



Actividad | 2 | Programa Banco

Mexicano (parte 1)

Nombre del curso

Ingeniería en Desarrollo de Software



TUTOR: Aarón Iván Salazar Macías.

ALUMNO: Uziel de Jesús López Ornelas.

FECHA: 30 de julio del 2025.

Tabla de Contenido

Introducción	1
Descripción	1
Justificación	3
Desarrollo	3
Interfaz.....	3
Codificación.....	6
Conclusión	13
Referencias	14

Introducción

Se han obtenido maneras en las que se ejecutan las instrucciones por medio de una sintaxis correcta, ¿esto qué quiere decir?, pues bien, una sintaxis es parte de un lenguaje de programación que define el conjunto de reglas que deben seguirse para escribir el código fuente de los programas para considerarse como correcto, datos como los siguientes son importantes:

- Todos los archivos pertenecen a un paquete.
- Importar archivos para un proyecto.
- Java usa clases para ejecutar el programa.
- Tipo de datos.
- Modificadores de acceso.
- El método principal.

Como ya se mencionó con anterioridad manejar la sintaxis correcta ayuda a que este se ejecute sin ningún problema. Se incluyen los elementos básicos con los que se relacionan como lo son:

- Sensibilidad a mayúsculas y minúsculas.
- Nombres de métodos.
- Nombre del archivo del programa.
- Comentarios de bloque.
- Comentarios de documentación.
- Comentarios de línea.
- Bloques de código.
- Sentencias de expresión.
- Sentencias de declaración de variables.
- Sentencias de paquete.
- Sentencias de importación.

Descripción

Conforme a lo explicado anteriormente nos damos cuenta de que como todos los IDE estos poseen muchas herramientas y elementos que ayudan a su funcionamiento como a la implementación de comandos para realizar tareas. Componentes existen de diferentes formas, pero, la más importante o los más

importantes son aquellos que están unidas a una base y que son prácticamente indispensables, como los siguientes:

- **Identificadores:** Secuencia de caracteres, estos comienzan con una letra, y contiene letras y números.
- **Variables:** Localidades de memoria en las que pueden almacenar datos.
 - **Variables de instancia:** Definir los atributos de un objeto.
 - **Variables de clase:** Sus valores son los mismos para todas las instancias de la clase.
 - **Variables locales:** Se declaran y se utilizan dentro de las definiciones de los métodos.
- **Operadores:** Son signos especiales, son el mecanismo por el cual los objetos interactúan.
 - **Operadores Java aritméticos:**
 - Suma (+).
 - Resta (-).
 - Multiplicación (*).
 - División (/).
 - Resto de la división (%).
 - **Operadores Java relacionales:**
 - Menor que (<).
 - Mayor que (>).
 - Menor o igual (<=).
 - Mayor o igual (>=).
 - Distinto (!=).
 - Igual (==).
 - **Operadores Java lógicos:**
 - AND (&&).
 - OR (||).
 - NOT (!).

- **Operadores Java unitarios:**
 - Signos negativo o positivo (-+).
 - Incremento y decremento (++-).
 - Complemento a 1 (~).

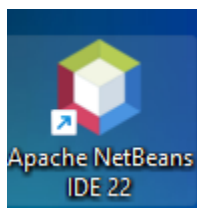
Justificación

¿Por qué es importante establecer dichos conceptos en la actividad?, pues es muy sencillo, al igual que con la anterior tenemos que utilizar elementos adecuados para que funcione correctamente, las instrucciones que le demos a nuestro programa, si existe una tarea en la que se tienen que sumar dos valores entonces la sintaxis o el elemento a necesitar sería (+), al igual si se tiene que hacer diversas operaciones aritméticas pues serán aquellos en las que se necesite utilizar para resolver alguna problemática. Si no conocemos la organización y orden correcto, entonces básicamente no podríamos realizar ningún problema. Por ello la importancia de conocer no solo uno sino diferentes lenguajes de programación ya que como sabemos estos, tienen similitudes como diferencias, pero lo más importante es saber utilizarlos para enriquecer nuestro conocimiento y experiencia en el uso de estas herramientas. El conocer no solo varios lenguajes, sino que adaptarlos en nuestro centro de trabajo es fundamental para nuestro desarrollo profesional como personal.

