**Funcionamiento del programa**

**Clase Ventana**

***Librerias:***

**Import Java.awt.Desktop:** Nos permite utilizar varias funcionalidades del escritorio.

* En nuestro la utilizamos porque necesitamos el navegador de internet por defecto.
* <https://docs.oracle.com/javase/9/docs/api/java/awt/Desktop.html>

**import java.io.IOException:** Esta librería nos indica si ocurrió algún error en las operaciones I/O(input/output).

* Nosotros lo utilizamos cuando para verificar que no ocurra algún error cuando mandamos a llamar al navegador de internet por defecto en la computadora del usuario.
* <https://docs.oracle.com/javase/8/docs/api/java/io/IOException.html>

**import java.net.MalformedURLException:** Esta librería nos indica si hay algún error con la url introducida.

* Nosotros la utilizamos para saber si pusimos alguna url de manera errónea en el navegador.
* <https://docs.oracle.com/javase/7/docs/api/java/net/MalformedURLException.html>

**import java.net.URISyntaxException:** Esta librería nos indica si hay algún error con el string que contiene la url y detecta si tiene todos los datos que debe conterner la URI de acuerdo al RFC 2396: Uniform Resource Identifiers(URI):Generic Syntax y también por el RFC 2732:Formar for Literal IPv6 Addresses in URLs.

* Nosotros lo utilizamos para saber si la URL contiene esas normas.
* <https://docs.oracle.com/javase/7/docs/api/java/net/URISyntaxException.html>
* <https://docs.oracle.com/javase/7/docs/api/java/net/URI.html>

**import java.net.URL:** Es la que nos indica que un string esta referenciando a una URL.

* Nosotros la utilizamos para indicarle al programa que introducimos una url.
* <https://docs.oracle.com/javase/7/docs/api/java/net/URL.html>

**import java.util.logging.Level:** Nos dice con un mensaje si ocurrió algún problema en el sistema, dependiendo el nivel.

* En nuestro caso nos dice su ocurrió un problema SEVERE (severo).
* <https://docs.oracle.com/javase/7/docs/api/java/util/logging/Level.html>

**import java.util.logging.Logger:** Nos muestra mensajes que nos permiten ver el movimiento del sistema.

* Nosotros lo utilizamos para que nos muestre el problema que ocurrió en el nivel SEVERE.
* <https://docs.oracle.com/javase/7/docs/api/java/util/logging/Logger.html>

***Variables:***

**char[] aCorregir:** Esta variable la utilizamos para sustituir los espacios “ “ por “+” y que pueda buscar la localización introducida de acuerdo con los links que analizamos de Google.

***Objetos:***

**Mapa browser = new Mapa():** Creamos un objeto a partir de la clase MAPA que le dará la imagen al panel para mostrar el mapa.

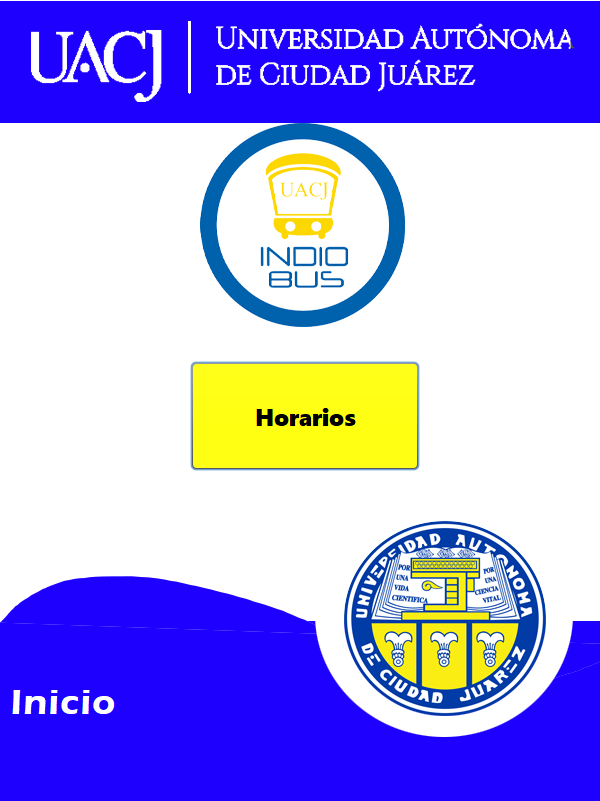
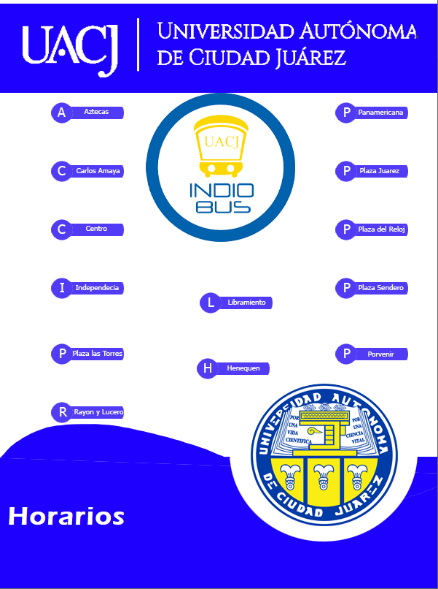
***Métodos:***

