

Introducción

JavaFX es una biblioteca de Java utilizado para construir aplicaciones dinámicas de Internet. Las aplicaciones escritas usando esta biblioteca puede funcionar constantemente a través de múltiples plataformas. Las aplicaciones desarrolladas utilizando JavaFX puede funcionar en diversos dispositivos, como ordenadores de sobremesa, teléfonos móviles, televisores, tabletas, etc ..

Para desarrollar GUI Applications usando el lenguaje de programación Java, los programadores se basan en las bibliotecas, tales como Advanced Windowing Tool kit y Swings . Después de la llegada de JavaFX, estos programadores de Java pueden ahora desarrollar aplicaciones GUI eficazmente con contenido rico.

En este tutorial, vamos a discutir todos los elementos necesarios de JavaFX que se pueden utilizar para desarrollar aplicaciones dinámicas de Internet eficaces.

Rich Internet Applications son aquellas aplicaciones web que proporcionan características y experiencia como la de las aplicaciones de escritorio similar. Ofrecen una mejor experiencia visual en comparación con las aplicaciones web normales a los usuarios. Estas aplicaciones se entregan como los complementos del navegador o como una máquina virtual y se utilizan para transformar aplicaciones estáticas tradicionales en más reforzada, líquido, animación y aplicaciones atractivas.

A diferencia de las aplicaciones de escritorio tradicionales, RIA de no requerir tener ningún software adicional para funcionar. Como alternativa, se debería instalar un software como ActiveX, Java, Flash, dependiendo de la aplicación.

En un RIA, la presentación gráfica se maneja en el lado del cliente, ya que tiene un plugin que proporciona soporte para gráficos ricos. En pocas palabras, la manipulación de datos en un RIA se lleva a cabo en el lado del servidor, mientras que la manipulación de objetos relacionados se lleva a cabo en el lado del cliente.

Tenemos tres tecnologías principales mediante los cuales podemos desarrollar un RIA. Estos incluyen los siguientes: -

- Adobe Flash
- Microsoft Silverlight
- JavaFX

Adobe Flash

Esta plataforma de software es desarrollado por Adobe Systems y se utiliza en la creación de aplicaciones dinámicas de Internet. Junto con estos, también se puede construir otras aplicaciones, como vector, animación, juegos de navegador, aplicaciones de escritorio, aplicaciones móviles y juegos, etc.

Esta es la plataforma más utilizada para el desarrollo y ejecución de RIA con una tasa de penetración navegador de escritorio del 96%.

Microsoft Silverlight

Al igual que Adobe Flash, Microsoft Silverlight es también un marco para el desarrollo de aplicaciones de software, así como la ejecución de aplicaciones dinámicas de Internet. Inicialmente este marco se utilizó para el streaming de medios de comunicación. Las versiones actuales soportan multimedia, gráficos y animación así.

Esta plataforma se utiliza muy poco, con una tasa de penetración navegador de escritorio del 66%.

JavaFX

JavaFX es una biblioteca de Java utilizado para construir aplicaciones dinámicas de Internet. Las aplicaciones escritas usando esta biblioteca puede funcionar constantemente a través de múltiples plataformas. Las aplicaciones desarrolladas utilizando JavaFX puede funcionar en diversos dispositivos, como ordenadores de sobremesa, teléfonos móviles, televisores, tabletas, etc.

Para desarrollar **GUI Applications usando el lenguaje de programación Java, los programadores se basan en las bibliotecas como Advanced Windowing Toolkit y Swings** . Después de la llegada de JavaFX, estos programadores de Java pueden ahora desarrollar aplicaciones GUI eficazmente con contenido rico.

Necesidad de JavaFX

Para desarrollar **Client Side Applications con las características ricas, los programadores utilizan a depender de varias bibliotecas para agregar características tales como los medios de comunicación, controles de interfaz de usuario, Web, 2D y 3D, etc. JavaFX incluye todas estas características en una sola biblioteca**. Además de éstos, los desarrolladores también pueden acceder a las funciones existentes de una biblioteca de Java tales como **Swings** .

JavaFX proporciona un rico conjunto de gráficos y de medios de API y que aprovecha la moderna **Graphical Processing Unit a través de gráficos acelerados por hardware**. JavaFX también proporciona interfaces usando cual los desarrolladores pueden combinar la animación de gráficos y de control de interfaz de usuario.

Uno puede utilizar JavaFX con tecnologías basadas en la JVM como Java, Groovy y JRuby. Si los desarrolladores optan por JavaFX, no hay necesidad de aprender tecnologías adicionales, como el conocimiento previo de cualquiera de las tecnologías antes mencionadas será lo suficientemente bueno para desarrollar el uso de JavaFX de RIA.

Características de JavaFX

Las siguientes son algunas de las características importantes de JavaFX -

- **Written in Java** - La biblioteca de JavaFX está escrito en Java y está disponible para los idiomas que se pueden ejecutar en una JVM, que incluyen - Java, Groovy and JRuby . Estas aplicaciones JavaFX también son independientes de la plataforma.
- **FXML** - JavaFX cuenta con un lenguaje conocido como FXML, que es un HTML como el lenguaje de marcado declarativo. El único propósito de este lenguaje es definir una interfaz de usuario.
- **Scene Builder** - JavaFX proporciona una aplicación llamada escena del constructor. Sobre la integración de esta aplicación en IDE, como Eclipse y NetBeans,

los usuarios pueden acceder a una interfaz de arrastrar y soltar de diseño, que se utiliza para desarrollar aplicaciones FXML (just like Swing Drag & Drop and DreamWeaver Applications) .

- **Swing Interoperability** - En una aplicación JavaFX, puede incrustar contenido de oscilación mediante el **Swing Node** clase. Del mismo modo, puede actualizar las aplicaciones Swing existentes con características JavaFX como contenido web integrado y rica medios gráficos.
- **Built-in UI controls** - JavaFX biblioteca atiende los controles de interfaz de usuario con el que podemos desarrollar una aplicación completa.
- **CSS like Styling** - JavaFX proporciona una **CSS como estilo**. Mediante el uso de esto, se puede mejorar el diseño de su aplicación con un simple conocimiento de CSS.
- **Canvas and Printing API** - JavaFX proporciona la lona, un estilo de modo inmediato de la representación de la API. Dentro del paquete `javafx.scene.canvas` que posee un conjunto de clases para el lienzo, con el que podemos dibujar directamente en una zona de la escena JavaFX. JavaFX también ofrece clases para la impresión en el paquete `javafx.print` .
- **Rich set of API's** - biblioteca JavaFX proporciona un rico conjunto de API de desarrollar aplicaciones GUI, gráficos 2D y 3D, etc. Este conjunto de API también incluye capacidades de la plataforma Java. Por lo tanto, el uso de esta API, puede acceder a las características de los lenguajes Java como los genéricos, anotaciones, multihilo y Expresiones Lambda. La biblioteca tradicional colecciones de Java se mejoró y conceptos como listas y mapas observables fueron incluidos en el mismo. El uso de estos, los usuarios pueden observar los cambios en los modelos de datos.
- **Integrated Graphics library** - JavaFX proporciona clases para 2d y 3d gráficos.
- **Graphics pipeline** - JavaFX es compatible con gráficos basados en el canal de gráficos acelerado por hardware conocido como **Prisma**. Cuando se utiliza con una tarjeta gráfica compatible o GPU que ofrece gráficos suaves. En caso de que el sistema no es compatible con la tarjeta gráfica a continuación, los valores predeterminados de prisma a la pila de software de renderizado.

Historia de JavaFX

JavaFX fue desarrollado originalmente por **Chris Oliver** , cuando trabajaba para una empresa llamada **See Beyond Technology Corporation** , que más tarde fue adquirida por **Sun Microsystems** en el año **2005**.

Los siguientes puntos nos dan más información de este proyecto -

- Inicialmente este proyecto fue nombrado como **F3 (Form Follows Functions)** y se desarrolló con la intención de proporcionar interfaces ricas para desarrollar aplicaciones GUI.
- **Sun Microsystems** adquirió la compañía ver más allá en junio de 2005, se adaptó el proyecto **F3** como **JavaFX** .
- En el año 2007, JavaFX fue anunciado oficialmente en **Java One** , en una conferencia de la **World Wide Web** que se celebra anualmente.

- En el año 2008, **Net Beans integrados con JavaFX estaba disponible**. En el mismo año, el Java **Standard Development Kit fue lanzado para JavaFX 1.0**.
- En el año 2009, Oracle Corporation adquirió Sun Microsystems y en el mismo año, la próxima versión de JavaFX (1.2) fue puesto en libertad también.
- En el año 2010, JavaFX 1.3 salió y en el año 2011 JavaFX 2.0 fue lanzado.
- La última versión, JavaFX8, fue lanzado como una parte integral de Java el 18 de de marzo de de 2014.

Instalación del entorno de Trabajo

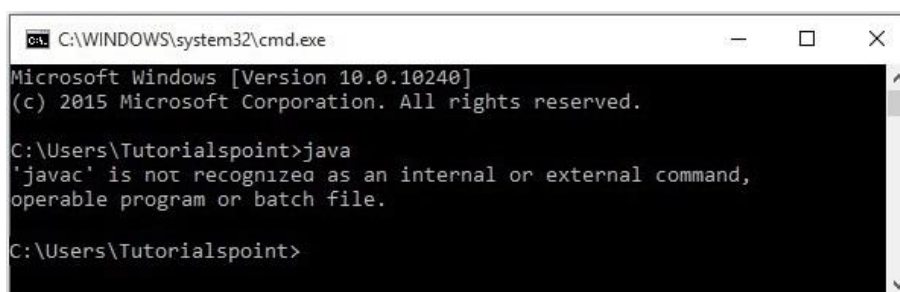
De Java8 en adelante, el JDK (**Java Development Kit**) incluye **JavaFX biblioteca en ella**. Por lo tanto, para ejecutar aplicaciones JavaFX, sólo hay que instalar Java8 o una versión posterior en su sistema.

Además de él, de IDE como Eclipse y NetBeans proporcionan soporte para JavaFX. Este capítulo le enseña cómo configurar el entorno para ejecutar aplicaciones JavaFX de diversas maneras.

Instalación Java8

En primer lugar, tendrá que comprobar si existe Java instaladas en el sistema o no abriendo el símbolo del sistema y escribiendo el comando “Java” en ella.