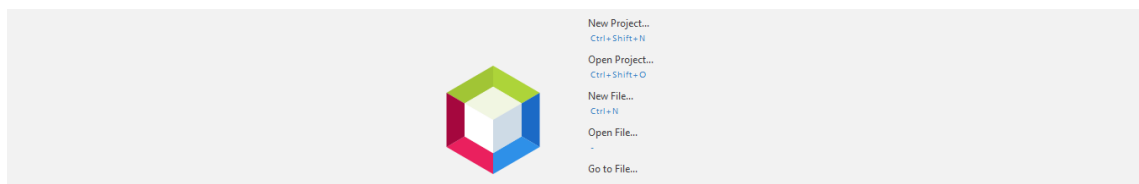
Desarrollo

Interfaz

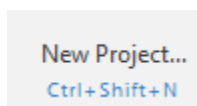
Para iniciar con esta etapa primero abriremos la aplicación de “NetBeans” en nuestro ordenador:



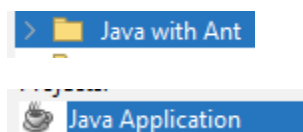
Ingresamos y esperamos a que cargue toda la configuración necesaria en donde se nos abrirá la ventana siguiente:



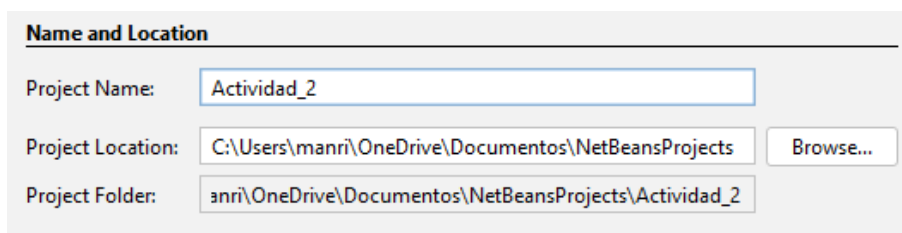
Para iniciar con nuestro proyecto seleccionaremos el siguiente apartado:



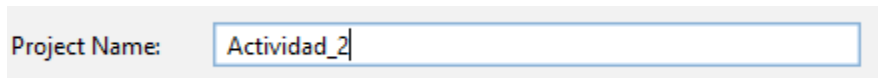
Con esto se nos abrirá una nueva ventana donde seleccionaremos apartados importantes para nuestro proyecto, como los siguientes:



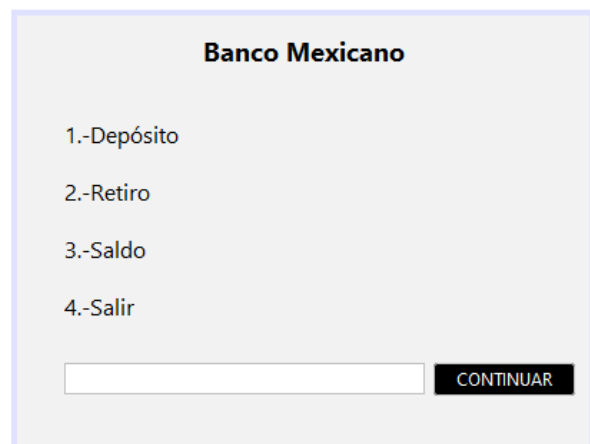
Después de ello daremos en siguiente para continuar, nos muestra una ventana en donde podemos cambiar el nombre, así como las rutas en las que se guardaría nuestros proyecto



Con esto cambiaremos el nombre de nuestro proyecto:



En la siguiente imagen tenemos el diseño de la interfaz del menú principal de nuestra aplicación:

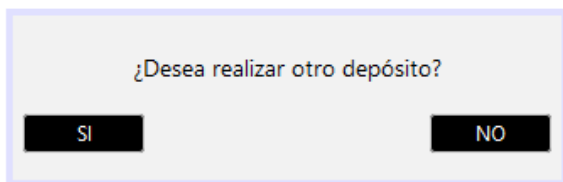


En la siguiente imagen tenemos la interfaz del apartado de “Depósito”:



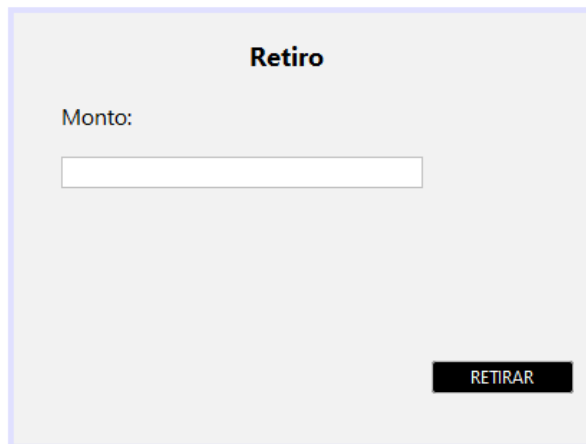
The image shows a light gray rectangular box representing a user interface for deposits. At the top center, the word **Depósito** is written in bold black text. Below it, the text 'Cantidad a depositar:' is followed by a white rectangular input field. In the bottom right corner of the box, there is a black button with the word **DEPOSITAR** in white capital letters.