**Btnactivo(boolean falso):** Este método nos permite Activar el funcionamiento de los botones y también que se muestren.

Este método esta privado porque no queremos que otras clases accedan y activen o desactiven los botones.

Ejemplo:

Si presionamos el botón de Horarios activara todos los botones de los horarios.



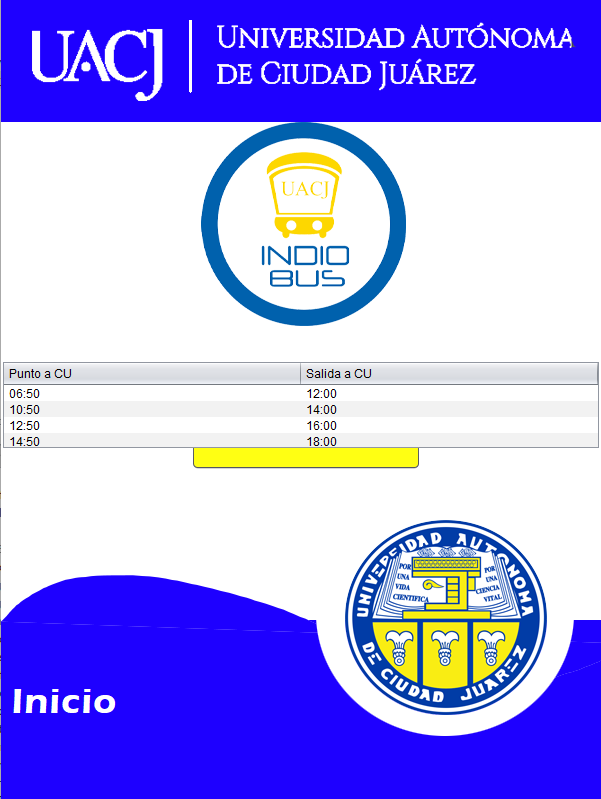
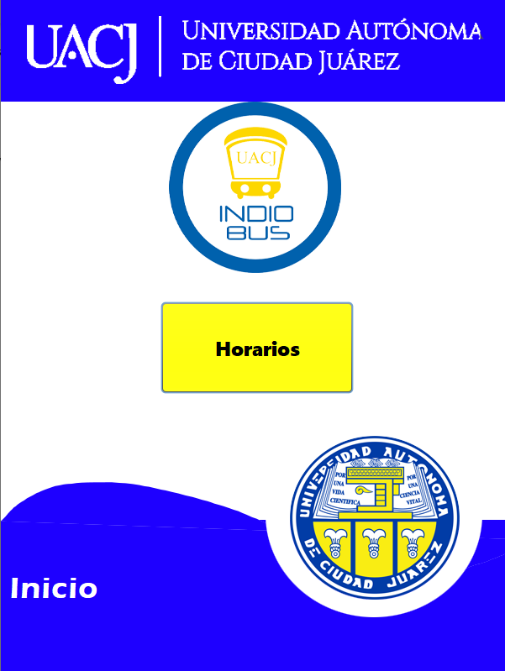
**Tblactivo(boolean falso):** Este método nos permite Activar el funcionamientos de las tablas y también las muestra.

