos archivos .jar al proyecto, aparecerán nuevos archivos a nuestro proyecto que harán parte de su configuración. Un ejemplo es el archivo settings.json el cual contiene los archivos .jar de javafx a los que haremos referencia.

Paso 16:

Para trabajar, es necesario que el editor de código vscode cuente con la sintaxis de aplicaciones javafx, para esto ir a la opción 'Run' en el menú de vscode y seleccionar la opción Add Configuration...

Dentro de este archivo, añadir un nuevo argumento de configuración llamado "vmArgs" y tendrá como valor la ruta en el equipo a la carpeta lib de java-fx-sdk descargada en el paso 14. Al final, el archivo debe verse parecido al siguiente, tenga en cuenta que solo debe añadir la línea resaltada:

```

{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "java",
      "name": "Launch Current File",
      "request": "launch",
      "mainClass": "${file}"
    },
    {
      "type": "java",
      "name": "Launch App",
      "request": "launch",
      "mainClass": "App",
      "vmArgs": "--module-path *RUTA EN DONDE SE ENCUENTRAN LIBS JAVAFX "C:/.."* --add-modules
      javafx.controls,javafx.fxml",
      "projectName": "CP_16_4fdb85f8"
    }
  ]
}

```

Para este paso tener en cuenta:

- Cambiar la configuración de permisos de software de 3ros (<https://support.apple.com/es-co/guide/mac-help/mh43185/mac>)
- La ruta en el parámetro VmArgs del archivo de configuración NO pueden tener espacios, ni signos especiales, Ñ, tildes o Comilla . Cualquiera de estos símbolos hace que java sea incapaz de procesar la ruta.
- El parámetro VmArgs con la ruta de las librerías javafx debe ser añadido en aquellas configuraciones que lo requieran . Ejemplo: Si necesito hacer el llamado de alguna librería javafx desde mi App.java debo añadir el VmArgs con la ruta de las librerías en la configuración que tenga por *"mainClass": "App"*
- En caso de reutilizar el launch.json de otro proyecto, verificar que el projectName de las configuraciones correspondan con el nombre del proyecto en el que se están pegando.

Paso 17:

Pegar en el archivo App.js el siguiente ejemplo para validar que todo funcione y presionar

en el botón Run Java, esto abrirá una venta de javafx de la prueba y listo.

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class App extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World!");
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(btn);
        primaryStage.setScene(new Scene(root, 300, 250));
        primaryStage.show();
    }
}
```

Conceptos claves

¿Qué es javafx?

Pensar en javafx como un grupo de paquetes que permite crear grandiosas aplicaciones de escritorio e Internet.

javafx te permite crear aplicaciones GUI, pero con menos programación y con más efectos visuales a tu disposición.

¿Qué es FXML?

FXML es un formato de archivo que utiliza javafx para crear el diseño de las pantallas.

¿Qué es un SDK?

Un kit de desarrollo de software (SDK) es un conjunto de herramientas que ofrece generalmente el fabricante de una plataforma de hardware, un sistema operativo (SO) o un lenguaje de programación. Los **SDK** permiten que los desarrolladores **de** software creen aplicaciones para esa plataforma, ese sistema o ese lenguaje **de** programación específicos.

¿Qué es un JDK?

Java Development Kit es un software que provee herramientas de desarrollo para la creación de programas en Java.

¿Qué es un archivo .jar?

Un archivo JAR es un tipo de archivo que permite ejecutar aplicaciones y herramientas escritas en el lenguaje Java

¿Qué es una librería?

Una librería Java se puede entender como un conjunto de clases que facilitan operaciones y tareas ofreciendo al programador funcionalidad ya implementada y lista para ser usada través de una Interfaz de Programación de Aplicaciones, comúnmente abreviada como API (por el anglicismo Application Programming Interface)

Ejemplos básicos

Contenedor

```
Scene scene = new Scene(root, 500, 300);
primaryStage.setScene(scene);
primaryStage.show();
```

Botón:

```
Button boton = new
Button("JavaFX");
boton.setDefaultButton(true);
    // Tamaño del boton
boton.setPrefSize(100, 50);
    // Tamaño del boton
    // Posición de boton
boton.setLayoutX(105);
boton.setLayoutY(110);

    // Se agrega el boton
root.getChildren().add(boton);
```

```
boton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        //acción que se debe ejecutar al hacer click en boton
    }
})
```

Label:

```
//Declaración de nuevo texto //label
Text text = new Text();
text.setText("Welcome to Portal"); //Setear contenido del texto text.setFont(Font.font("Calibri",
FontWeight.BOLD, FontPosture.REGULAR, 20)); //Definir estilos del texto
```


Formulario:

```
//Formulario, label + textfield o input  
Label name = new Label("UserName");  
Label pass = new Label("Password");  
TextField tf1 = new TextField();  
PasswordField tf2 = new PasswordField();
```

Tabla o Grid:

```
GridPane root = new GridPane(); //Declaramos el panel con  
cuadrícula root.addRow(1, text); //Añadr fila a cuadrícula  
root.addRow(id, elm) root.setVgap(10); //Altura de los espacios  
verticales entre filas. root.addRow(2, name, tf1);  
root.setVgap(10);
```

Menu:

```
MenuBar menubar = new MenuBar();  
Menu FileMenu = new Menu("File");  
MenuItem filemenu1 = new MenuItem("Clear Messages");  
MenuItem filemenu2 = new MenuItem("Clear Forms");  
FileMenu.getItems().addAll(filemenu1, filemenu2);  
menubar.getMenus().addAll(FileMenu);
```

DOCUMENTACIÓN javafx CLASSES:

<https://docs.oracle.com/javase/8/javafx/api/index.html>