Si no ha instalado Java en su sistema, el símbolo del sistema muestra el mensaje que se muestra en la siguiente captura de pantalla.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Tutorialspoint>
```

A continuación, instalar Java siguiendo los pasos que se indican a continuación.

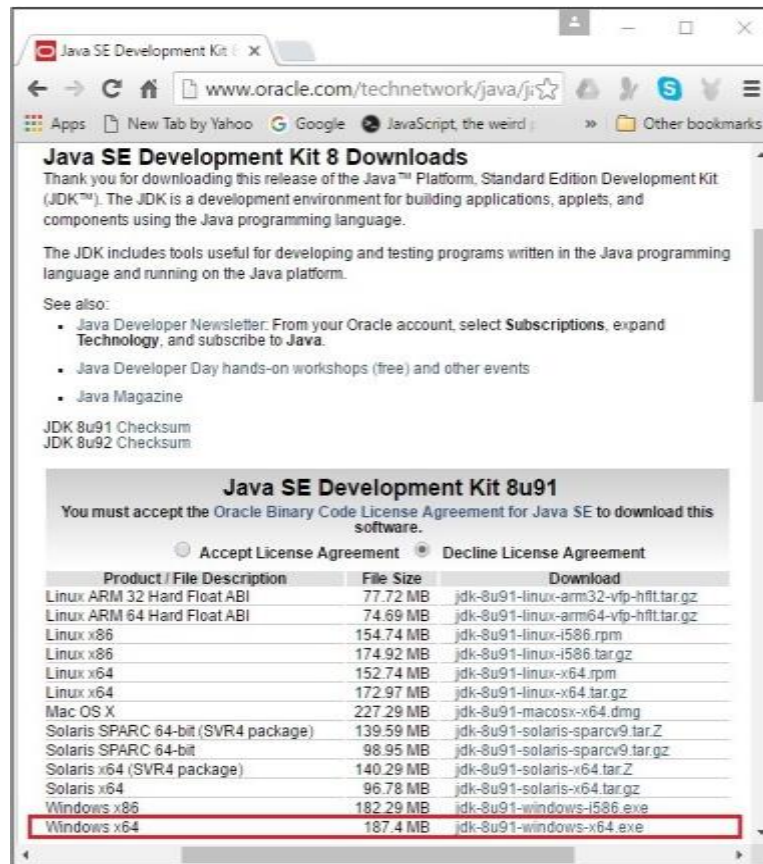
Step 1 - Visita a la JavaSE Descargas página, haga clic en el JDK Download botón como se destaca en la siguiente captura de pantalla



Step 2 - Al hacer clic en el botón de descarga, será redirigido a la Java SE Development Kit 8 Downloads página. Esta página le proporciona enlaces de JDK para varias plataformas.

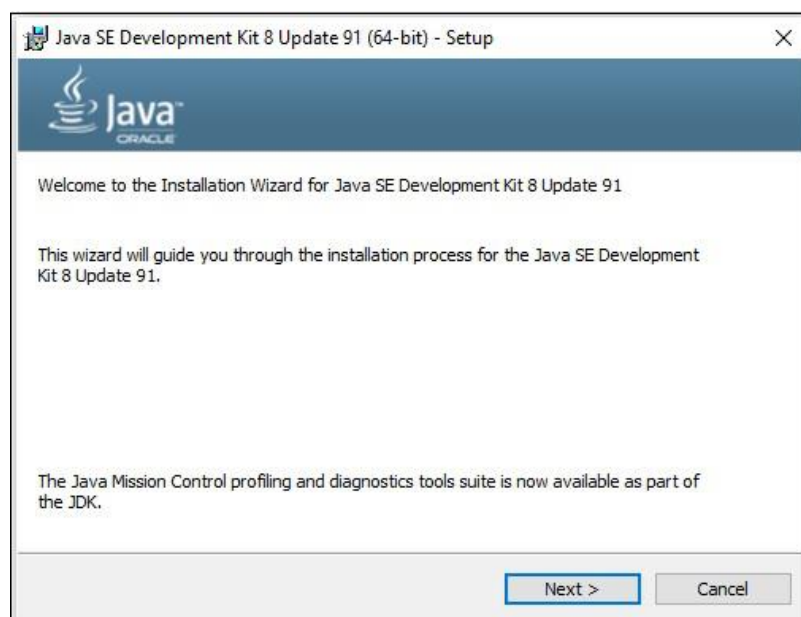
Aceptar el contrato de licencia y descargar el software necesario haciendo clic en su respectivo enlace.

Por ejemplo, si está trabajando en un sistema operativo Windows de 64 bits, entonces usted necesita para descargar la versión de JDK resaltado en la siguiente captura de pantalla.

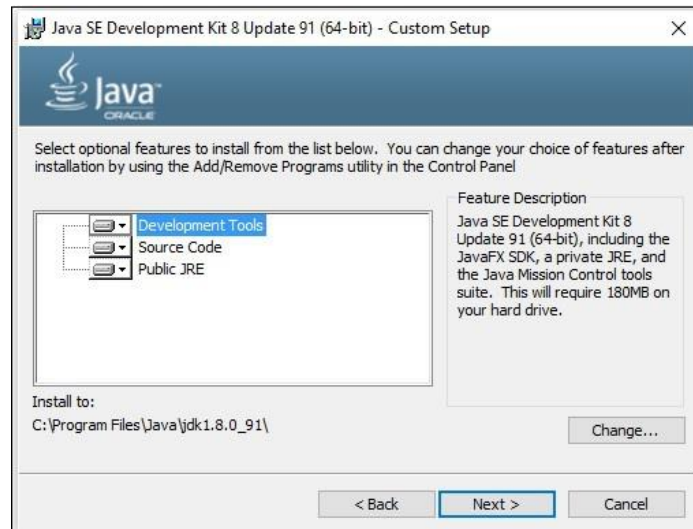


Al hacer clic sobre el enlace resaltado, el Kit de Desarrollo Java8 adecuado para el sistema operativo Windows de 64 bits estará introduciendo en su sistema.

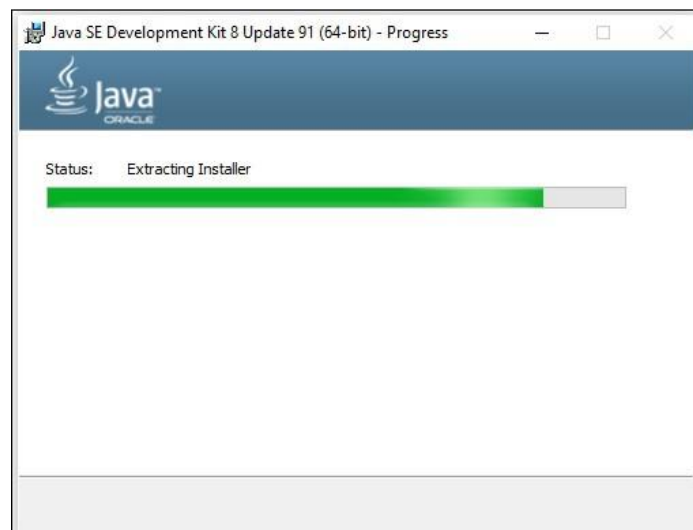
Step 3 - Ejecutar el archivo ejecutable binario descargado para iniciar la instalación de JDK8.



Step 4 - Seleccione el directorio de instalación.



Step 5 - Al seleccionar la carpeta de destino y hacer clic en Siguiente, el proceso de instalación de JavaFX empieza a mostrar la barra de progreso como se muestra en la siguiente captura de pantalla.



Step 6 - Cambiar el directorio de instalación si es necesario, de lo contrario mantener la falta de pago y seguir adelante.



Step 7 - Finalizar el proceso de instalación haciendo clic en el botón Cerrar como se muestra en la siguiente captura de pantalla.



Configuración del Camino para Windows

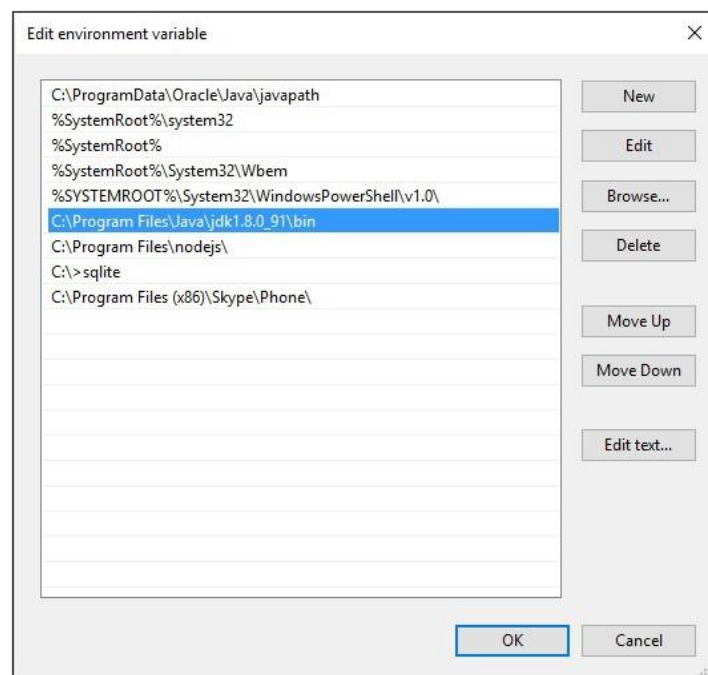
Después de instalar Java, es necesario establecer las variables de ruta. Suponga que tiene Java instalado en **C:\Program Files\java\jdk1.8.0_91 directorio**.

Ahora puede seguir los pasos que se indican a continuación -

Haga clic derecho en 'Mi PC' y seleccione 'Propiedades'.

Haga clic en el botón 'Variables de entorno' en la pestaña 'Avanzado'.