En la siguiente imagen tenemos la interfaz para decidir si es usuario quiere volver a depositar o que lo lleve al menú principal:



The image shows a light gray rectangular box. At the top center, the text '¿Desea realizar otro depósito?' is displayed. Below this text, there are two black buttons with white text: 'SI' on the left and 'NO' on the right.

La imagen para la interfaz de “Retiro”:



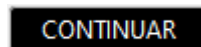
The image shows a light gray rectangular box representing a user interface for withdrawals. At the top center, the word **Retiro** is written in bold black text. Below it, the text 'Monto:' is followed by a white rectangular input field. In the bottom right corner of the box, there is a black button with the word **RETIRAR** in white capital letters.

La siguiente imagen representa el “Saldo”:



Codificación

Crearemos el código para darle instrucciones al botón de continuar que está en el apartado de menú principal:



Ahora colocaremos las instrucciones para darle vida a nuestra interfaz, en donde principalmente le estamos ordenando de acuerdo a lo que indica las opciones del menú, tenemos que escribir solo el número correspondiente para que este nos dirija a la ventana siguiente, caso contrario si no se escribe lo que se indica saltara una ventana de texto que mande un mensaje de error y que en automático se elimine lo que se escribe en la caja de texto:

```
String opcion = jtf_Menu.getText();

switch (opcion) {
    case "1":
        Deposito_Frame deposito = new Deposito_Frame();
        deposito.show();
        break;
    case "2":
        Retiro_Frame retiro = new Retiro_Frame();
        retiro.show();
        break;
    case "3":
        Saldo_Frame saldo = new Saldo_Frame();
        saldo.show();
        break;
    case "4":
        this.dispose();
        break;
    default:
        JOptionPane.showMessageDialog(null, "Opción no valida, favor de intentar nuevamente.");
}

jtf_Menu.setText("");
}
```


Es momento de programar el código para que cuando se deposite una cantidad este nos mande un mensaje de si queremos volver a depositar o regresar al menú principal; con esto al momento de seleccionar el botón de depositar este nos mandará un mensaje:

```
private void jButton_DepositarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Deposito_Opcion_Frame deposito_opcion = new Deposito_Opcion_Frame();
    JOptionPane.showMessageDialog(null, "Su depósito se ha realizado con éxito.");
    deposito_opcion.show();
    this.hide();

    jTextField_Deposito.setText("");
}
```

Programaremos el código para la opción de los depósitos, este es con el botón de “SI”:

```
private void jButton_SI_depositoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.dispose();
    Deposito_Frame back = new Deposito_Frame();
    back.show();
}
```

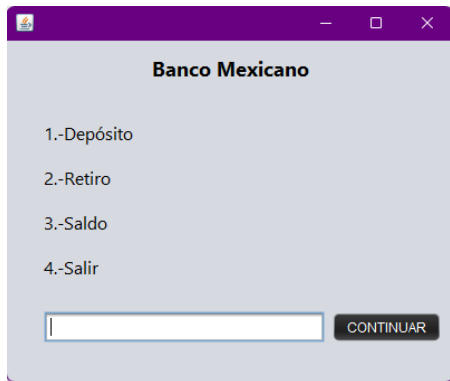
Vayamos con el botón “NO”:

```
private void jButton_NO_depositoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.dispose();
    Menu_Principal_Frame back = new Menu_Principal_Frame();
    back.show();
}
```

