Laboratorio
Sintaxis básica, estructuras de control y tipos de datos del lenguaje Java

---

a. Objetivo General

Construir una aplicación sencilla, donde el estudiante se **apropie** de la sintaxis básica de Java y conozca los tipos de datos primitivos del lenguaje. Se utilizarán los tipos de datos **double, String, char, boolean y arreglos estáticos.**

b. Objetivos Específicos:

- Declarar y utilizar variables de tipo double, String, char y boolean.
- Declarar y utilizar variables de tipo arreglo.
- Sintaxis básica del lenguaje Java
- Identificar algunas estructuras de control básicas el Java (if, if doble, while)
- Conocer los tipos de datos disponibles en el lenguaje Java
- Compilar y ejecutar una aplicación Java desde BlueJ.

c. Contexto:

Se requiere desarrollar una pequeña aplicación que se encargue de manejar las cuentas de un sistema bancario. Este debe ser capaz de registrar datos de clientes (cédula, nombre y apellidos), abrir cuentas con un monto inicial mínimo de 1000 colones, realizar transacciones (depósitos, retiros y consulta de saldo), consultar clientes por medio del número de cédula o por nombre, y finalmente consultar cada una de las cuentas abiertas. Al abrir la cuenta, debe registrarse el número de la cuenta como String, el cual deberá estar compuesto del primer carácter de la cédula del cliente más el índice de la posición en el arreglo donde se almacenó el nombre del cliente. Debe verificarse que la cuenta no haya sido abierta previamente. El número máximo de clientes será de 10 usuarios. **Es importante anotar que este contexto NO será implementado usando el paradigma de orientación a objetos, dado que su objetivo es familiarizar al estudiante con el lenguaje de programación Java**.

d. Desarrollo:

El laboratorio consiste de 2 partes. La primera parte será una guía que el estudiante deberá realizar paso a paso, en la cual aprenderá y practicará a la vez, como parte de la adquisición del conocimiento y la segunda será la implementación de un conjunto de funcionalidades adicionales como ejercicios de comprobación y complementación a este laboratorio.

PRIMERA PARTE

En esta apartado el estudiante deberá crear la clase AplCuenta, debiendo implementar las variables y métodos necesarios para su debido funcionamiento, para ello dispondrá del diagrama de clase con notación UML correspondiente y de una serie de pasos presentados seguidamente.

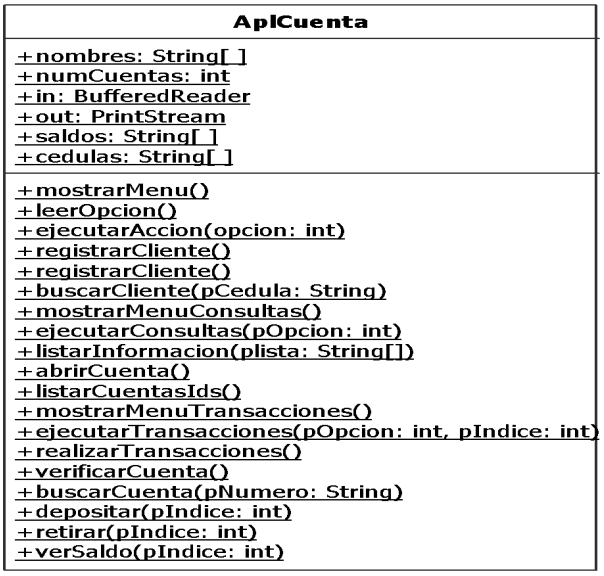

Diagrama de Clase
Notación UML

Pasos a seguir:

IMPORTANTE: NO HAGA COPY/PASTE.

1. **Crear el proyecto**

   Cree un nuevo proyecto y asígnele el nombre de la siguiente forma:

   PRYLAB2_APELLIDO_NOMBREDELESTUDIANTE, por ejemplo: **PRYLAB2_SOTO_LUIS**

   Use su apellido y nombre según corresponda.