Ahora, modificar la variable 'Path' de modo que también contiene la ruta al ejecutable de Java. Por ejemplo, si la ruta está establecido actualmente en 'C: \ Windows \ System32', a continuación, cambiar su ruta para leer 'C: \ WINDOWS \ system32; C: \ Archivos de programa \ Java \ jdk1.8.0_91 \ bin'.



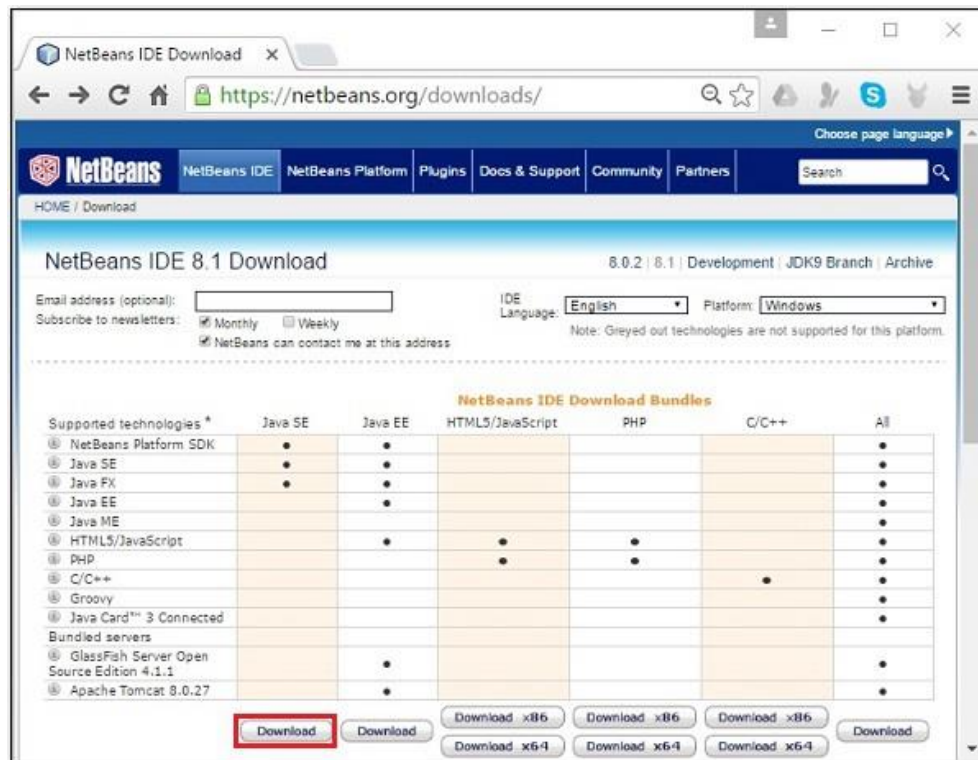
Ajuste de NetBeans Medio Ambiente de JavaFX

NetBeans8 proporciona soporte incorporado para JavaFX. Sobre la instalación de esto, se puede crear una aplicación JavaFX y sin ningún tipo de plugins adicionales o archivos JAR. Para configurar el entorno NetBeans, tendrá que seguir los pasos que se indican a continuación.

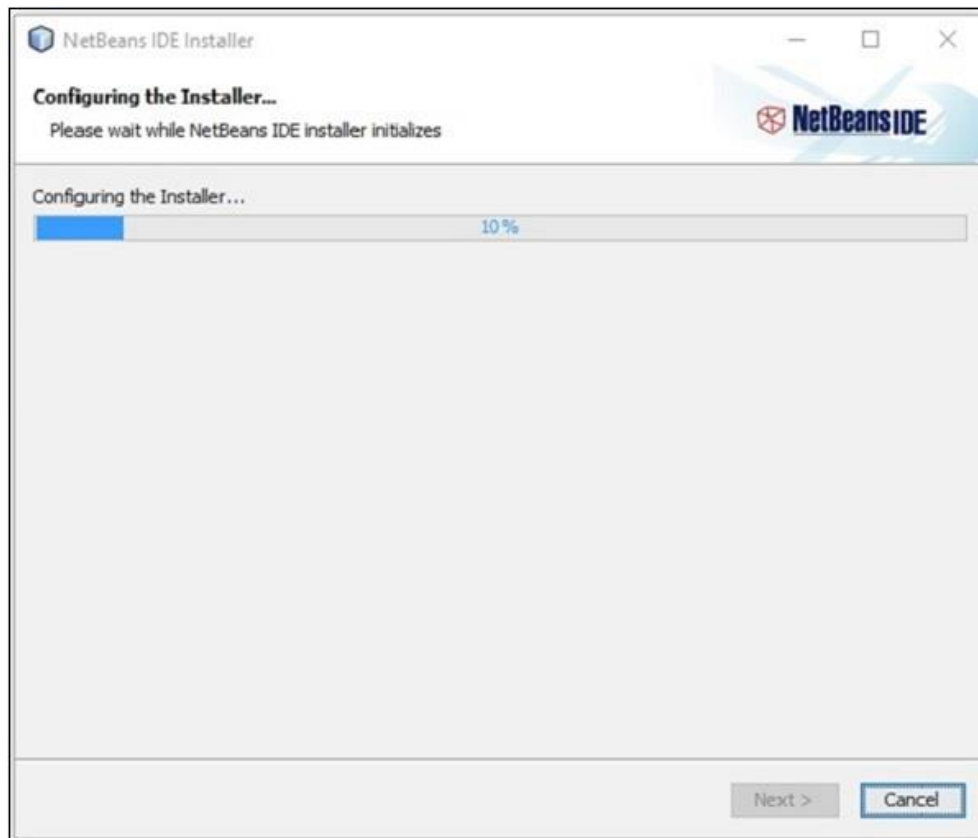
Step 1 - Visita el NetBeans sitio web sitio web de NetBeans y haga clic en el botón Descargar para descargar el software NetBeans.



Step 2 - Al hacer clic en Download , se llega a la página de descargas del software NetBeans, que ofrece paquetes de NetBeans para diversas aplicaciones Java. Descargar el software NetBeans para JavaSE como se muestra en la siguiente captura de pantalla.

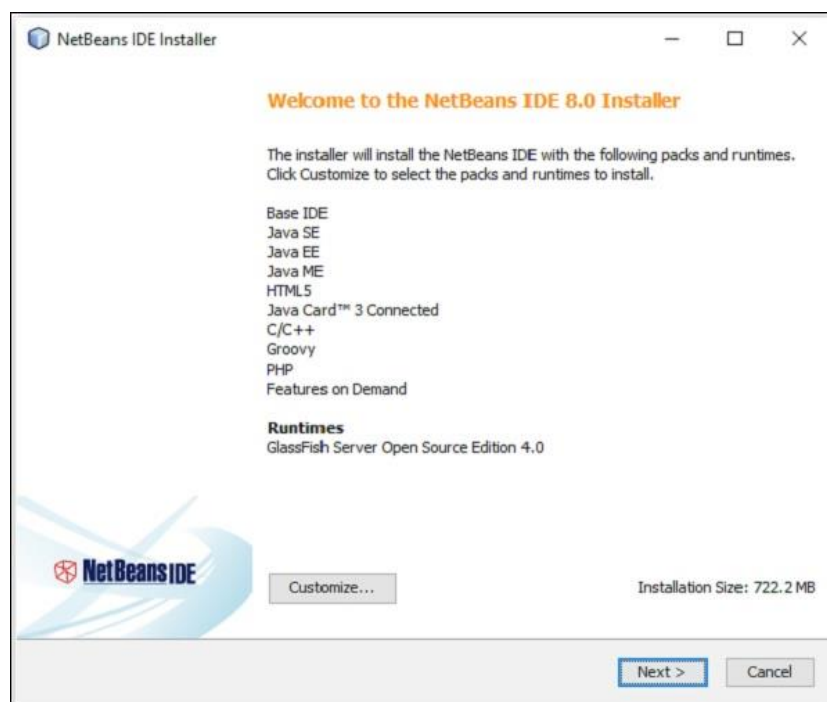


Step 3 - Al hacer clic sobre este botón, un archivo llamado netbeans-8.0-windows.exe estará introduciendo en su sistema. Ejecutar este archivo con el fin de instalarlo. Al ejecutar este archivo, un instalador de NetBeans se iniciará como se muestra en la siguiente captura de pantalla.

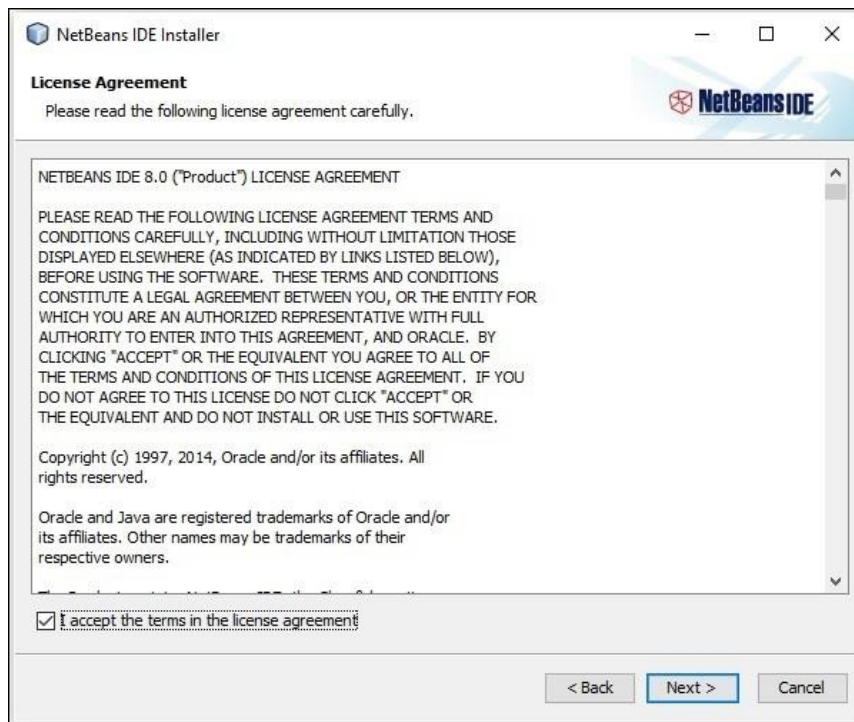


Después de completar la configuración, verá la **Welcome Page of the installer**.

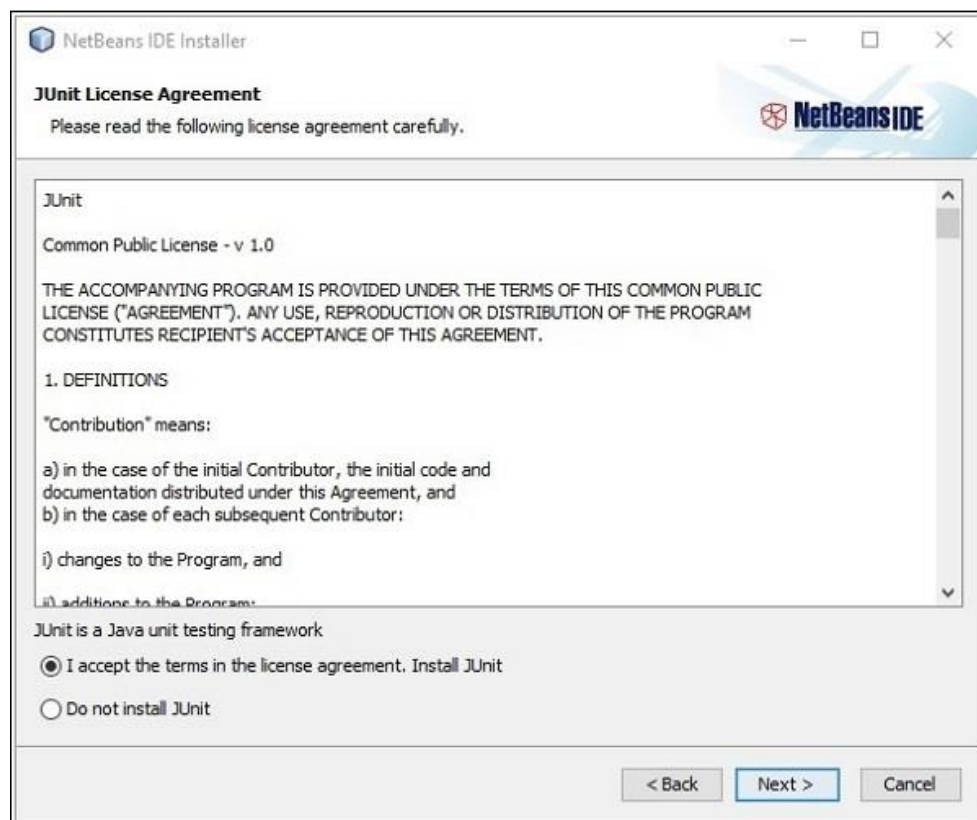
Step 4 - Haga clic en el botón Siguiente y continúe con la instalación.



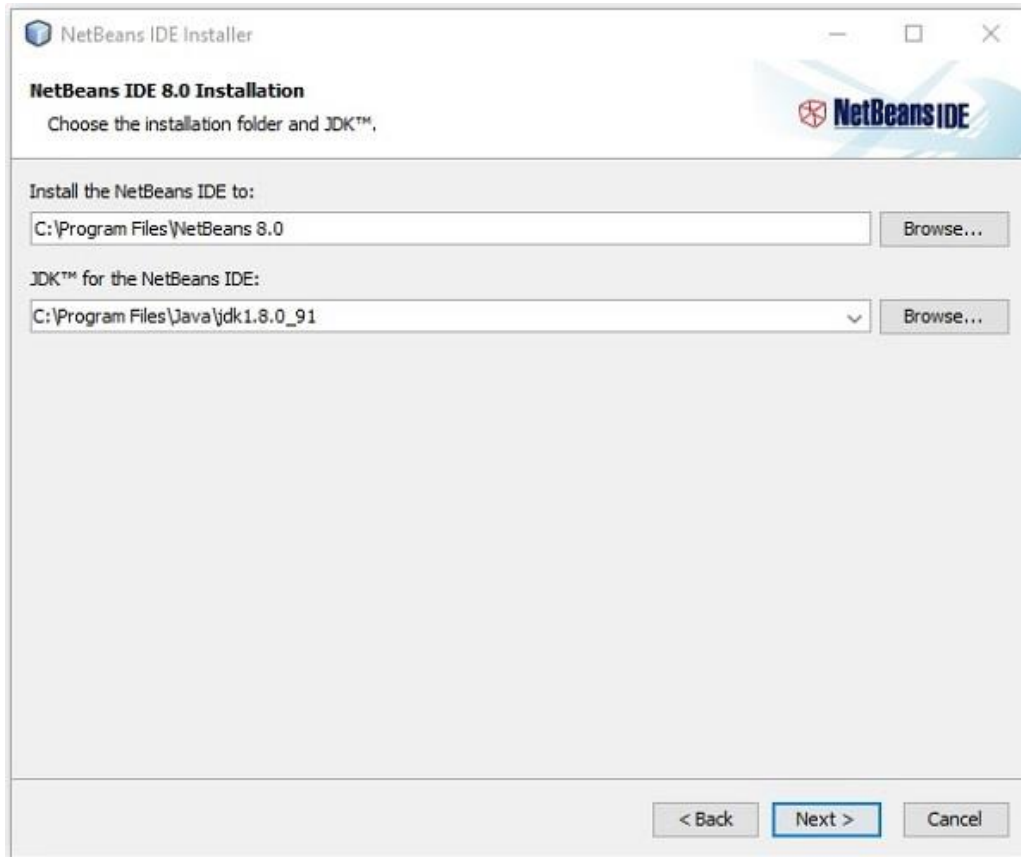
Step 5 - La siguiente ventana tiene el NETBEANS IDE 8.0 license agreement . Leer cuidadosamente y acepte el acuerdo marcando la casilla de verificación en “Acepto los términos del contrato de licencia” y haga clic en el **Next botón.**



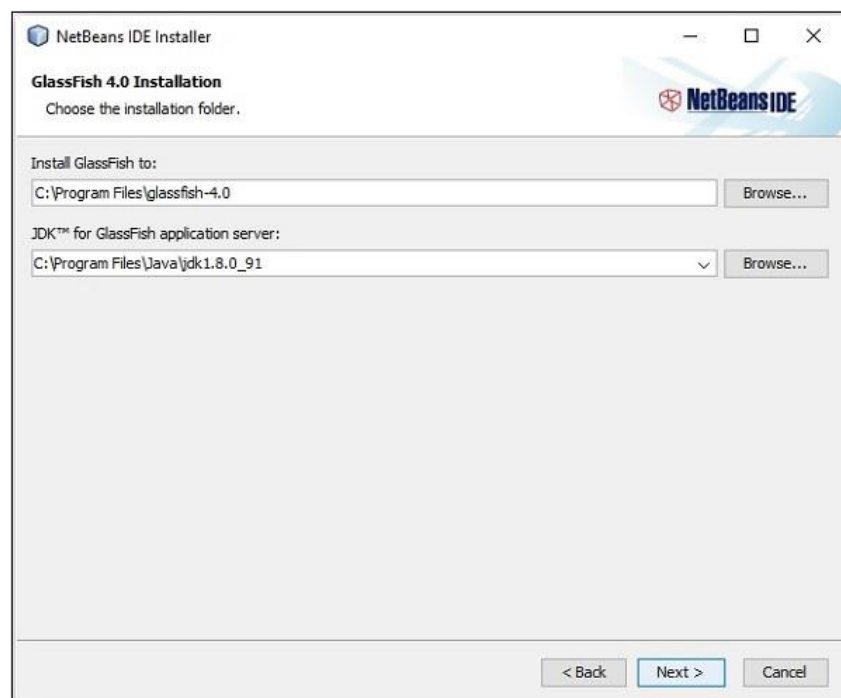
Step 6 - En la siguiente ventana, se encontrará con el contrato de licencia para JUnit , aceptarlo seleccionando el botón de radio en “Acepto los términos del contrato de licencia, JUnit Instalar” y haga clic en Next .



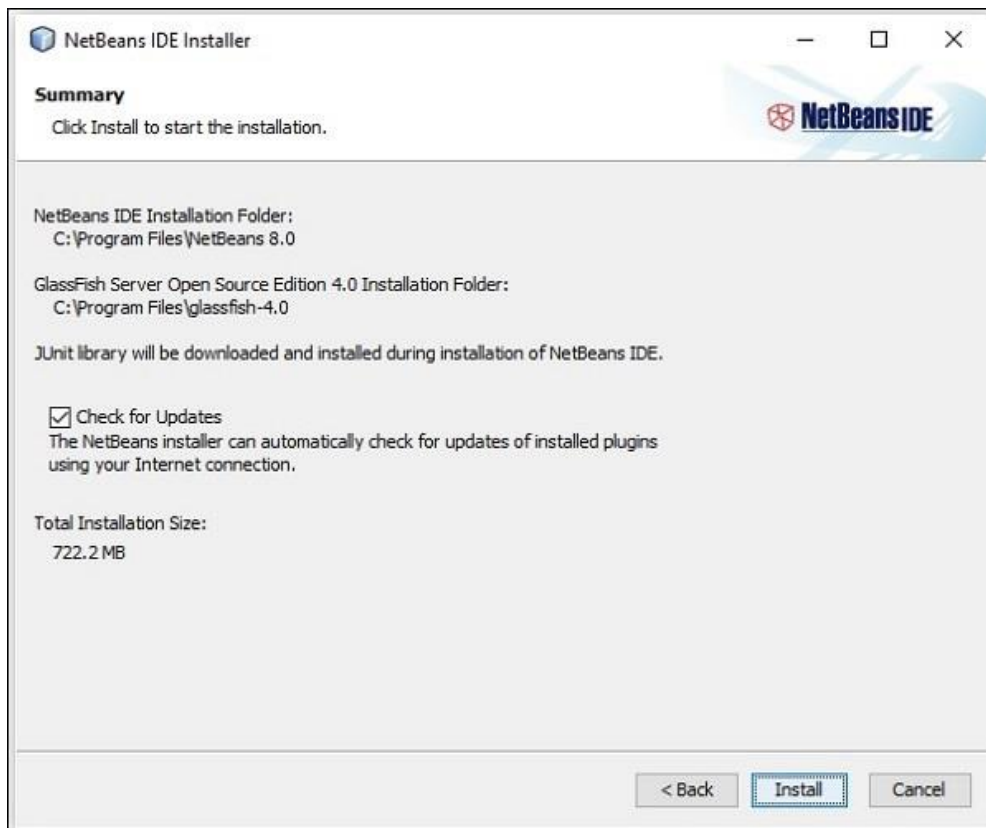
Step 7 - Seleccione el directorio de destino donde se necesita el Netbeans 8.0 a ser instalado.
Por otra parte, también se puede navegar a través del directorio en el **Java Development Kit** está instalado en su sistema y haga clic en el **Next** botón.



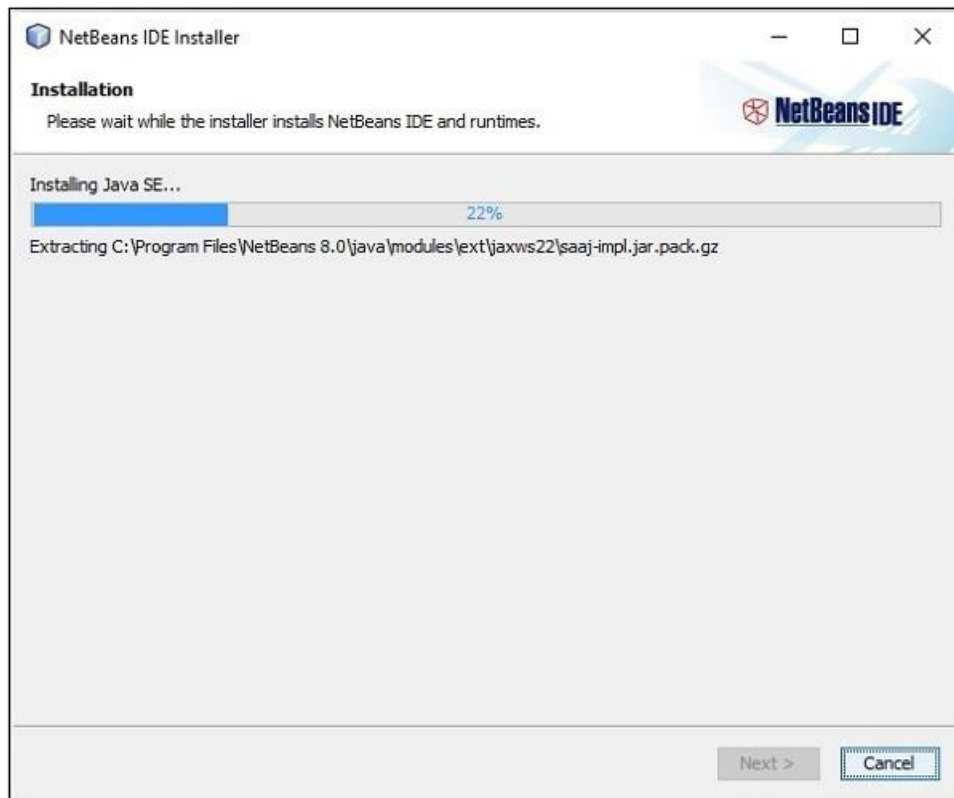