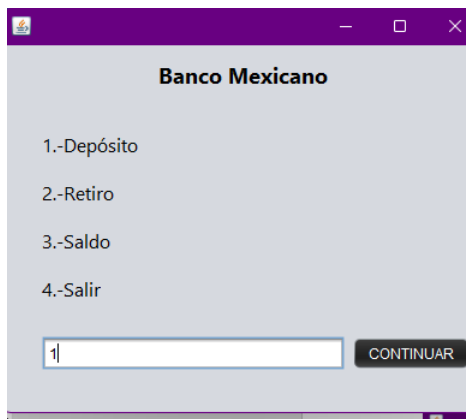
Vamos a programar el botón de “Retirar” para que nos mande al menú principal:

```
private void jButton_RetirarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.hide();
    Menu_Principal_Frame back = new Menu_Principal_Frame();
    JOptionPane.showMessageDialog(null, "Su Retiro ha sido exitoso.");
    back.show();
}
```

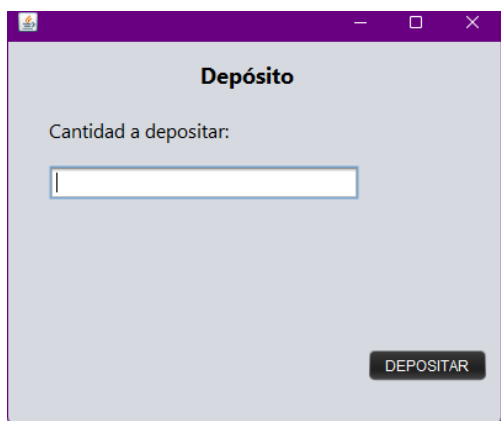
Mostraremos el funcionamiento de la interfaz de nuestra aplicación para el banco, primero mostraremos el menú principal ya con la aplicación ejecutándose:



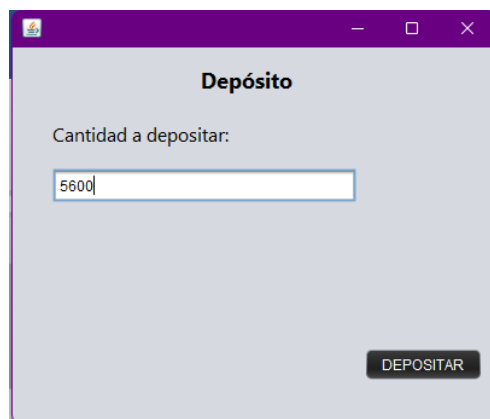
Seleccionaremos el número 1 para que nos mande a la opción de “Deposito”:



Al darle continuar este nos manda a la ventana de “Deposito”:

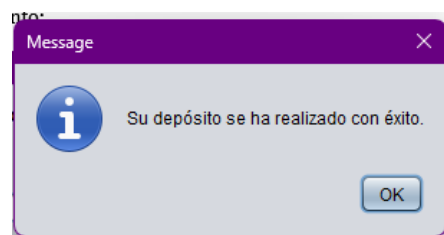


Si colocamos un monto:

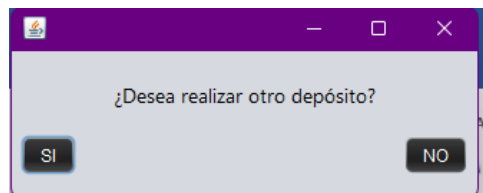


A screenshot of a software window titled "Depósito". It has a light gray background and a dark gray title bar with standard window controls. The text "Cantidad a depositar:" is followed by a text input field containing the number "5600". At the bottom right, there is a dark gray button labeled "DEPOSITAR".

Y seleccionamos en "Depositar" nos mandara otra ventana de advertencia:

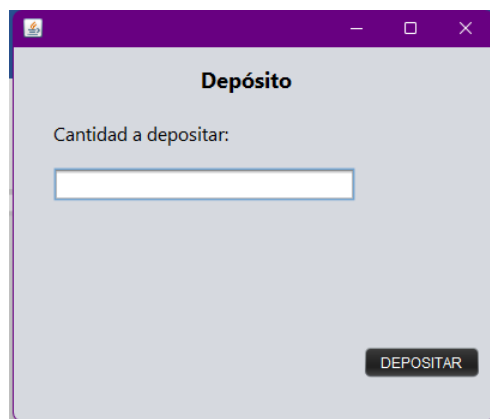


Al darle en "OK" este nos mandara a la opción si queremos volver a depositar:



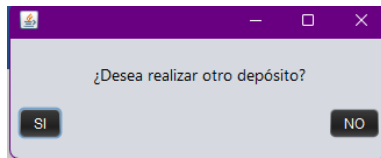
A screenshot of a confirmation dialog box with the text "¿Desea realizar otro depósito?". It has two buttons at the bottom: "SI" (Yes) on the left and "NO" (No) on the right.