Este método esta privado porque no queremos que otras clases accedan y activen o desactiven las tablas.

Ejemplo:

Al momento de inicializar el programa, las tablas están allí, pero están desactivadas y no se muestra.

Así se vería el programa si al momento de inicializarlo no tuviera ocultas las tablas y estuvieran activas.

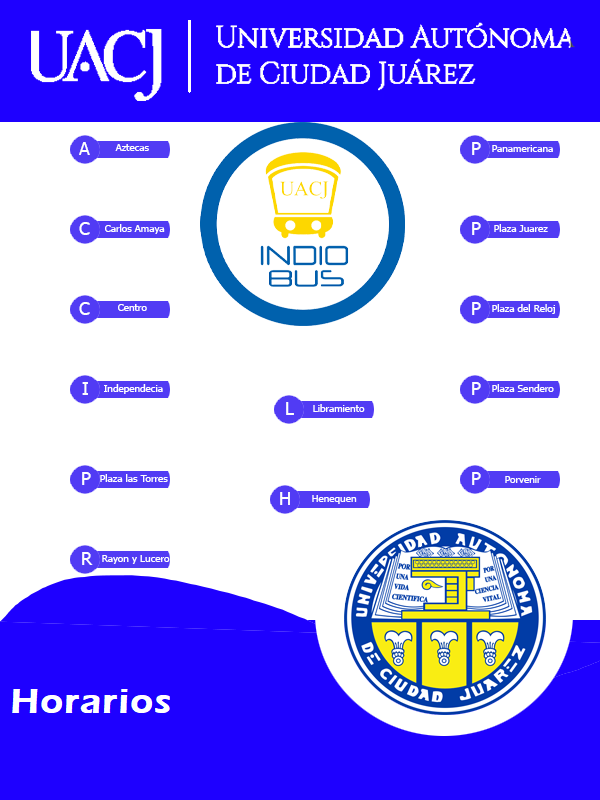
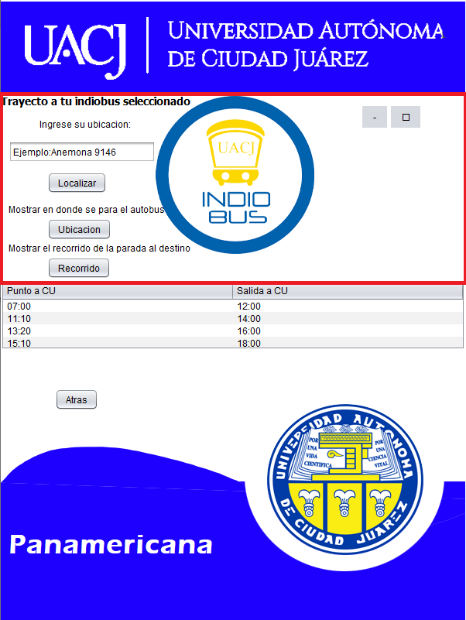


**Rastreo(boolean falso):** Este método nos mostrara botones, labels y textboxs que están ligados con la ventana de rastreo.

Este método esta privado porque no queremos que otras clases accedan y activen o desactiven lo que contiene el método.

Ejemplo:

Al momento de inicializar el programa, todo el contenido se encuentra allí pero no es visible y no está funcionando hasta que se les mande a hablar.



**Corrector(String Corregir):** Este método nos permite cambiar el contenido de una textbox y nos devolvería un string con los espacios sustituidos por “+”.

Ejemplo:

Introducimos el string “Anemona 9434” y el programa nos convertiría ese string en “Anemona+9434” pero eso lo hacemos de forma oculta por que el usuario no necesita ver eso.