Step 8 - Del mismo modo, elegir el directorio de destino de Glassfish Server instalación.
Navegar por el directorio de Java Development Kit (now for Glassfish Reference) y luego haga clic en **Next** .



Step 9 - Compruebe el Check for Updates caja para actualizaciones automáticas y haga clic en el botón Instalar para iniciar la instalación.

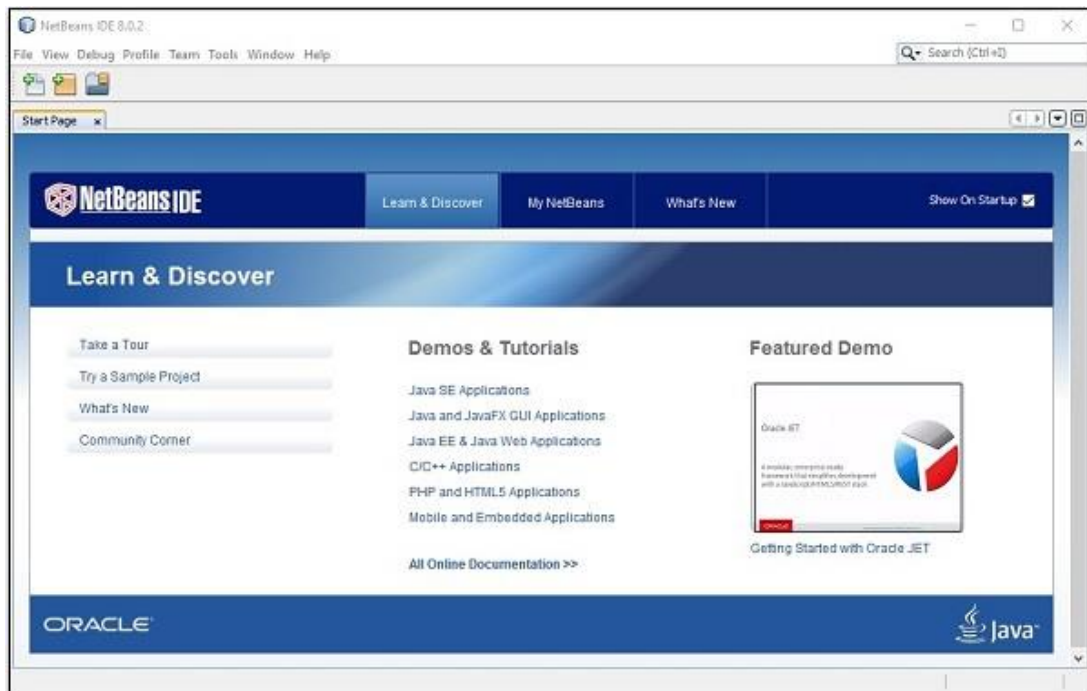


Step 10 - Este paso inicia la instalación de NetBeans IDE 8.0 y puede tomar un tiempo.

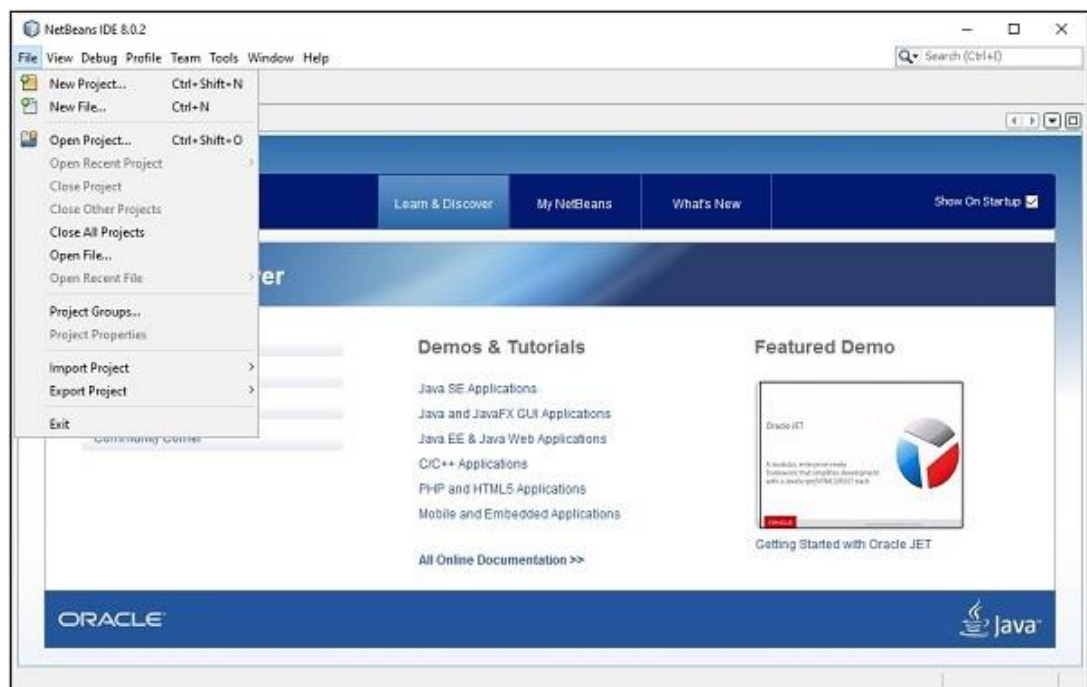


Step 11 - Una vez que el proceso se haya completado, haga clic en el Finish botón para finalizar la instalación.

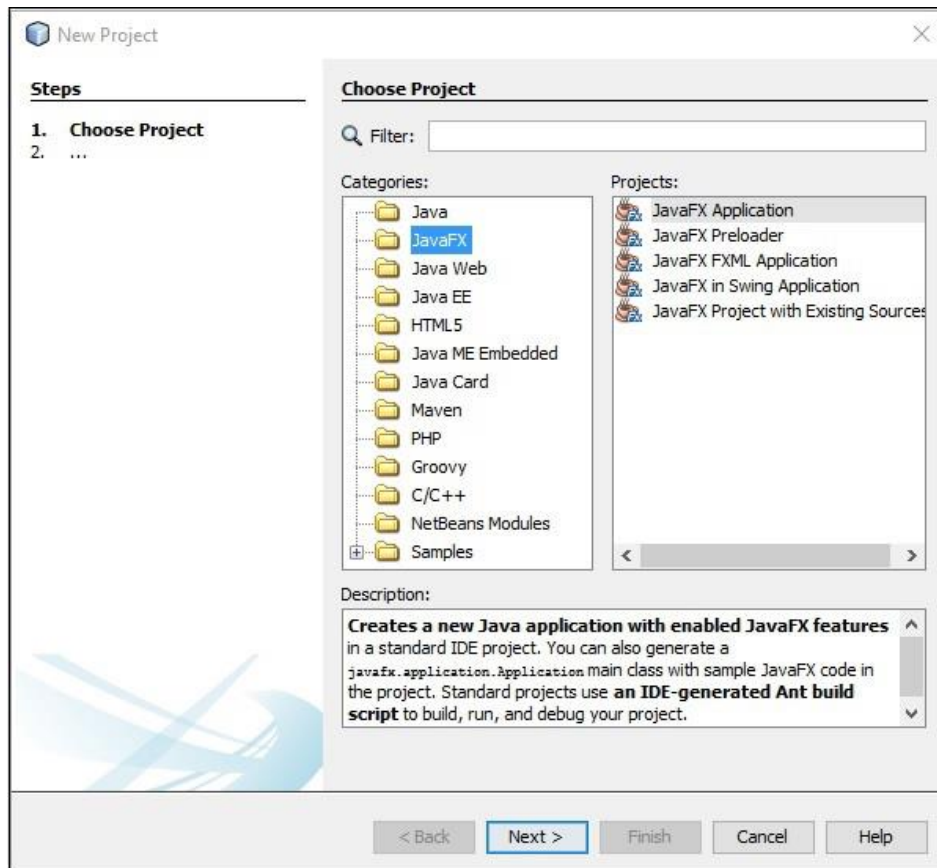
Step 12 - Una vez que inicie el IDE NetBeans, verá la página de inicio como se muestra en la siguiente captura de pantalla.



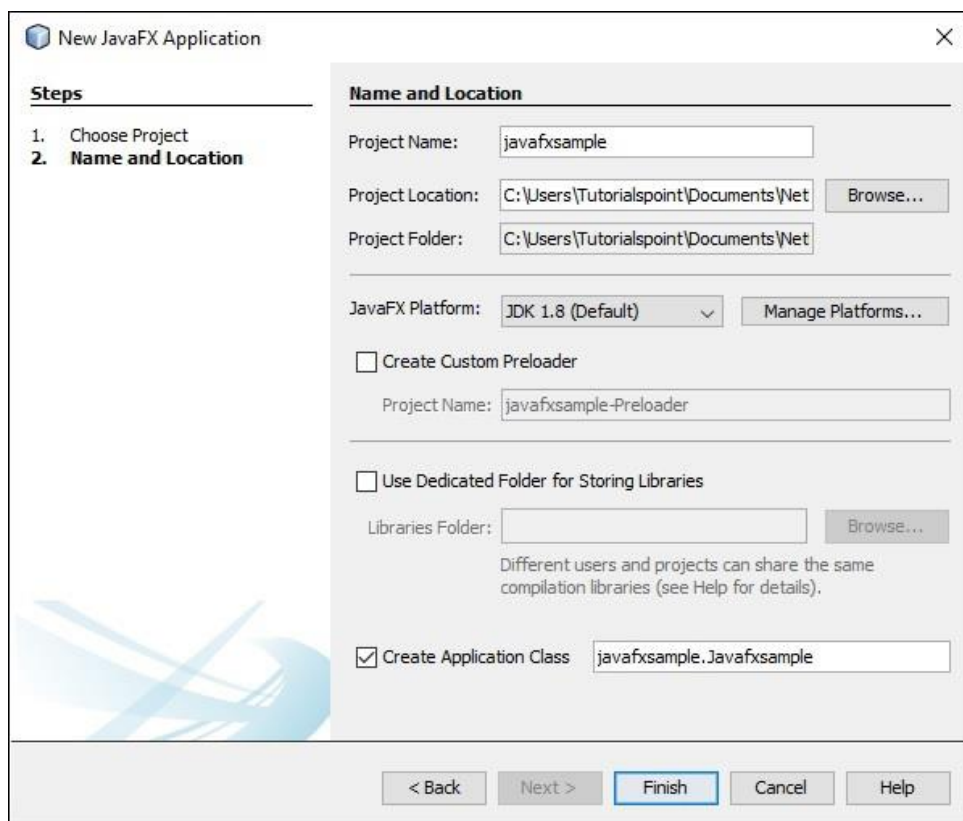
Step 13 - En el menú Archivo, seleccione New Project ... para abrir el asistente de nuevo proyecto como se muestra en la siguiente captura de pantalla.



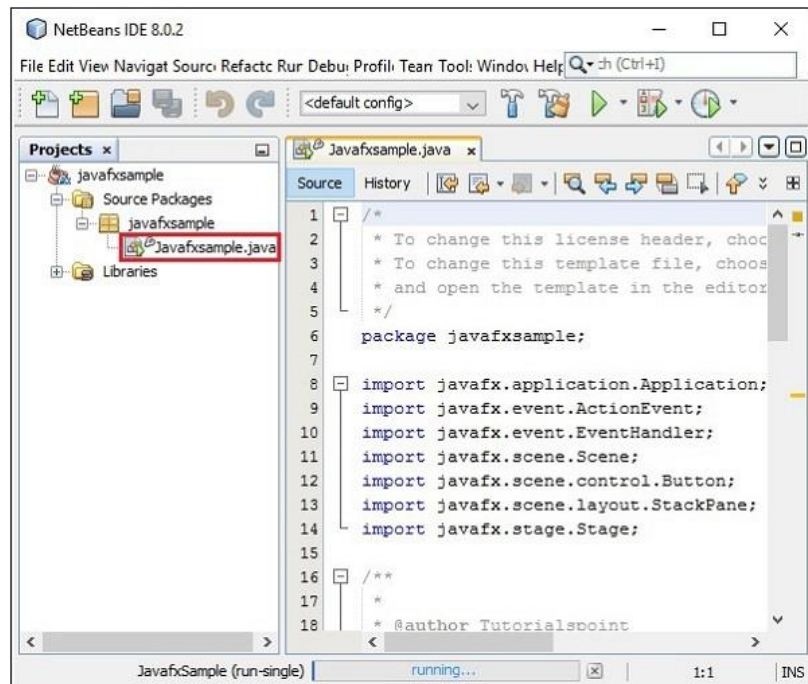
Step 14 - En el New Project del asistente, seleccione JavaFX y haga clic en Next . Se empieza a crear una nueva aplicación JavaFX para usted.



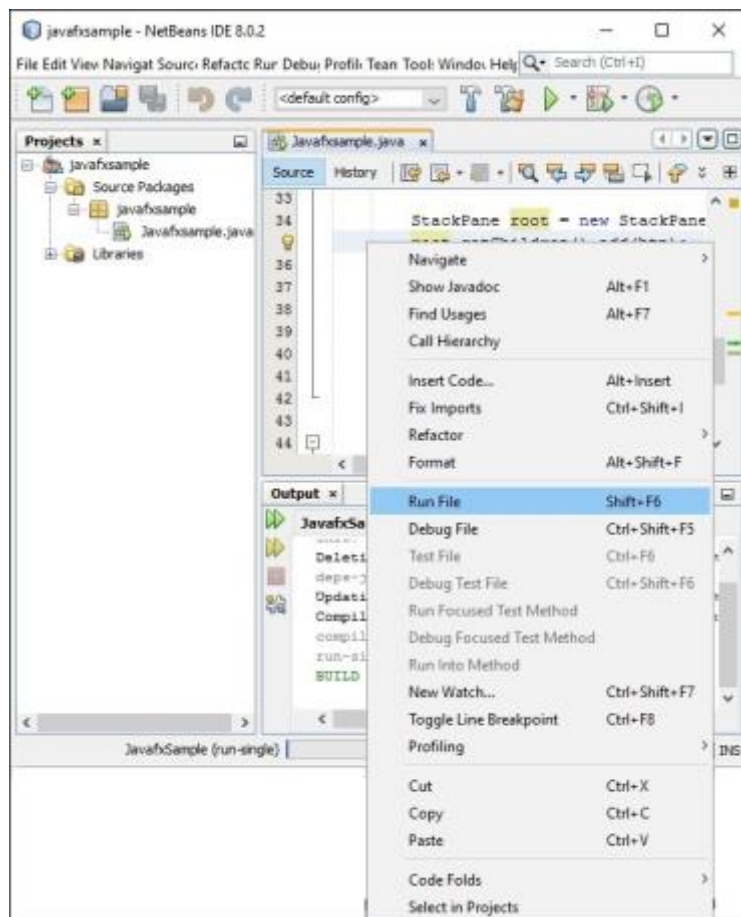
Step 15 - Seleccione el nombre del proyecto y la ubicación del proyecto en la NewJavaFX Application ventana y luego haga clic Finish . Se crea una aplicación de ejemplo con el nombre dado.



En este caso, una aplicación con un nombre **javafxsample** se crea. Dentro de esta aplicación, el IDE NetBeans generará un programa Java con el nombre **Javafxsample.java** . Como se muestra en la siguiente captura de pantalla, este programa se creará dentro de los paquetes de NetBeans Fuente → **javafxsample** .



Step 16 - Haga clic derecho sobre el archivo y seleccione Run File para ejecutar este código como se muestra en la siguiente captura de pantalla.



Este programa crea automáticamente contiene el código que genera una ventana de JavaFX sencilla que tiene un botón con la etiqueta **Say 'Hello World' en ella**. Cada vez que se hace clic en este botón, la cadena **Hello World** se mostrará en la consola como se muestra a continuación.

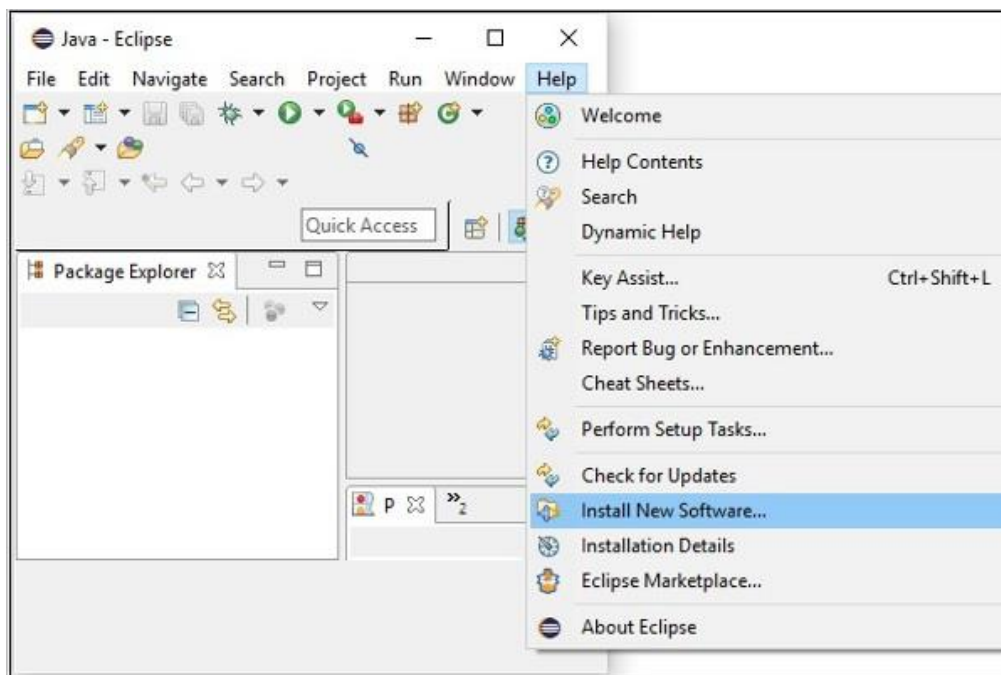


La instalación de JavaFX en Eclipse

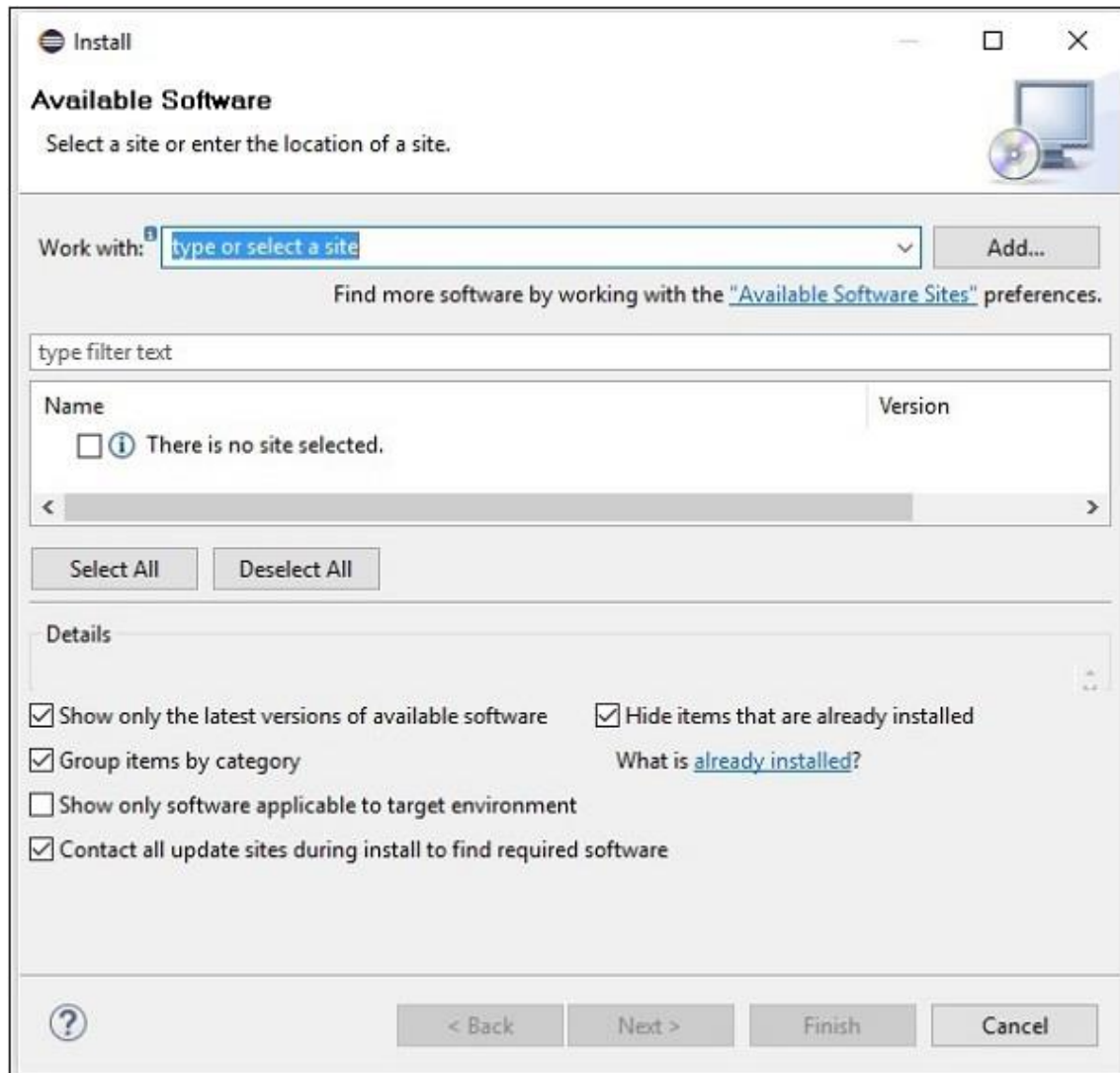