2. **Crear Paquete llamado "aplicación" dentro del proyecto creado en el paso anterior. Para ello haga clic derecho en la pantalla y seleccione la opción crear paquete.**

3. **Crear Clase "AplCuenta" dentro del paquete "aplicacion". Para ello haga clic en el botón "Nueva Clase."**

Por convención el nombre de clases comienza con letra mayúscula, el nombre de las clases siempre DEBE ser en SINGULAR

Recordar guardar el proyecto después de cada cambio realizado

4. **Abrir el archivo AplCuenta y colocar los import que se detallan abajo: (después de la indicación del paquete)**

```
import java.io.BufferedReader;
import java.io.PrintStream;
import java.io.InputStreamReader;
```

5. **Crear dentro de la clase AplCuenta las variables necesarias.**

```
public class AplCuenta {
    static String[] nombres;
    static String[] cedulas;
    static double[] saldos;
    static String[] idCuentas;
    static int numCuentas = 0;
    // Estas son las declaraciones de los objetos de la entrada y salida estándar
    static BufferedReader in;
    static PrintStream out;
```

```
}
```

6. **Crear dentro de la clase AplCuenta el método main y asignar a cada una de las variables, lo que se indica.**

```
public static void main( String[] args )throws java.io.IOException
 {
      nombres= new String[10];
      cedulas= new String[10];
      saldos= new double[10];
      idCuentas= new String[10];
      // Se crean los objetos tanto para la entrada de datos desde el
      // teclado (in) como para la salida de datos a la consola (out)
      in = new BufferedReader( new InputStreamReader( System.in ) );
      out = System.out;
      boolean noSalir = true;
      int opcion;

 }
```

7. **Escribir, dentro del método main, después de la última declaración, el ciclo do-while.**

```
      do {
         mostrarMenu();
         opcion = leerOpcion();
         noSalir = ejecutarAccion( opcion );
      } while ( noSalir );
```

8. **Escribir, dentro de la clase AplCuenta, después del método main, la función mostrarMenu()**

```
static void mostrarMenu() {
      out.println( "1. Registrar Cliente" );
      out.println( "2. Abrir una Cuenta" );
      out.println( "3. Realizar Transacción" );
      out.println( "4. Consultar Información " );
      out.println( "5. Salir" );
}
```

9. **Escribir, dentro de la clase AplCuenta, después de mostrarMenu, la función leerOpción()**

Observar en la primera línea como se incluye la instrucción "throws java.io.IOException", la cual indica que dentro de esta función se puede lanzar una excepción de tipo IOException si se trata de asignar un dato de un tipo que no concuerda con lo esperado. Será necesario hacer throws de java.io.IOException dado que se estará interactuando con el usuario de la aplicación y eso podría provocar un Error (Excepcion).

```
static int leerOpcion()throws java.io.IOException {
      int opcion;
      out.print( "Seleccione su opción: " );
      opcion = Integer.parseInt( in.readLine() );
      out.println();
      return opcion;
}
```

10. **Escribir, dentro de la clase AplCuenta, después de leerOpcion, la función ejecutarAccion( )**

```
static boolean ejecutarAccion(int opcion)throws java.io.IOException
{
      boolean noSalir = true;
      int opc = 0;
      switch( opcion ){
                case 1:  registrarCliente();
                         break;
```

```
                case 2:  abrirCuenta();
                         break;
                case 3: realizarTransacciones();
                         break;
                case 4: mostrarMenuConsultas();
                         opc = leerOpcion();
                         ejecutarConsultas( opc );
                         break;
                case 5: noSalir = false;
                         break;
                default: out.println( "Opción no válida" );
                         break;
        } // fin del switch
        out.println();
        return noSalir;
    }
```

11. Escribir, dentro de la clase AplCuenta, después de ejecutarAcción, la función registrarCliente()

```
    static void registrarCliente() throws java.io.IOException{
        int indice;
        String cedula;
        if ( numCuentas <= 10 ) {
                out.print( "Ingrese el número de cédula: " );
                cedula =in.readLine();
                out.println();
                indice = buscarCliente( cedula );
                if(indice == -1){
                        cedulas[ numCuentas ] = cedula;
                        out.print( "Ingrese el nombre y el apellido: " );
                        nombres[numCuentas] = in.readLine();
                        numCuentas += 1;
                        out.println( "\n El cliente ha sido registrado!!!" );
                        out.println();
                }
                else
                {
                        out.println( "\n El cliente ya existe" );
                }
        }
        else
        {
                out.println( "\n No se pueden registrar más clientes" );
        }
    }
```