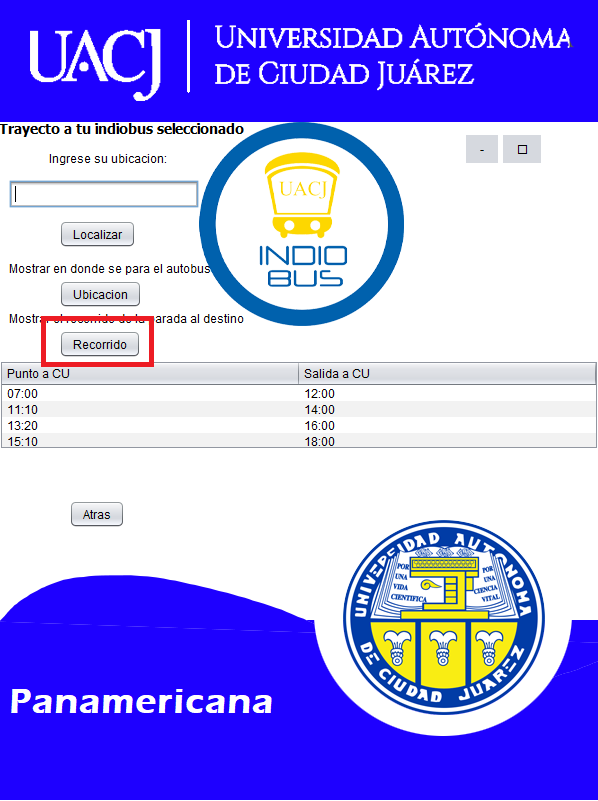
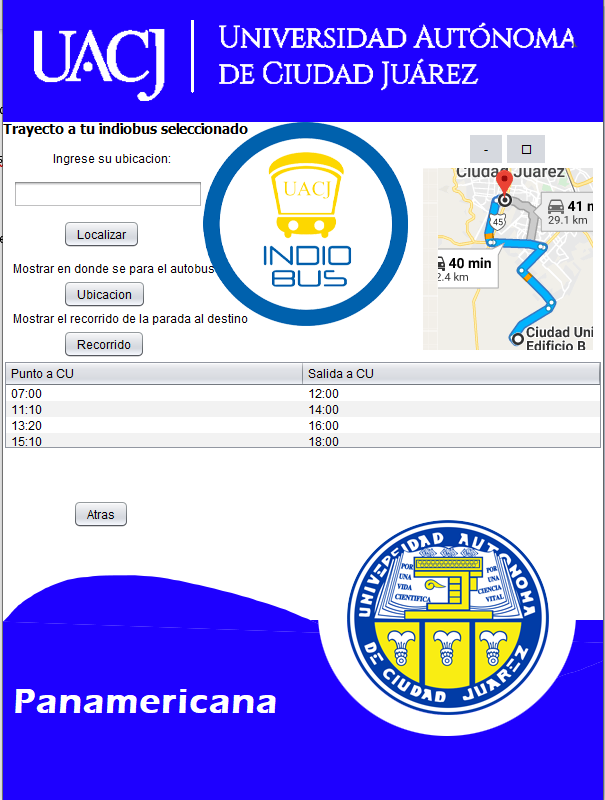


Hicimos sobrecarga porque tenemos 2 botones que funcionan igual pero uno tiene que tener mas argumentos.

**Navegador(String url):** Este método nos permite llamar el sitio web de Google maps y mostrarlo dentro de un panel para poder interactuar con él, pero solo recibe la url.

Ejemplo:

Al iniciar entrar a la ventana de un horario no nos muestra el sitio web, pero una vez presionamos el botón de recorrido este nos manda a la dirección web y nos lo muestra.



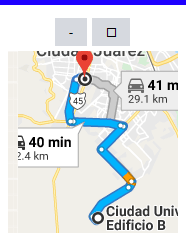
**Navegador(String url, String coordenadasHogar, String coordenadasDestino):** Es lo mismo que el otro navegador, solo que en este necesitamos mas argumentos para poder realizar la función, en este caso ocupamos una url base, las coordenadas asignadas por la persona y las coordenadas del indio bus al que quieres llegar.