Un plugin llamado **e(fx)clipse** también está disponible en JavaFX. Se pueden utilizar los siguientes pasos para configurar JavaFX en Eclipse. En primer lugar, asegúrese de que usted ha Eclipse en su sistema. Si no es así, descargar e instalar Eclipse en su sistema.

Una vez instalado Eclipse, siga los pasos que se indican a continuación para instalar **e(fx)clipse** en su sistema.

Step 1 - Abrir Eclipse en la Help de menú y seleccione Install New Software ... opción como se muestra a continuación.



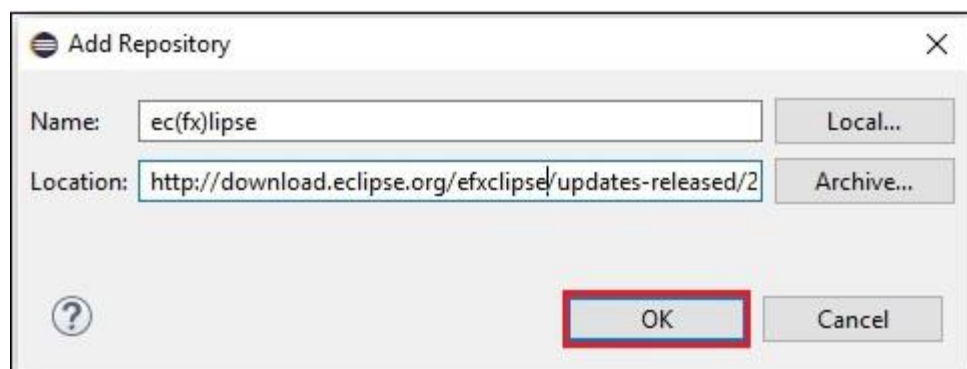
Al hacer clic, se mostrará el **Available Software** ventana, como se muestra en la siguiente captura de pantalla.



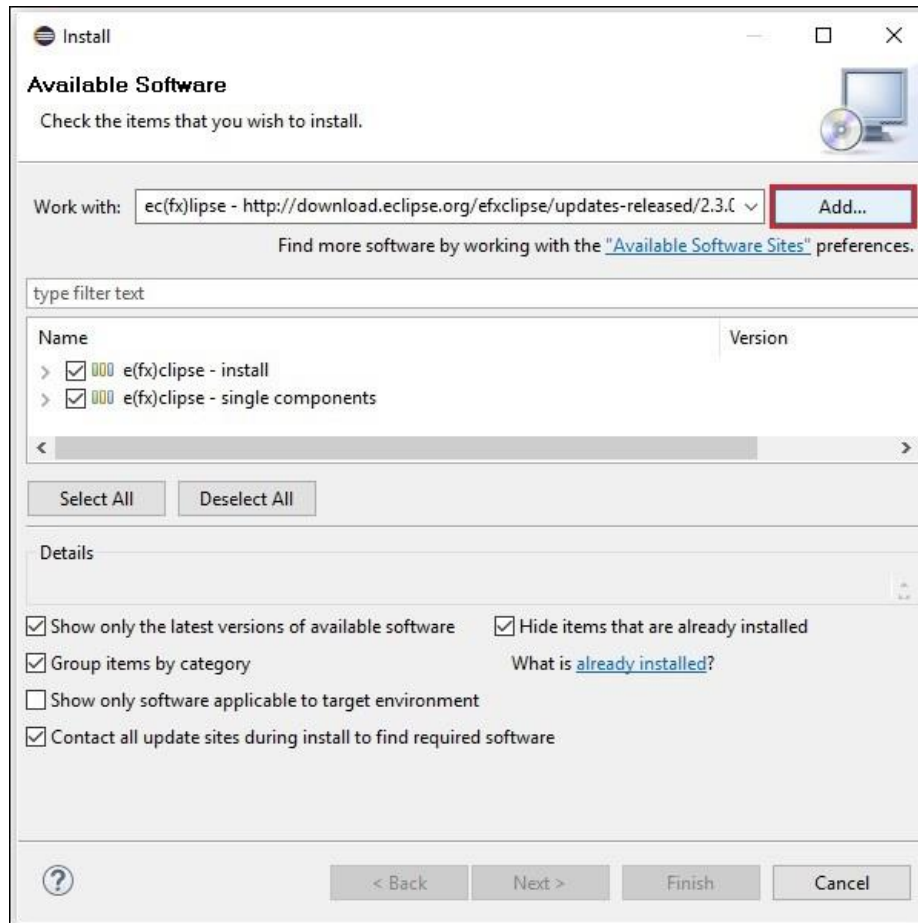
En el cuadro de texto **Work with** de esta ventana, es necesario proporcionar el enlace del plug-in para el software necesario.

Step 2 - Haga clic en el Add ... botón. Proporcionar el nombre del plugin como **e(fx)clipse** . A continuación, Proporcionar el siguiente enlace como localización. <http://download.eclipse.org/efxclipse/updates-released/2.3.0/site/>

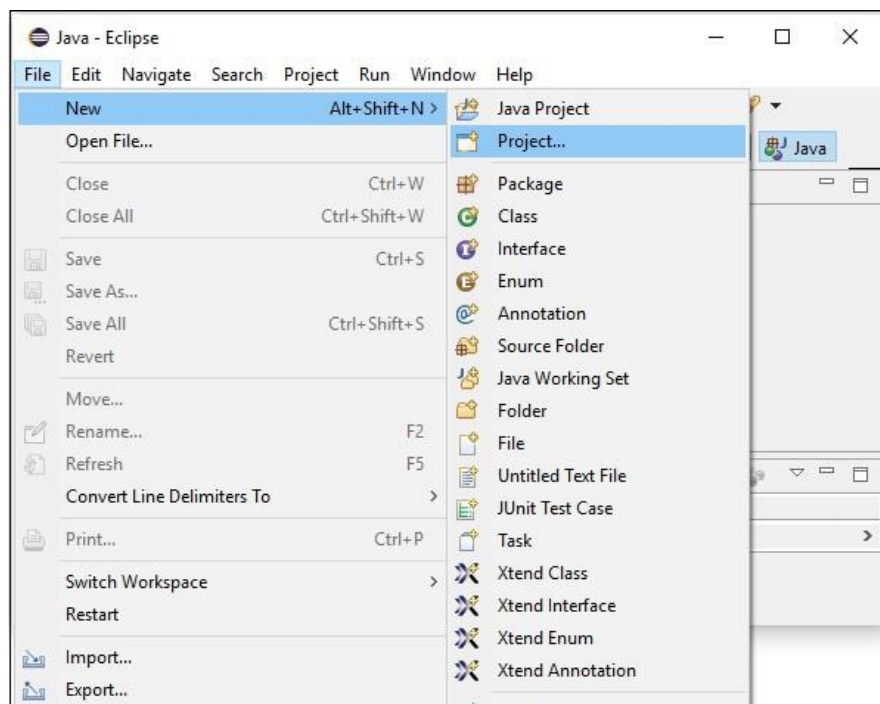
Step 3 - Después de especificar el nombre y la ubicación del complemento, haga clic en el botón Aceptar, como se destaca en la siguiente captura de pantalla.



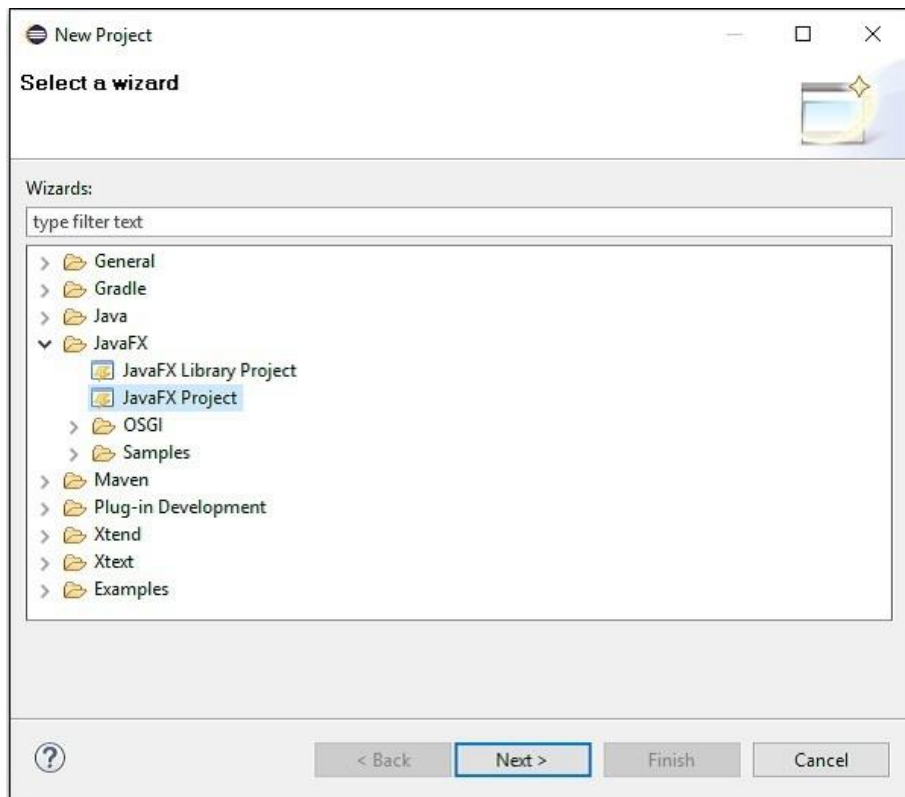
Step 4 - Poco después de agregar el plugin, se encuentran dos casillas de verificación de e(fx)clipse – install y e(fx)clipse – single components . Compruebe estas dos casillas de verificación y haga clic en el Add ... botón como se muestra en la siguiente captura de pantalla.



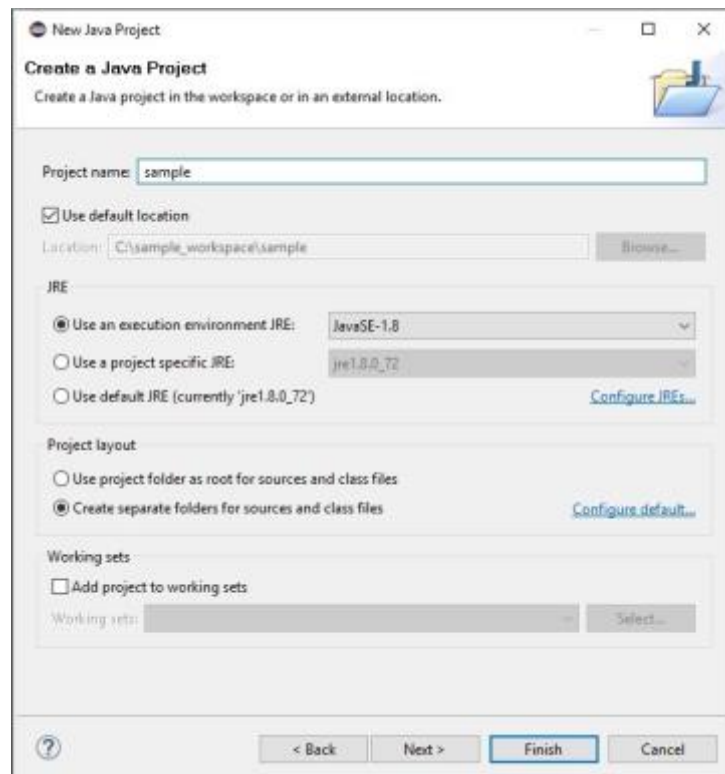
Step 5 - A continuación, abra el IDE Eclipse. Haga clic en el menú Archivo y seleccione Proyecto como se muestra en la siguiente captura de pantalla.



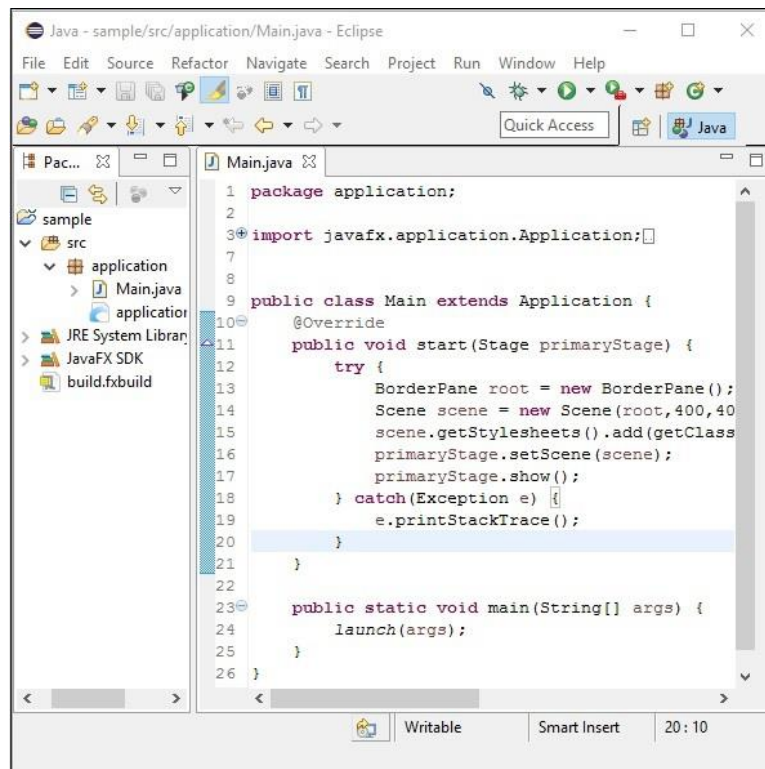
Step 6 - A continuación, aparecerá una ventana donde se puede ver una lista de asistentes proporcionados por Eclipse para crear un proyecto. Ampliar el JavaFX asistente, seleccione JavaFX Project y haga clic en el Next botón como se muestra en la siguiente captura de pantalla.



Step 7 - Al hacer clic en Next , se abre un Asistente para nuevos proyectos. A continuación, puede escribir el nombre del proyecto necesario y haga clic Finish .



Step 8 - Al hacer clic en **Finalizar**, una aplicación se crea con el nombre dado (sample) . En el sub-paquete llamado **application** , un programa con el nombre **Main.java** se genera como se muestra a continuación.

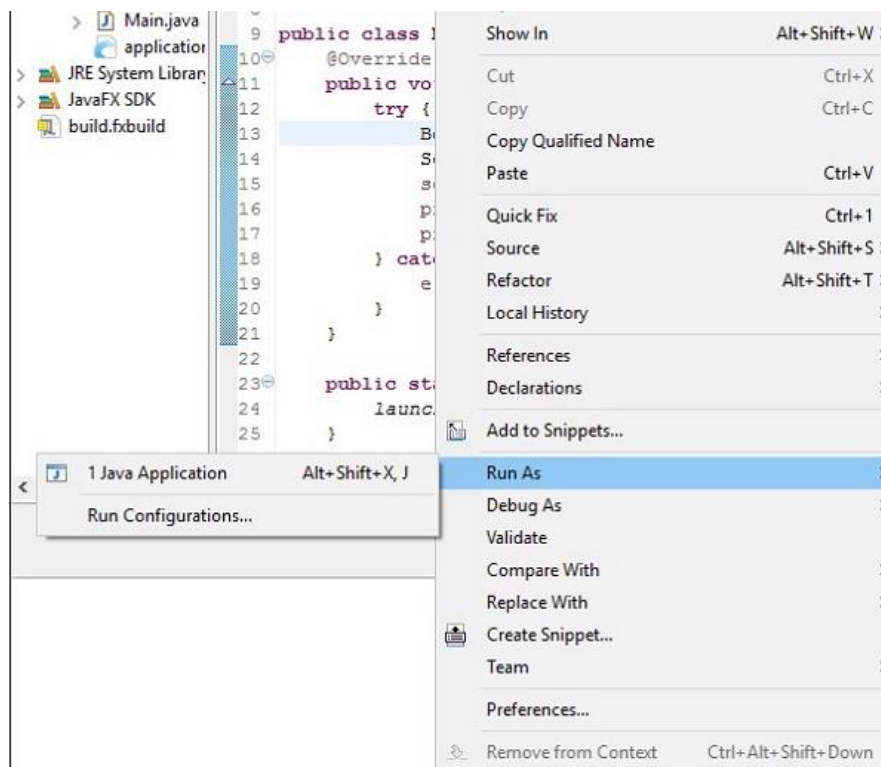