12. Escribir, dentro de la clase AplCuenta, después de registrarCliente, la función abrirCuenta()

Observar cómo se extrae un carácter de una cadena de caracteres por medio del método (función) charAt(), indicándole la posición en la cadena donde se desea extraer el carácter. En toda cadena su primer carácter se encuentra en la posición 0. Se puede averiguar la longitud de la cadena por medio del atributo "length".

```
  static void abrirCuenta()throws java.io.IOException{
    String cedula;
    int indice;
    double monto = 0;
    boolean noSalir = true;
    String continuar = "";
    out.print( "Ingrese el número de cédula del cliente que será dueño de la cuenta: " );
    cedula = in.readLine();
    out.println();
    do
    {
      indice = buscarCliente( cedula );
```

```
        if ( indice != -1 )
        {
            if ( buscarCuenta("" + cedula.charAt(0) + indice)== -1)
            {
                    do
                    {
                            out.print( "Ingrese el monto( >= a 1000) con el que desea abrir la cuenta: ");
                        monto = Double.parseDouble( in.readLine() );
                        out.println();
                    } while( monto < 1000 );
                    saldos[ indice ] = monto;
                    idCuentas[ indice ]= "" + cedula.charAt(0) + indice;
            }
            else
            {
                    out.println( "Cuenta ya fue abierta" );
            }
            noSalir = false;
        }
        else
        {
                    out.println( "Cliente no registrado" );
                    out.print( "Desea buscar otro (s/n): " );
                    continuar = in.readLine();
                    if ( continuar.charAt(0) == 's' )
                    {
                            out.print("Ingrese el número de cédula: ");
                            cedula = in.readLine();
                            out.println();
                    }



                    else
                    {
                            noSalir = false;
                    }
        }
        out.println();
    } while ( noSalir );
}
```

13. Escribir, dentro de la clase AplCuenta, después de abrirCuenta, la función buscarCliente()

Observar cómo se comparan cadenas de caracteres con el método "equals" y no con el operador "==" como se hace tradicionalmente con los valores numéricos o primitivos. Se puede averiguar la cantidad de elementos en un arreglo con el atributo "length" de forma similar cuando se determina el tamaño de una cadena de caracteres.

```
static int buscarCliente( String pcedula )
{
    boolean encontrado = false;
    int i = 0;
    int indice = -1;
    while( !encontrado && cedulas[i]!= null && i < cedulas.length)
            {
            if( cedulas[i].equals( pcedula ) )
                    {
                    indice = i;
                    encontrado = true;
            }
            i += 1;
    }
    return indice;
```

```
}
```

14. Escribir, dentro de la clase AplCuenta, la función listarCuentasIds()

```
static void listarCuentasIds()
{
    int i = 0;
    while( i < idCuentas.length )
    {
        if( idCuentas[i] != null )
            out.println( idCuentas[i] );
        i+= 1;
    }
}
```

15. Escribir, dentro de la clase AplCuenta, después de listarCuentasIds, la función mostrarMenuTransacciones()

```
static void mostrarMenuTransacciones() {
    out.println("1. Depositar");
    out.println("2. Retirar");
    out.println("3. Ver Saldo");
    out.println("4. Salir");
    out.println();
}
```

16. Escribir, dentro de la clase AplCuenta, la función verificarCuenta()

```
static int verificarCuenta()throws java.io.IOException
{
    String numCuenta;
    int indice = -1;
    out.print( "Ingrese el número de cuenta: " );
    numCuenta = in.readLine();
    out.println();
    indice = buscarCuenta( numCuenta );
    return indice;
}
```

17. Escribir, dentro de la clase AplCuenta, la función buscarCuenta()

```
static int buscarCuenta( String pnumero )
{
    int indice = -1;
    int i = 0;
    while( i < idCuentas.length && indice == -1)
    {
        if( idCuentas[i] != null )
        {
            if( idCuentas[i].equals( pnumero ) )
                indice = i;
        }
        i+= 1;
    }
    return indice;
}
```

18. Escribir, dentro de la clase AplCuenta, después de buscarCliente, la función mostrarMenuConsultas()

```
static void mostrarMenuConsultas()
{
    out.println("1. Ver clientes por cédula");
```

```
        out.println("2. Ver clientes por nombre");
        out.println("3. Ver Cuentas por número");
        out.println();
    }
```