Si seleccionamos que "Si" este nos mandara de vuelta a la ventana de "Deposito":

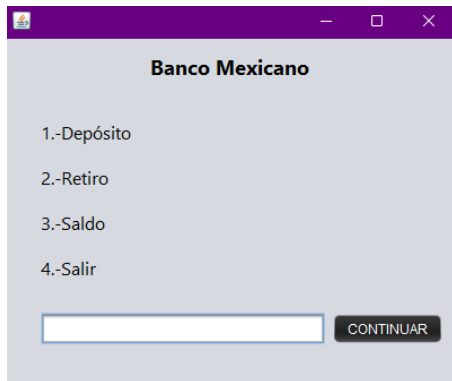


A screenshot of the "Depósito" window, identical in layout to the first one. The text input field for "Cantidad a depositar:" is now empty. The "DEPOSITAR" button remains at the bottom right.

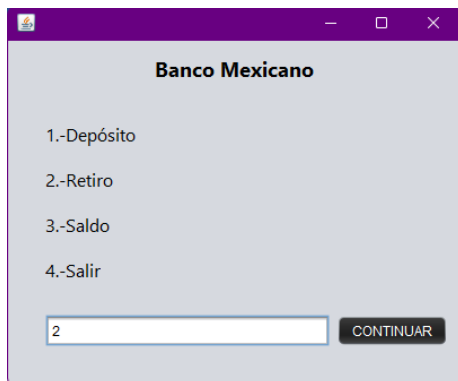
Si volvemos a la ventana de opción y seleccionamos “No”:



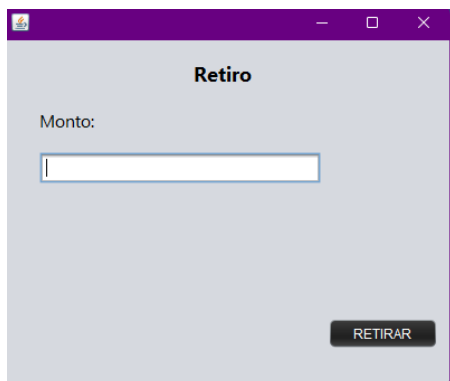
Este nos mandara de vuelta al menú principal:



Una vez en el menú principal, vamos a seleccionar la opción “2”:



Nos mandara a la ventana de “Retiro”:



Al colocar la cantidad y seleccionar el botón de “Retirar” este nos mandara un mensaje:

The screenshot shows a window titled "Retiro" with a label "Monto:" and a text input field containing the value "5600". At the bottom right, there is a button labeled "RETIRAR".

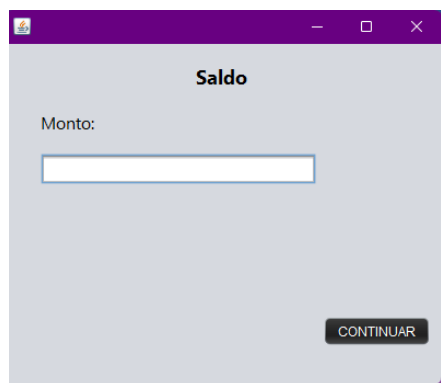
The screenshot shows a "Message" dialog box with a blue information icon and the text "Su Retiro ha sido exitoso." (Your withdrawal was successful). There is an "OK" button at the bottom right.

Nos devolverá al menú principal:

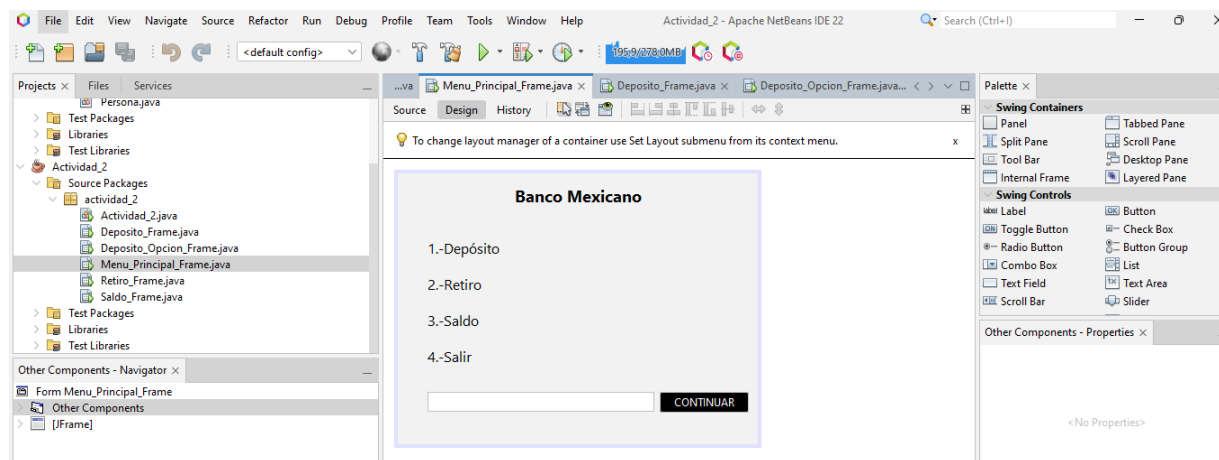
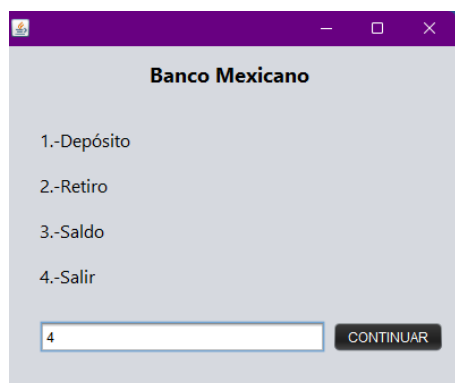
The screenshot shows the main menu of "Banco Mexicano". It lists four options: "1.-Depósito", "2.-Retiro", "3.-Saldo", and "4.-Salir". Below the list is a text input field and a "CONTINUAR" button.

Si colocamos la opción “3”, este nos mandara a la ventana de “Saldo”:

The screenshot shows the same "Banco Mexicano" main menu. The text input field now contains the number "3", and the "CONTINUAR" button is visible.



La última opción nos cierra el programa por completo:



Crearemos una nueva clase en donde colocaremos la conexión a nuestra base de datos, será sencilla:

Name and Location

Class Name:

Esta sería la codificación sencilla en la que estaríamos conectando la base de datos a nuestra aplicación de Java:

```
import java.sql.DriverManager;
import java.sql.Connection;

/**
 *
 * @author manri
 */
public class Conexion_BD {

    public Connection getConnection(){
        Connection Con = null;
        String base = "Cajero_BD";
        String url = "jdbc:sql://ROG-ONE\\SQLEXPRESS/" + base;
        String user = "root";
        String password = "";

        try {
            Class.forName("com.sql.jdbc.Driver");
            Con = (Connection) DriverManager.getConnection(url, user, password);
        } catch (Exception e) {
            System.err.print (e);
        }

        return Con;
    }
}
```

Conclusión

En esta actividad me he dado cuenta de cómo podemos crear nuestra propia aplicación con pasos específicos para tener desde los botones, el diseño y la codificación correcta hasta una interfaz funcional. En la primera actividad se logró crear una interfaz que tenía como objetivo resolver un problema; se trataba de calcular el IMC de una persona y dependiendo del número o etapa en la que se encontraba este mandaba un mensaje representando el IMC, sin lugar a dudas este era funcional en su totalidad, solo que teníamos únicamente una ventana, no existía una interacción o forma de acceder a otra, y, aquí es en donde entra la actividad número 2 ya que no solo se codifica, sino que se navega entre diferentes ventanas de acuerdo con las instrucciones que se tengan. El caso de colocar un número y que este nos mande o re dirija a otra ventana demuestra la capacidad y opciones que se pueden realizar, por que como sabemos existen muchas aplicaciones en las cuales van dirigidas a un usuario final, claro está que una aplicación o software no se va a liberar sin una revisión previa, existe puestos encargados para revisar las fallas o errores que puedan ser perjudiciales y que como en todo caso se corrigen para entregar un producto en las mejores condiciones posibles.

Link de GitHub

<https://github.com/UZLOP984/Lenguajes-de-Programaci-n-IV.git>

Referencias

Los elementos del lenguaje Java. (s. f.).

<http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/introduccion/primer0.htm>

Elementos involucrados en un programa en Java – Alianza B@UNAM, CCH & ENP ante la pandemia. (s. f.). <https://alianza.bunam.unam.mx/cch/elementos-involucrados-en-un-programa-en-java/>