```

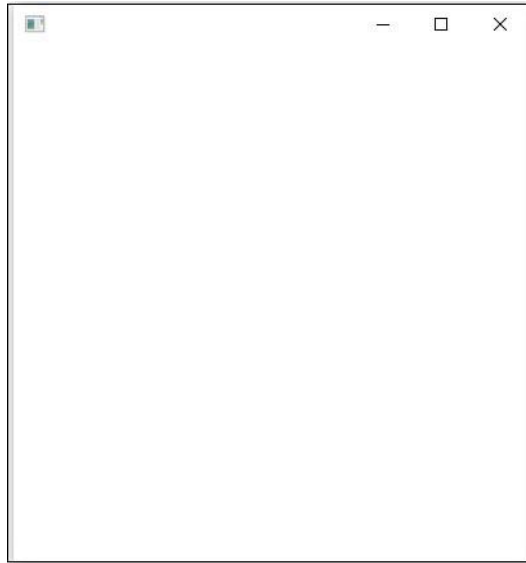
1 package application;
2
3 import javafx.application.Application;
4
5
6
7
8
9 public class Main extends Application {
10     @Override
11     public void start(Stage primaryStage) {
12         try {
13             BorderPane root = new BorderPane();
14             Scene scene = new Scene(root, 400, 400);
15             scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
16             primaryStage.setScene(scene);
17             primaryStage.show();
18         } catch (Exception e) {
19             e.printStackTrace();
20         }
21     }
22
23     public static void main(String[] args) {
24         launch(args);
25     }
26 }

```

Step 9 - Este programa genera automáticamente contiene el código para generar una ventana de JavaFX vacía. Haga clic en este archivo, seleccione **Run As → Java Application** como se muestra en la siguiente captura de pantalla.



Al ejecutar esta aplicación, se le da una ventana de JavaFX vacía como se muestra a continuación.

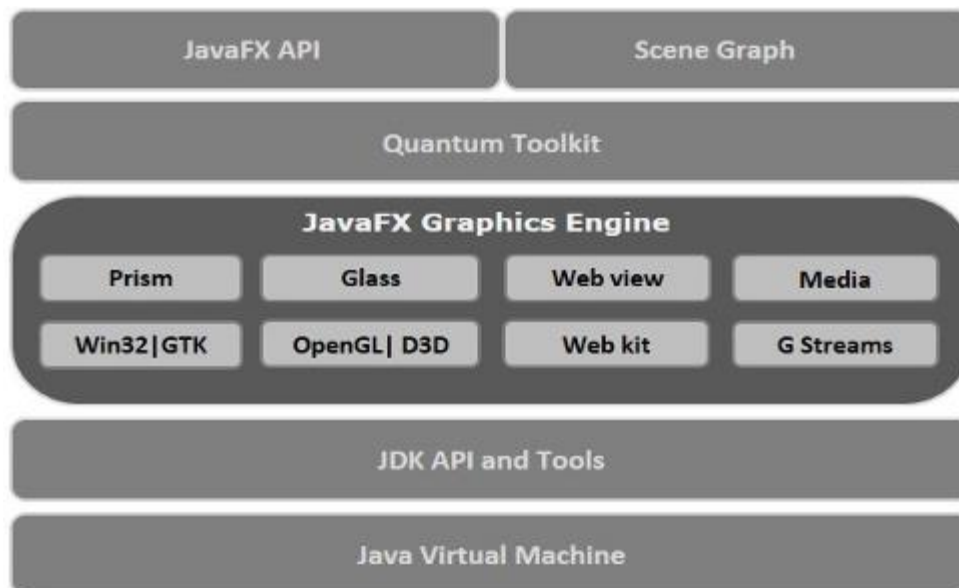


Arquitectura de JavaFX

JavaFX proporciona una API completa con un rico conjunto de clases e interfaces para construir aplicaciones GUI con gráficos ricos. Los paquetes importantes de esta API son -

- **javafx.animation** - Contiene clases para agregar animaciones basadas transición tales como relleno, se desvanecen, rotar, escalar y la traducción, a los nodos de JavaFX.
- **javafx.application** - Contiene un conjunto de clases responsables del ciclo de vida de las aplicaciones JavaFX.
- **javafx.css** - Contiene clases para añadir estilo CSS-como a las aplicaciones JavaFX GUI.
- **javafx.event** - Contiene clases e interfaces para entregar y manejar eventos JavaFX.
- **javafx.geometry** - Contiene clases para definir objetos 2D y realizar operaciones con ellos.
- **javafx.stage** - Este paquete contiene las clases de contenedor de nivel superior para aplicaciones JavaFX.
- **javafx.scene** - Este paquete proporciona clases e interfaces para apoyar el escenario gráfico. Además, también proporciona sub-paquetes, tales como lienzo, tabla, control, efecto, imagen, de entrada, diseño, medios, pintura, forma, texto, transformar, tela, etc Hay varios componentes que apoyan esta rica API de JavaFX .

La ilustración siguiente muestra la arquitectura del API JavaFX. Aquí puede ver los componentes que soportan la API de JavaFX.



Escenario gráfico

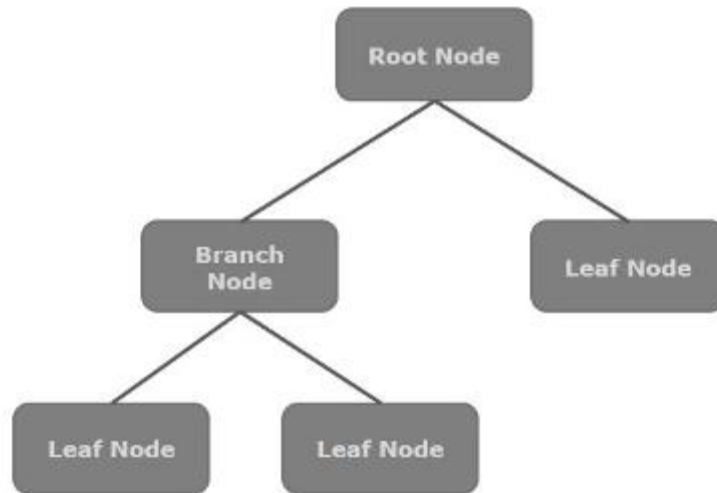
En JavaFX, las aplicaciones GUI fueron codificados usando un escenario gráfico. Un gráfico de la escena es el punto de partida de la construcción de la aplicación GUI. Contiene los (GUI) primitivas de aplicación que se ha denominado como nodos.

Un nodo es un objeto visual / gráfica y puede incluir -

- **Geometrical (Graphical) objects** - (2D and 3D) como círculo, rectángulo, polígono, etc.
- **UI controls** - como Botón, Casilla de verificación, cuadro de selección, área de texto, etc.

- **Containers - (layout panes)** , tales como panel de Border, Cuadrícula Pane, la Ventana de flujo, etc.
- **Media elements - como objetos de audio, vídeo e imagen.**

En general, una colección de nodos hace un gráfico de escena. Todos estos nodos están dispuestos en un orden jerárquico tal como se muestra a continuación.



Cada nodo en el gráfico de la escena tiene un solo padre, y el nodo que no contiene ningún padre es conocido como el **root node** .

De la misma manera, cada nodo tiene uno o más hijos, y el nodo sin hijos que se denomina como **leaf node** ; un nodo con los niños que se denomina como un **branch node** .

Una instancia de nodo se puede añadir a un escenario gráfico sólo una vez. Los nodos de un grafo de escena puede tener efectos, la opacidad, transformaciones, controladores de eventos, controladores de eventos, estados específicos de aplicación.

Prisma

Prisma es una **high performance hardware–accelerated graphical pipeline que se utiliza para representar los gráficos en JavaFX**. Se puede hacer tanto en 2-D y 3-D gráficos.

Para representar gráficos, utiliza un prisma -

- DirectX 9 en Windows XP y Vista.
- DirectX 11 en Windows 7.
- OpenGL en Mac y Linux, Embedded Systems.

En caso de que el soporte de hardware para gráficos en el sistema no es suficiente, entonces Prisma utiliza el software de render senda para procesar los gráficos.

Cuando se utiliza con una tarjeta gráfica compatible o GPU, que ofrece gráficos más suaves. Sólo en caso de que el sistema no admite una tarjeta gráfica, a continuación, los valores predeterminados de prisma en la pila de software de renderizado (either of the above two) .

GWT (Glass Windowing Toolkit)

Como su nombre indica, GWT ofrece servicios para la gestión de Windows, temporizadores, superficies y las colas de eventos. GWT conecta la plataforma JavaFX para el sistema operativo nativo.

Quantum Toolkit

Es una abstracción sobre los componentes de bajo nivel de Prisma, vidrio, Media Engine y Web del motor. Se ata Prisma y GWT juntos y los pone a disposición de JavaFX.

WebView

El uso de JavaFX, también se puede incorporar contenido HTML en un escenario gráfico. WebView es el componente de JavaFX que se utiliza para procesar este contenido. Se utiliza una tecnología llamada **Web Kit , que es un motor de renderizado de código abierto interno**. Este componente es compatible con diferentes tecnologías web como HTML5, CSS, JavaScript, DOM y SVG.

El uso de WebView, que puede -

- Representar el contenido HTML de URL local o remoto.
- Historial de soporte y proporcionar Atrás y navegación hacia adelante.
- Actualizar el contenido.
- Aplicar efectos al componente web.
- Editar el contenido HTML.
- Ejecutar comandos de JavaScript.
- Manejar eventos.

En general, el uso de vista Web, se puede controlar el contenido web de Java.

Media Engine

El **JavaFX media engine se basa en un motor de código abierto conocido como un Streamer .** Este motor multimedia es compatible con la reproducción de vídeo y audio.

El motor de medios JavaFX proporciona soporte para audio de los siguientes formatos de archivo -

Audio	MP3
	WAV
	AIFF
Video	FLV

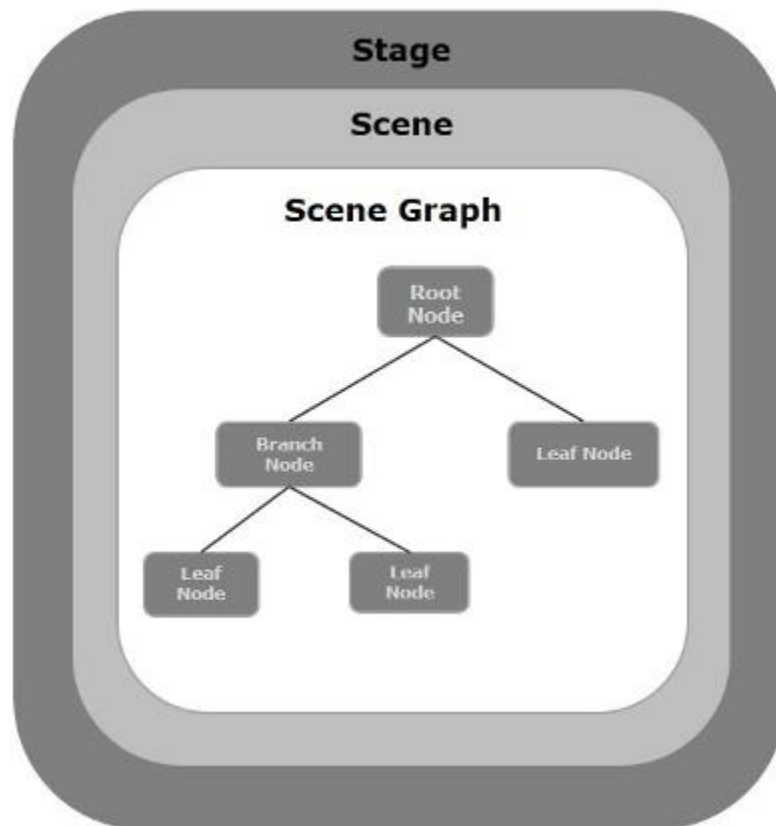
El paquete **javafx.scene.media** contiene las clases e interfaces para proporcionar la **funcionalidad multimedia en JavaFX**. Se proporciona en la forma de tres componentes, que son:

- **Media Object** - Esto representa un archivo multimedia
- **Media Player** - Para reproducir contenido multimedia.

- **Media View** - Para mostrar los medios de comunicación.

Esquema de una Aplicación Básica

En general, una aplicación JavaFX tendrá tres componentes principales, a saber **Stage**, **Scene** y **Nodes** como se muestra en el siguiente diagrama.



Escenario

Una etapa (a window) contiene todos los objetos de una aplicación JavaFX. Se representa por **Stage de clases del paquete javafx.stage** . La etapa primaria es creado por la propia plataforma. El objeto creado etapa se pasa como argumento al **start()** método de la **Application de clase** (explained in the next section) .

Una etapa tiene dos parámetros que determinan su posición a saber **Width y Height** . Se divide como Área de contenido y decoraciones (Title Bar and Borders) .

Hay cinco tipos de etapas disponibles:

- Decorado
- sin decorar
- Transparente
- unificada
- Utilidad

Tienes que llamar al **show()** método para mostrar el contenido de un escenario.

Escena

Una escena representa el contenido físico de una aplicación JavaFX. Contiene todos los contenidos de un escenario gráfico. La clase **Scene** del paquete **javafx.scene** representa el **objeto de escena**. En un caso, se añade el objeto de escena a una sola etapa.

Puede crear una escena creando una instancia de la clase de escena. Se puede optar por el tamaño de la escena que pasa por sus dimensiones (height and width) junto con el **root node** de su constructor.

Gráfico escena y Nodos

Un scene graph es una estructura de datos en forma de árbol (hierarchical) que representa el contenido de una escena. En contraste, un node es un objeto visual / gráfica de un gráfico de la escena.

Un nodo puede incluir:

- Geométricas (Graphical) objetos (2D and 3D) , tal como - círculo, rectángulo, polígono, etc.
- Los controles de interfaz de usuario, tales como - Botón, Casilla de verificación, Elección de pelo, Servicio de texto, etc.
- Contenedores (Layout Panes) , tales como panel de Border, Cuadrícula Pane, la Ventana de flujo, etc.
- elementos multimedia, como audio, vídeo y objetos de imagen.

El Node Clase del paquete javafx.scene representa un nodo en JavaFX, esta clase es la superclase de todos los nodos.

Como se señaló anteriormente un nodo es de tres tipos:

- Root Node - la primera escena Graph es conocido como el nodo raíz.
- Branch Node/Parent Node - El nodo con nodos secundarios son conocidos como nodos rama / padres. La clase abstracta llamada Parent del paquete javafx.scene es la clase base de todos los nodos padre, y esos nodos padre será de los siguientes tipos -
- Group - Un nodo de grupo es un nodo colectiva que contiene una lista de nodos hijos. Siempre que se representa el nodo de grupo, todos sus nodos hijos son prestados en orden. Cualquier transformación, efecto de estado aplicado en el grupo se aplicará a todos los nodos secundarios.
- Region - Es la clase base de todos los controles de interfaz de usuario basado JavaFX nodo, como de gráfico, la Ventana y Control.
- WebView - Este nodo gestiona el motor web y muestra su contenido.
- Leaf Node - El nodo sin nodos secundarios se conoce como el nodo hoja. Por ejemplo, rectángulo, elipse, Caja, ImageView, MediaView son ejemplos de nodos hoja.

Es obligatorio para aprobar el nodo raíz al escenario gráfico. Si el grupo se pasa como raíz, todos los nodos se recortan a la escena y cualquier alteración en el tamaño de la escena no afectará a la disposición de la escena.

Crear una aplicación JavaFX

Para crear una aplicación JavaFX, es necesario crear una instancia de la clase de aplicación y aplicar su método abstracto **start()** . En este método, vamos a escribir el código de la aplicación JavaFX.

Clase de aplicaciones

La **Application** clase del paquete **javafx.application** es el punto de entrada de la solicitud en **JavaFX**. Para crear una aplicación JavaFX, tiene que heredar esta clase abstracta y poner en práctica su método **start()** . En este método, es necesario escribir el código completo para los gráficos JavaFX

En el **main** método, usted tiene que lanzar la aplicación mediante el **launch()** método. Este método llama internamente al **start()** método de la clase de aplicación como se muestra en el siguiente programa.