19. Escribir, dentro de la clase AplCuenta, después de mostrarMenuConsultas, el encabezado de ejecutarConsultas()

```
    static void ejecutarConsultas( int popcion )
    {
        switch( popcion)
        {
                case 1: listarInformacion(cedulas);
                        break;
                case 2: listarInformacion(nombres);
                        break;
                case 3: listarCuentasIds();
                        break;
                default: out.println("Valor no válido");
                         break;
        }
        out.println();
            }
```

20. Escribir, dentro de la clase AplCuenta, la función listarInformación()

```
    static void listarInformacion( String[] plista )
    {
        int i=0;
        while( plista[i] != null && i < plista.length )
        {
                out.println( plista[i] );
                i+=1;
        }
    }
```

21. Escribir, dentro de la clase AplCuenta, escriba la función ejecutarTransacciones()

```
    static boolean ejecutarTransacciones( int popcion, int pindice ) throws java.io.IOException
    {
        boolean seguir = true;
                switch( popcion )
                        {
                        case 1: depositar( pindice );
                                break;
                        case 2: retirar( pindice );
                                break;
                        case 3: verSaldo( pindice );
                                break;
                        case 4: seguir = false;
                                break;
                        default: out.println( "Valor no válido" );
                                 break;
                }
                out.println();
                return seguir;
    }
```

22. Escribir, dentro de la clase AplCuenta, después de ejecutarTransacciones, el encabezado de la función realizarTransacciones()

```
    static void realizarTransacciones()throws java.io.IOException
```

```
{
    int opc = 0;
    boolean seguir = true;
    int indice = verificarCuenta();
    if ( indice!= -1 )
    {
            out.println( "Cedula: " + cedulas[ indice ] );
            out.println( "Cliente: "+ nombres[ indice ] );
            out.println();
            do{
                    mostrarMenuTransacciones();
                    opc = leerOpcion();
                    seguir = ejecutarTransacciones( opc, indice );
            } while ( seguir );
    }
    else
            out.println( "Cuenta no existe" );
}
```

23. Escribir, dentro de la clase AplCuenta, la función depositar()

```
static void depositar( int pindice )throws java.io.IOException
{
    double monto = 0;
    out.print( "Ingrese el monto del depósito: " );
    monto = Double.parseDouble( in.readLine( ) );
    out.println();
    saldos[pindice] += monto;
    out.println( "Su depósito fue de " + monto + "colones");
}
```

24. Escribir, dentro de la clase AplCuenta, después de depositar, la función retirar()

```
static void retirar( int pindice )throws java.io.IOException
{

}
```

25. Escribir, dentro de la clase AplCuenta, después de retirar, la función verSaldo()

```
static void verSaldo( int pindice )throws java.io.IOException
{

}
```

26. Para Ejecutar haga clic derecho sobre el cuadro de la clase AplCuenta y seleccione la ejecución del método main()

No deben existir errores de compilación, de ser así, deben corregirse según las recomendaciones del IDE. Debe verificar cada una de las funcionalidades realizadas.

SEGUNDA PARTE

En esta sección se deberá implementar la funcionalidad de aquellos métodos de la clase AplCuenta que contienen únicamente su encabezado, de esta forma el estudiante completará la funcionalidad de la aplicación solicitada en un inicio, demostrando y practicando al mismo tiempo los conocimientos que han sido adquiridos a través del mismo.

e. Análisis de Resultados

Al finalizar el laboratorio el estudiante estará capacitado para realizar una pequeña aplicación respondiendo a los requerimientos solicitados. A partir de ello será capaz de crear diversos proyectos en los cuales declare y utilice variables de diferentes tipos de datos primitivos, implementando correctamente la funcionalidad de cada uno de los métodos del programa, de tal forma que puedan finalmente compilarse y ejecutarse en Java.

f. Aspectos relativos a la entrega

- El laboratorio debe realizarse de forma individual.
- **En la codificación de este laboratorio DEBE aplicar las reglas del estándar de codificación.**
- El proyecto creado en este laboratorio debe ser enviado a través del TEC-DIGITAL en un archivo comprimido .ZIP con el nombre: **PRYLAB2_APELLIDO_NOMBRE.ZIP**, por ejemplo: **PRYLAB2_SOTO_LUIS.ZIP**
  - **Rubro de entrega: Laboratorio 02**

Para reflexionar