**Navegador2(String url):** Este método tiene la función de abrir el navegador con el string que se le entrego para poder visualizar en primera persona la parada del indiobus, ya que el motor que utilizamos para poder correr las paginas web no soporta tantas imágenes 360.

**MiniMapa(boolean falso):** Este método nos muestra y habilita el minimapa con sus botones.

Ejemplo:

Al presionar el botón de un horario, nos va a mostrar el jpanel y sus botones con los que interactúa.



**BtnRetorno(boolean falso):** Este método simplemente habilita y oculta el botón de retorno.

**Conjunto():** Este método simplemente junta algunos métodos para no tener que repetirlos tantas veces.

* Btnactvio()
* BtnRetorno()
* Rastreo()
* MiniMapa()

***Funcionamiento:***

**Ventana de Inicio**

initComponents(): El código empieza inicializando todos los componentes.

this.setTitle("Mapa"): Aquí le asignamos el nombre a la ventana.

this.setResizable(false): Le decimos que la ventana no pueda maximizarse.

this.setLocationRelativeTo(null): Aquí le decimos que la ventana se abra en el centro de la pantalla.

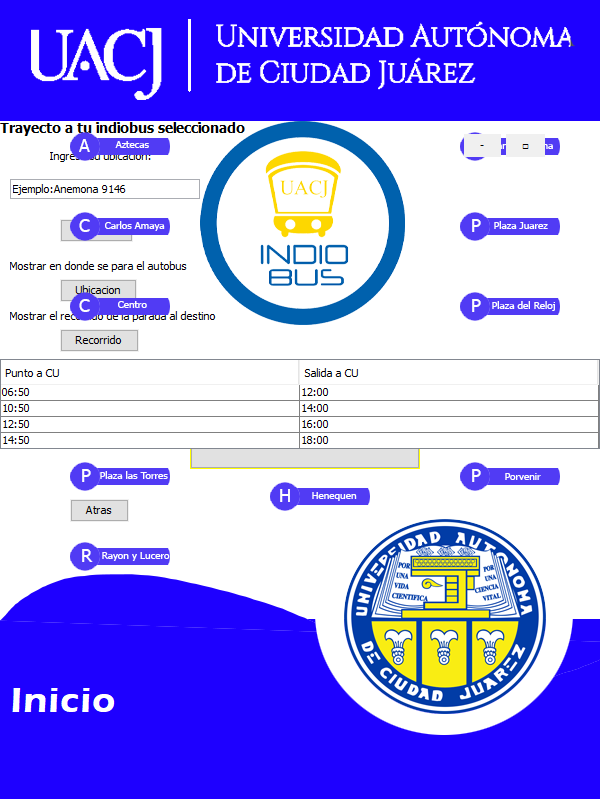
Btnactivo(false): Desactivamos todos los botones.

Tblactivo(false): Desactivamos todas las tablas.

Rastreo(false): Desactivamos todos los elementos de rastreo.

MiniMap(false): Desactivamos el minimapa.

Por lo que en ves de verse así:



Se ve así:

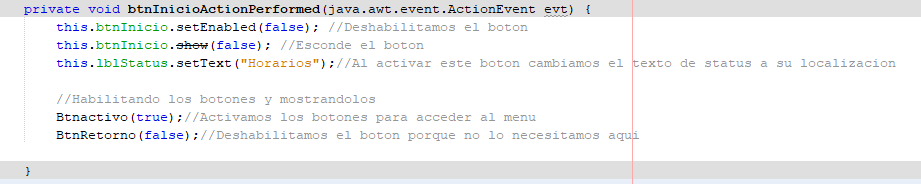


**Ventana de horarios**

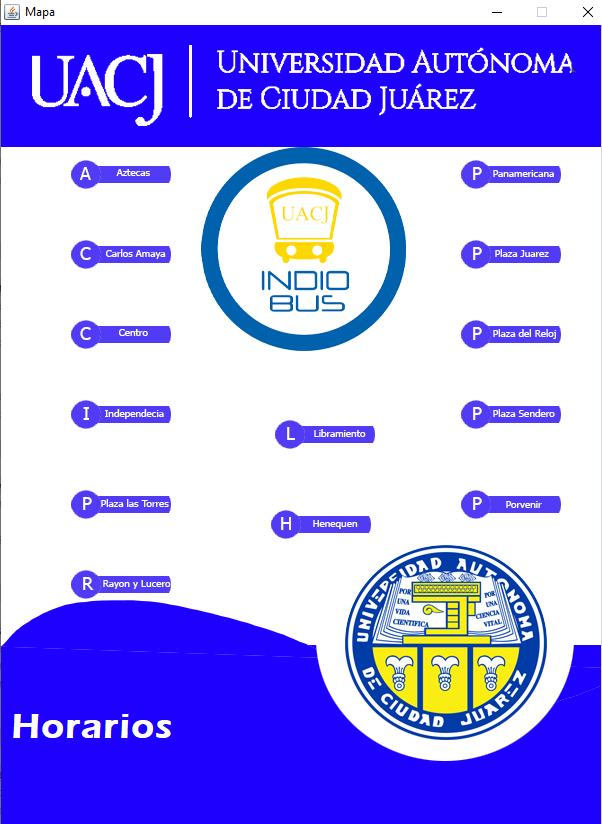
Después le agregamos una acción al botón

Que básicamente cuando le damos click a ese botón y pasa lo siguiente:

1. Deshabilitamos y ocultamos el botón de Horarios
2. Le cambiamos el texto al labelStatus por Horarios (Este nos diría en que ventana estamos).
3. Activamos todos los botones que contendrán los horarios.
4. Deshabilitamos y ocultamos el botón de retorno.



Por lo que al momento de dar click a horarios cambiaria la ventana a esta forma.

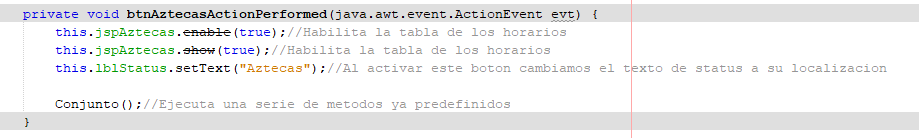


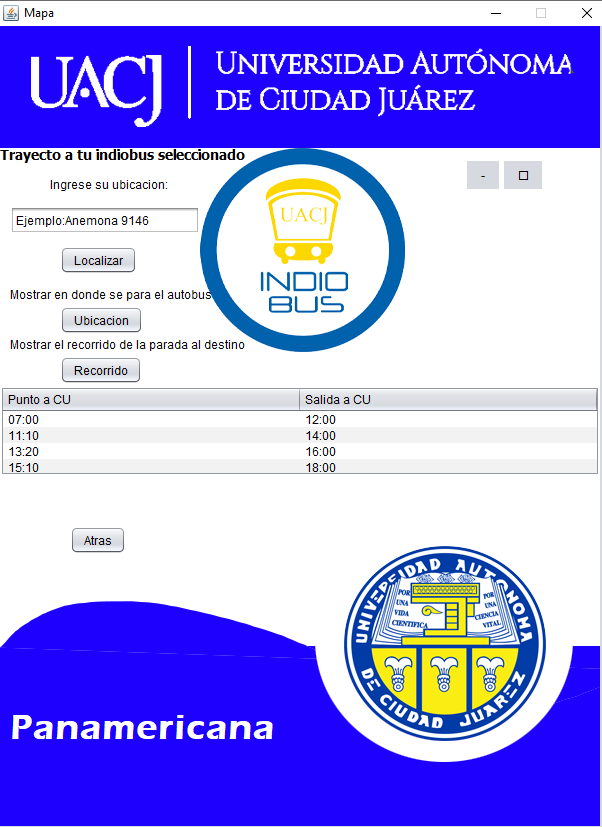
**Ventana de cada Horario**

Cada botón contiene un horario, cada uno tiene unos atributos muy parecidos por lo que solo mostraremos 1.