```
public class JavafxSample extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        /*
         * Code for JavaFX application.
         * (Stage, scene, scene graph)
         */
    }
    public static void main(String args[]) {
        launch(args);
    }
}
```

En el **start()** método, con el fin de crear una aplicación típica de JavaFX, es necesario seguir los pasos que se indican a continuación -

- Preparar un gráfico de la escena con los nodos requeridos.
- Preparar una escena con las dimensiones requeridas y añadir el escenario gráfico (root node of the scene graph) a la misma.
- Preparar una etapa y añadir la escena al escenario y mostrar el contenido de la etapa.

Preparando el escenario gráfico

Según su aplicación, es necesario preparar un escenario gráfico con nodos necesarios. Desde el nodo raíz es el primer nodo, es necesario crear un nodo raíz. Como un nodo raíz, se puede elegir entre el **Group**, **Region** or **WebView** .

Group - Un nodo de grupo está representado por la clase llamada **Group** que pertenece al paquete **javafx.scene** , puede crear un nodo de grupo creando una instancia de esta clase, como se muestra a continuación.

```
Group root = new Group();
```

El **getChildren()** método de la **Group** clase le da un objeto de la **ObservableList** clase que contiene los nodos. Podemos recuperar este objeto y añadir nodos a ella como se muestra a continuación.

```
// Retrieving the observable list object
ObservableList list = root.getChildren();

// Setting the text object as a node
list.add(NodeObject);
```

También podemos añadir objetos **Node** al grupo, con sólo pasarlos al **Group** clase y de su constructor en el momento de creación de instancias, como se muestra a continuación.

```
Group root = new Group(NodeObject);
```

Region - Es la clase base de todos los controles de interfaz de usuario basada en nodos JavaFX, tales como -

- **Chart** - Esta clase es la clase base de todas las listas y que pertenece al paquete `javafx.scene.chart`. Esta clase tiene dos subclases, que son - `PieChart` y `XYChart`. Estos dos a su vez tienen subclases tales como `AreaChart`, `BarChart`, `BubbleChart`, etc. utilizan para dibujar diferentes tipos de gráficos plano xy en JavaFX. Puede utilizar estas clases para incrustar gráficos en su aplicación.
- **Pane** - Un panel es la clase base de todos los paneles de diseño tales como `AnchorPane`, `BorderPane`, `DialogPane`, etc. Esta clase pertenecen a un paquete que se llama como - `javafx.scene.layout`. Puede utilizar estas clases para insertar diseños predefinidos en la aplicación.
- **Control** - Es la clase de base de los controles de interfaz de usuario tales como `Accordion`, `ButtonBar`, `ChoiceBox`, `ComboBoxBase`, `HTML editor`, etc. This class belongs to the package `javafx.scene.control`. Puede utilizar estas clases para insertar varios elementos de la interfaz de la aplicación.

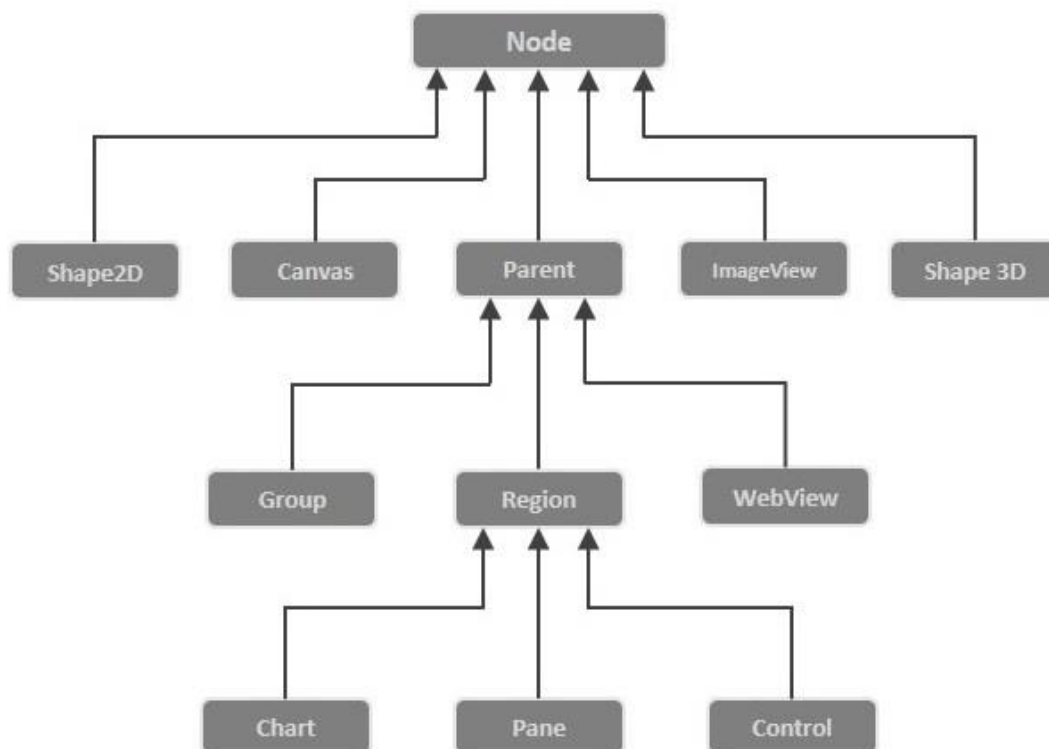
En un grupo, puede crear una instancia de cualquiera de las clases antes mencionadas y utilizarlos como nodos raíz, como se muestra en el siguiente programa.

```
// Creating a Stack Pane
StackPane pane = new StackPane();

// Adding text area to the pane
ObservableList list = pane.getChildren();
list.add(Node Object);
```

WebView - Este nodo gestiona el motor web y muestra su contenido.

A continuación se presenta un diagrama que representa la jerarquía de clases nodo de JavaFX.



Preparación de la escena

Una escena JavaFX está representada por la **Scene de clases del paquete javafx.scene** . Se puede crear una escena creando una instancia de esta clase, como se muestra en el siguiente bloque de código.

Al crear la instancia, es obligatorio pasar el objeto raíz al constructor de la clase escena.

```
Scene scene = new Scene(root) ;
```

También se puede pasar dos parámetros de tipo doble que representan la altura y la anchura de la escena tal como se muestra a continuación.

```
Scene scene = new Scene(root, 600, 300) ;
```

Preparación de la Etapa

Este es el contenedor de cualquier aplicación JavaFX y proporciona una ventana para la aplicación. Se representa por la **Stage de clases del paquete javafx.stage** . Un objeto de esta clase se pasa como un parámetro de la **start() método de la Application clase**.

El uso de este objeto, se pueden realizar varias operaciones en el escenario. En primer lugar se puede realizar lo siguiente -

- Establecer el título de la etapa usando el método **setTitle()** .
- Una el objeto de escena a la etapa de utilizar el **setScene() método**.
- Mostrar el contenido de la escena usando el **show() método tal como se muestra a continuación**.

```
// Setting the title to Stage.  
primaryStage.setTitle("Sample application") ;  
  
// Setting the scene to Stage  
primaryStage.setScene(scene) ;  
  
// Displaying the stage  
primaryStage.show() ;
```

Ciclo de vida de JavaFX Aplicación

La clase de aplicaciones JavaFX tiene tres métodos de ciclo de vida, que son:

- **start()** - El método del punto de entrada en el que el código de gráficos JavaFX se va a escribir.
- **stop()** - Un método vacío que puede ser anulado, aquí se puede escribir la lógica para detener la aplicación.

init() - Un método vacío que puede ser anulado, pero no se puede crear el escenario o escena en este método.

Además de éstos, se proporciona un método estático denominado **launch()** para iniciar la aplicación JavaFX.

Desde la **launch() método es estático, es necesario llamar desde un contexto estático (main generally)** . Cada vez que se lanza una aplicación JavaFX, las siguientes acciones serán llevadas a cabo (in the same order) .

- Se crea una instancia de la clase de aplicación.
- **Init() se llama al método**.

- El **start()** se llama al método.
- El lanzador espera a que la aplicación termine y llama a la **stop()** método.

Terminar la aplicación JavaFX

Cuando la última ventana de la aplicación está cerrada, la aplicación JavaFX se termina de forma implícita. Puede activar este comportamiento fuera pasando el valor booleano “falso” al método estático **setImplicitExit()** (**should be called from a static context**) .

Puede finalizar una aplicación JavaFX utilizando métodos explícitamente la **Platform.exit()** o **System.exit(int)**.

Ejemplo 1 - Creación de una ventana vacía

Esta sección enseña cómo crear una aplicación de ejemplo JavaFX que muestra una ventana vacía. Los siguientes son los pasos:

Paso 1: Creación de una clase

Crear una clase Java y heredar la Application de clase del paquete javafx.application e implementar el start() método de esta clase de la siguiente manera.

```
public class JavafxSample extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
    }
}
```

Paso 2: Creación de un objeto Grupo

En el start() método crea un objeto de grupo creando una instancia de la clase llamada Group, que pertenece al paquete javafx.scene , de la siguiente manera.

```
Group root = new Group();
```

Paso 3: Crear un objeto de la escena

Crear una escena creando una instancia de la clase llamada **Scene** que pertenece al paquete **javafx.scene** . Para esta clase, pasar el objeto de grupo (**root**) , creado en el paso anterior.

Además del objeto raíz, también puede pasar dos parámetros dobles que representan la altura y el ancho de la pantalla junto con el objeto de la clase de grupo de la siguiente manera.

```
Scene scene = new Scene(group, 600, 300);
```

Paso 4: Ajuste del título de la Etapa

Se puede establecer el título de la etapa con el setTitle() método de la Stage clase. El primaryStage es un objeto Stage, que se pasa al método inicio de la clase escena, como un parámetro.

Uso de la `primaryStage` objeto, establecer el título de la escena como `Sample Application` como se muestra a continuación.

```
primaryStage.setTitle("Sample Application");
```

Paso 5: Adición de escenas a la Etapa

Puede agregar un objeto de escenas a la etapa usando el método **`setScene()`** de la clase denominada **`Stage`** . Añadir el objeto Escena preparado en los pasos anteriores utilizando este método, como se muestra a continuación.

```
primaryStage.setScene(scene);
```

Paso 6: Visualización del contenido de la Etapa

Mostrar el contenido de la escena utilizando el método denominado **`show()`** de la **`Stage`** de clase de la siguiente manera.

```
primaryStage.show();
```

Paso 7: Inicio de la aplicación

Iniciar la aplicación JavaFX mediante una llamada al método estático `launch()` de la `Application` de clase desde el principal método de la siguiente manera.

```
public static void main(String args[]) {  
    launch(args);  
}
```

Ejemplo

El siguiente programa genera una ventana de JavaFX vacía. Guarde este código en un archivo con el nombre **`JavafxSample.java`**

```
import javafx.application.Application;  
import javafx.scene.Group;  
import javafx.scene.Scene;  
import javafx.scene.paint.Color;  
import javafx.stage.Stage;  
  
public class JavafxSample extends Application {  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        //creating a Group object  
        Group group = new Group();  
  
        //Creating a Scene by passing the group object, height and width  
        Scene scene = new Scene(group, 600, 300);  
  
        //setting color to the scene  
        scene.setFill(Color.BROWN);  
  
        //Setting the title to Stage.  
        primaryStage.setTitle("Sample Application");  
  
        //Adding the scene to Stage  
        primaryStage.setScene(scene);  
  
        //Displaying the contents of the stage
```

```

    primaryStage.show();
}
public static void main(String args[]) {
    launch(args);
}
}

```

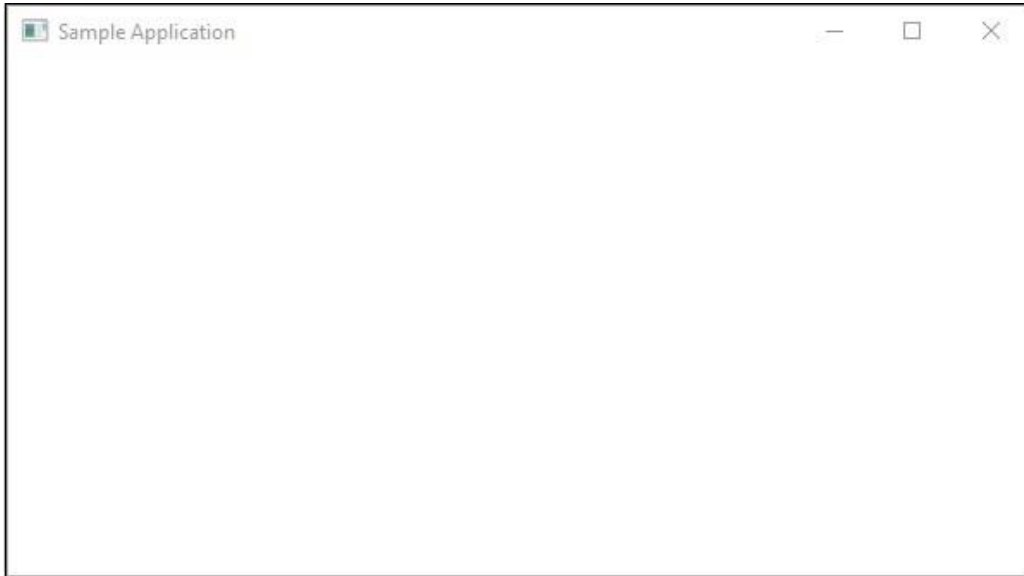
Compilar y ejecutar el archivo Java que se almacenan en el símbolo del sistema mediante los siguientes comandos.

```

javac JavafxSample.java
java JavafxSample

```

Al ejecutar, el programa anterior genera una ventana JavaFX como se muestra a continuación.



Ejemplo 2 - dibujando una línea recta

En el ejemplo anterior, hemos visto cómo crear un escenario vacío, ahora en este ejemplo vamos a tratar de dibujar una línea recta usando la biblioteca de JavaFX.

Los siguientes son los pasos:

Paso 1: Creación de una clase

Crear una clase Java y heredar la Application de clase del paquete javafx.application e implementar el start() método de esta clase de la siguiente manera.

```

public class DrawingLine extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
    }
}

```

Paso 2: Creación de una línea

Puede crear una línea en JavaFX creando una instancia de la clase llamada Line que pertenece a un paquete javafx.scene.shape, una instancia de esta clase de la siguiente manera.

```
// Creating a line object  
Line line = new Line();
```

Paso 3: Configuración de las propiedades de la Línea

Especificar las coordenadas para dibujar la línea en un plano XY estableciendo las propiedades startX, startY, endX y endY , utilizando sus respectivos métodos setter como se muestra en el siguiente bloque de código.

```
line.setStartX(100.0);  
line.setStartY(150.0);  
line.setEndX(500.0);  
line.setEndY(150.0);
```

Paso 4: Creación de un objeto Grupo

En el start() método de crear un objeto de grupo creando una instancia de la clase llamada Group, que pertenece a la javafx.scene paquete.

Pasar la línea (node) objeto, creado en el paso anterior, como un parámetro para el constructor de la clase de grupo, con el fin de añadir al grupo de la siguiente manera:

```
Group root = new Group(line);
```

Paso 5: Creación de un objeto de escena

Crear una escena creando una instancia de la clase llamada **Scene** que pertenece al paquete **javafx.scene** . Para esta clase, pasar el objeto de grupo (**root**) que se creó en el paso anterior.

Además del objeto raíz, también puede pasar dos parámetros dobles que representan la altura y el ancho de la pantalla junto con el objeto de la clase de grupo de la siguiente manera.

```
Scene scene = new Scene(group, 600, 300);
```

Paso 6: Ajuste del título de la Etapa

Se puede establecer el título de la etapa con el setTitle() método de la Stage clase. El primaryStage es un objeto Stage, que se pasa al método inicio de la clase escena, como un parámetro.

Uso de la primaryStage objeto, establecer el título de la escena como Sample Application como sigue.

```
primaryStage.setTitle("Sample Application");
```

Paso 7: Adición de escenas a la Etapa

Puede agregar un objeto de escenas a la etapa usando el método **setScene()** de la clase denominada **Stage** . Añadir el objeto Escena preparado en los pasos anteriores utilizando este método como sigue.

```
primaryStage.setScene(scene);
```

Paso 8: Visualización del contenido de la Etapa

Mostrar el contenido de la escena utilizando el método denominado **show()** de la **Stage** de clase de la siguiente manera.

```
primaryStage.show();
```

Paso 9: Inicio de la aplicación

Iniciar la aplicación JavaFX mediante una llamada al método estático **launch()** de la **Application** de clase desde el principal método de la siguiente manera.

```
public static void main(String args[]) {  
    launch(args);  
}
```

Ejemplo

El programa siguiente muestra cómo generar una línea recta utilizando JavaFX. Guarde este código en un archivo con el nombre **JavafxSample.java**.

```
import javafx.application.Application;  
import javafx.scene.Group;  
import javafx.scene.Scene;  
import javafx.scene.shape.Line;  
import javafx.stage.Stage;  
  
public class DrawingLine extends Application{  
    @Override  
    public void start(Stage stage) {  
        // Creating a line object  
        Line line = new Line();  
  
        // Setting the properties to a line  
        line.setStartX(100.0);  
        line.setStartY(150.0);  
        line.setEndX(500.0);  
        line.setEndY(150.0);  
  
        // Creating a Group  
        Group root = new Group(line);  
  
        // Creating a Scene  
        Scene scene = new Scene(root, 600, 300);  
  
        // Setting title to the scene  
        stage.setTitle("Sample application");  
  
        // Adding the scene to the stage  
        stage.setScene(scene);  
  
        // Displaying the contents of a scene  
        stage.show();  
    }  
    public static void main(String args[]) {  
        launch(args);  
    }  
}
```

Compilar y ejecutar el archivo Java que se almacenan en el símbolo del sistema mediante los siguientes comandos.

```
javac DrawingLine.java  
java DrawingLine
```

Al ejecutar el programa anterior genera una ventana de JavaFX mostrar una línea recta como se muestra a continuación.



Ejemplo 3 - Visualización de texto

También podemos incrustar texto en escena JavaFX. Este ejemplo muestra cómo insertar texto en JavaFX.

Los siguientes son los pasos -

Paso 1: Creación de una clase

Crear una clase de Java y heredar la Application de clase del paquete javafx.application e implementar el start() método de esta clase de la siguiente manera.

```
public class DrawingLine extends Application {  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
    }  
}
```

Paso 2: Inserción de texto

Puede incrustar texto en una escena JavaFX creando una instancia de la clase llamada **Text** que pertenece a un paquete **javafx.scene.shape** , una instancia de esta clase.

Para el constructor de esta clase pasar el texto a ser incrustado en el formato de cadena de la siguiente manera.

```
// Creating a Text object  
Text text = new Text(" Welcome to Tutorialspoint ");
```

Paso 3: Ajuste de la fuente

Puede establecer la fuente al texto utilizando el **setFont()** método del **Text** de clase. Este método acepta un objeto de fuente como parámetros. Establecer la fuente del texto dado a 45, como se muestra a continuación.

```
// Setting font to the text
```

```
text.setFont(new Font(45));
```

Paso 4: Ajuste de la posición del texto

Se puede establecer la posición del texto en el plano XY estableciendo la coordenadas X, Y utilizando la respectiva setter métodos **setX()** y **setY()** como sigue.

```
//setting the position of the text  
text.setX(50);  
text.setY(150);
```

Paso 5: Configuración de las propiedades de la Línea

Especificar las coordenadas para dibujar la línea en un plano XY estableciendo las propiedades startX, startY, endX y endY, utilizando sus respectivos métodos setter como se muestra en el siguiente bloque de código.

```
line.setStartX(100.0);  
line.setStartY(150.0);  
line.setEndX(500.0);  
line.setEndY(150.0);
```

Paso 6: Creación de un objeto Grupo

En el start() método, cree un objeto de grupo creando una instancia de la clase llamada Group, que pertenece al paquete javafx.scene.

Pasar el texto (node) objeto, creado en el paso anterior, como un parámetro para el constructor de la clase de grupo, con el fin de añadir al grupo de la siguiente manera:

```
Group root = new Group(text)
```

Paso 7: Creación de un objeto de escena

Crear una escena creando una instancia de la clase llamada **Scene** que pertenece al paquete **javafx.scene**. Para esta clase, pasar el objeto de grupo (**root**), creado en el paso anterior.

Además del objeto raíz, también puede pasar dos parámetros dobles que representan la altura y el ancho de la pantalla junto con el objeto de la clase de grupo de la siguiente manera.

```
Scene scene = new Scene(group, 600, 300);
```

Paso 8: Ajuste del título de la Etapa

Se puede establecer el título de la etapa con el setTitle() método de la Stage clase. El primaryStage es un objeto Stage, que se pasa al método inicio de la clase escena, como un parámetro.

Uso de la primaryStage objeto, establecer el título de la escena como Sample Application como se muestra a continuación.

```
primaryStage.setTitle("Sample Application");
```

Paso 9: Adición de escenas a la Etapa

Puede agregar un objeto de escenas a la etapa usando el método **setScene()** de la clase denominada **Stage** . Añadir el objeto Escena preparado en los pasos anteriores utilizando este método como sigue.

```
primaryStage.setScene(scene) ;
```

Paso 10: Visualización del contenido de la Etapa

Mostrar el contenido de la escena utilizando el método denominado **show()** de la **Stage** de clase de la siguiente manera.

```
primaryStage.show() ;
```

Paso 11: Inicio de la aplicación

Iniciar la aplicación JavaFX mediante una llamada al método estático **launch()** de la **Application** de clase desde el principal método de la siguiente manera.

```
public static void main(String args[]) {  
    launch(args) ;  
}
```

Ejemplo

A continuación se presenta el programa para mostrar texto utilizando JavaFX. Guarde este código en un archivo con el nombre **DisplayingText.java** .

```
import javafx.application.Application;  
import javafx.collections.ObservableList;  
import javafx.scene.Group;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
import javafx.scene.text.Font;  
import javafx.scene.text.Text;  
  
public class DisplayingText extends Application {  
    @Override  
    public void start(Stage stage) {  
        // Creating a Text object  
        Text text = new Text() ;  
  
        // Setting font to the text  
        text.setFont(new Font(45) );  
  
        // setting the position of the text  
        text.setX(50) ;  
        text.setY(150) ;  
  
        // Setting the text to be added.  
        text.setText(" Welcome to Tutorialspoint") ;  
  
        // Creating a Group object  
        Group root = new Group() ;  
  
        // Retrieving the observable list object  
        ObservableList list = root.getChildren() ;  
  
        // Setting the text object as a node to the group object  
        list.add(text) ;  
  
        // Creating a scene object
```

```

Scene scene = new Scene(root, 600, 300) ;

// Setting title to the Stage
stage.setTitle("Sample Application") ;

// Adding scene to the stage
stage.setScene(scene) ;

// Displaying the contents of the stage
stage.show() ;
}
public static void main(String args[]) {
    launch(args) ;
}
}

```

Compilar y ejecutar el archivo Java que se almacenan en el símbolo del sistema mediante los siguientes comandos.

```

javac DisplayingText.java
java DisplayingText

```

Al ejecutar el programa anterior genera una ventana de visualización de texto JavaFX como se muestra a continuación.