Le agregamos una acción a los botones para que funcione cuando le demos click.

1. Habilita y Muestra su respectiva tabla.
2. Cambia el texto de labelStatus por el nombre de ese horario.
3. Después ejecutan el método de Conjunto()
   * Deshabilita y oculta los botones
   * Activa el botón de retorno.
   * Activa todos los objetos de rastreo.
   * Activa todos los objectos del minimapa.





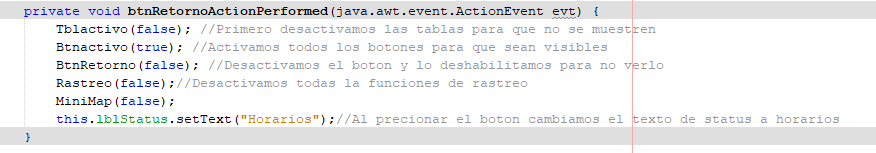
**Funcionamiento de la ventana de un horario**

*Botón de Retorno*



Al hacer click hace las siguientes acciones:

1. Desactiva y oculta la tabla.
2. Activa todos los botones.
3. Se desactiva y oculta el botón de retorno.
4. Desactiva y oculta todos los componentes de rastreo.
5. Desactiva y oculta todos los componentes de minimapa.
6. Cambia el texto del labelStatus a Horarios.



*Botón de Recorrido*



Al hacer click hace las siguientes acciones:

1. Lee el texto del labelStatus.
2. Inserta el texto en un switch.
3. Dependiendo lo que diga el texto este le enviara una url al método de navegador que solo tiene un argumento y abrirá la página web en el programa.



*Botón Ubicación*



Hace exactamente lo mismo que el botón de recorrido, solo que este abre el navegador de internet en vez de utilizar el motor, porque el motor no puede leer imágenes 360.



*Botón localización*



Al hacer click en este botón se realizan las siguientes acciones:

1. Ejecuta el método Corrector() y le inserta el string que la persona lleno en el textbox.
2. El método Corrector() intercambia los espacios por “+”.
3. Después inserta el texto del labelStatus en un switch.
4. Dependiendo de los casos se insertan los argumentos necesarios para poder llamar a una página web.
5. 
6. Finalmente muestra el resultado en el panel que mostrara el mapa de la página web.

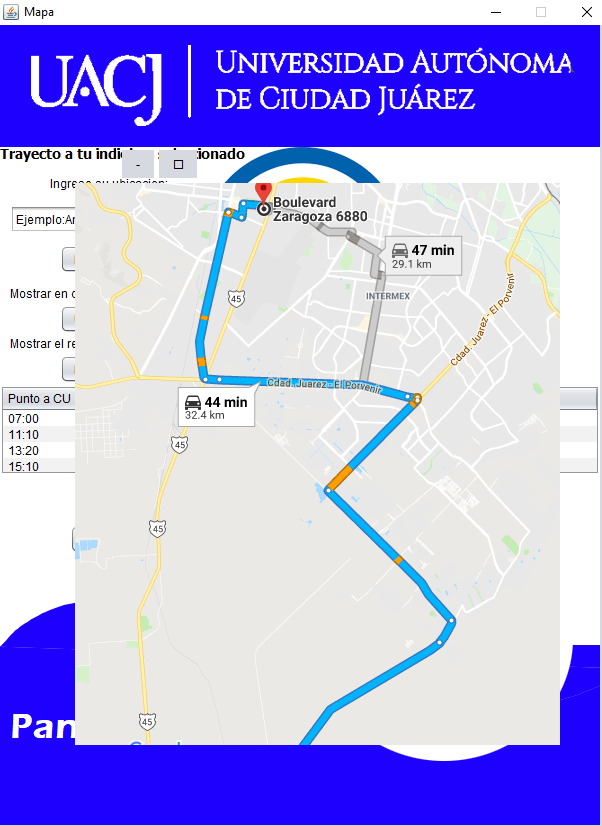


*Botones minimizar y maximizar*



Estos botones son los encargados de minimizar y maximizar el panel que muestra el mapa en el programa.

1. Cambia la localización del panel.
2. Cambia el tamaño del panel.



**Clase Mapa**

**import java.net.MalformedURLException:** Esta librería nos indica si hay algún error con la url introducida.

* Nosotros la utilizamos para saber si pusimos alguna url de manera errónea en el navegador.
* <https://docs.oracle.com/javase/7/docs/api/java/net/MalformedURLException.html>

**import java.net.URL:** Es la que nos indica que un string está referenciando a una URL.

* Nosotros la utilizamos para indicarle al programa que introducimos una url.
* <https://docs.oracle.com/javase/7/docs/api/java/net/URL.html>

**import javafx.embed.swing.JFXPanel:** Esta librería nos indica que vamos a crear un panel jfx

* Nosotros la utilizamos porque es el que va a mostrar el mapa.
* <https://openjfx.io/javadoc/11/javafx.swing/javafx/embed/swing/JFXPanel.html>

**import javafx.scene.Scene:** Esta librería es la que se encarga de crear la scena el el panel jfx para poder mostrar el contenido.

* Nosotros la utilizamos para poder utilizar el mapa.
* <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Scene.html>

**import javafx.scene.web.WebEngine:** Es el motor que utiliza java para poder utilizar el navegador en el.

* Nosotros la utilizamos para poder navegar en Google maps.
* <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/web/WebEngine.html>

**import javafx.scene.web.WebView:** Este es el que muestra la imagen que se genera atravez del WebEngine.

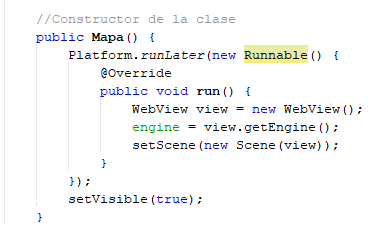
* Se utiliza para mostrar el mapa del engine.
* <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/web/WebView.html>

Variables

**Private WebEngine engine**: Se crea una variable de tipo webengine que es la encargada de utilizar el motor que mostrara el “Navegador” en java.

Métodos y constructores

Se crea un constructor de la clase mapa



**Platform.runLater(new Runnable()):** Se utiliza para realizar rápidas y simples operaciones.

* En nuestro caso lo genero automáticamente netbeans.

**Public void run():** Se ejecuta cuando se lee el constructor y es el que se encarga de hechar a andar el motor para poder mostrar la pagina web en java.

**WebView view = new WebView():** Se crea un objecto de tipo webview.

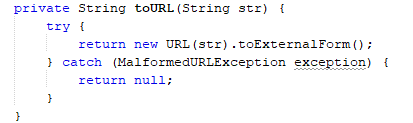
* Este es el encargado de darle la imagen al objeto que tendrá el navegador.

**engine = view.getEngine():** Es el que nos dice que se va a ejecutar el engine.

**setScene(new Scene(view)):** Este es el que nos dice que se creo una escena ( seria como una ventana del navegador).

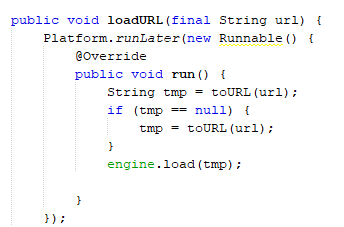
**setVisible(true):** Nos dice que va a estar visible en el momento que se inicializa la clase.

Se crea un método que recibe un string y nos regresa una url o sino nos regresa un nulo.



El try lo que hace es que intenta regresar un Url que contiene un string y se lo entrega a una forma externa y si no logra entregar la url esta regresa un nulo porque el url es invalido.

Se crea un método que cargara la url de la pagina que ocupara un string de url final porque solo queremos utilizar una vez la url y nada mas.



**String tmp = toURL(url):** se declara una variable de tipo string que contendrá la url.

Si tmp es igual a null entonces se crea una url en blanco.

**engine.load(tmp):** es el que carga la url en el engine del navegador.