



CIENCIAS DE LA INGENIERÍA

**CARRERA DE SISTEMAS DE INFORMACIÓN
CARRERA DE INFORMÁTICA**

PROGRAMACIÓN II

SEMESTRE 2023 B

LABORATORIO S2

TEMA: Conceptos esenciales de la Programación Orientada a Objetos

PROFESOR: Ing. Patricio Coba M.

QUITO, 2023

1. TEMA: Conceptos esenciales de la POO

2. OBJETIVOS:

- Identificar las características de la POO
- Aplicar los conceptos esenciales de la POO en programas con Java

3. INTRODUCCIÓN:

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. La programación orientada a objetos disminuye los errores y promueve la reutilización del código. IBM

La programación orientada a objetos (POO) es un paradigma de la programación que desde hace varias décadas instituyó una nueva forma de programa aplicando conceptos para que el código resultante sea:

- Reutilizable
- Organizado.
- Sencillo de mantener.

Conceptos básicos de la programación orientada a objetos

En este tipo de programación se organiza el código en clases y, posteriormente, se crean los objetos. Una clase es una plantilla que define de manera general cómo serán los objetos de un tipo concreto.

Por ejemplo, imaginemos la clase “Bicicletas”. Todas las bicicletas tienen características comunes: dos ruedas, un sillín, un manillar, dos pedales..., lo que en programación conocemos como “Atributos”.



En programación, las clases son todos los tipos de datos que el programador establece previamente y que sirven como modelo o plantilla para cualquier objeto que se incluyan posteriormente en el software.

Una vez están desarrolladas las clases, pueden crearse los objetos, que son los elementos que se encuentran dentro de cada clase. Siguiendo el ejemplo de las bicicletas, una bicicleta de montaña podría ser un objeto de la clase “bicicleta”. Comparten las características comunes, pero también tienen atributos distintos.

Respecto a los atributos, son las características que permiten diferenciar a un objeto de otro, que en el caso de las bicicletas podrían ser el color, la velocidad, la marcha... De igual forma, cada clase también cuenta con atributos concretos.

Por último, los métodos. Estos son todas las funciones que se han establecido en una clase y que definen cómo se comportan los objetos. Los métodos de las bicicletas serían, por ejemplo, avanzar o frenar, entre otros.

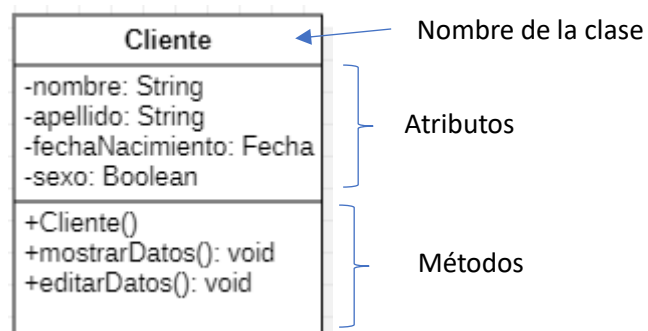
Principios de POO

- **Abstracción.** Obtener de los objetos solo las características y atributos que se requieren, se desecha lo que no hace falta. Para obtenerlos hay que siempre tomar en cuenta el contexto del programa a desarrollar.
- **Encapsulamiento.** Es la información que tiene un objeto (interna) que no debe mostrarse y la parte de la información que puede ser visible (externa). Esto protege a los datos de los objetos de no ser manipulados al antojo.
- **Herencia.** Capacidad de crear clases a partir de otras que ya existen, esto permite utilizar lo ya creado anteriormente y organizar las clases de manera jerárquica. A la clase de la que se hereda se le conoce como clase base y la que hereda clase heredada. Esta última, además de heredar los rasgos de la primera, también añade nuevos y modifica algunos, gracias al principio de polimorfismo.
- **Polimorfismo.** Capacidad de comportarse de varias maneras (formas), aplicable tanto a objetos como a los métodos

Clase

Es una plantilla, un molde con el cual se pueden crear los objetos. Una clase define las características (atributos) y las funcionalidades (métodos) que todos los objetos que pertenezcan a esa clase tienen. Las clases determinan los datos que todos los objetos del mismo tipo comparten, así como su comportamiento. Una clase además es considerada como un nuevo tipo de dato pues con el se crean variables (objetos) las mismas que se pueden instanciar (asignar memoria), editar sus datos, mostrar sus datos, pasarlas como parámetros y liberar de memoria (destruir).

La clase tiene un nombre el cual usualmente es un sustantivo escrito en singular y su primera letra siempre está en MAYÚSCULA, si el nombre de la clase tiene dos palabras la segunda también debe comenzar con MAYÚSCULA.



Representación gráfica de una clase en UML

Nota: Los métodos de la clase se consideran como su interfaz, es decir que a través de ellos se puede acceder a los atributos ya que estos deben permanecer asegurados.

Atributos

Las clases de componen de atributos que son las características de los objetos, estos son los datos que poseerán dichos objetos.

Se recomienda poner el nombre de los atributos aplicando camelCase y que sean lo más descriptibles. Los atributos siempre son privados.

Métodos

Todo objeto tiene una funcionalidad, lo que se puede hacer con él, esto se le conoce como métodos. Los métodos permiten acceder a los atributos de la clase por lo que usualmente los métodos son públicos y los atributos privados. Se recomienda colocar nombres a los métodos que representen la funcionalidad que van a realizar.

En POO los métodos pueden ser sobrecargados, esto quiere decir que se pueden definir varios métodos con el mismo nombre dentro de una clase, pero deben diferenciarse por el tipo o el número de argumentos que reciben.

Modos de acceso

Para que se pueda mantener la seguridad de los atributos y que los métodos se constituyan en la interfaz de la clase se requiere de manejar los modos de acceso, estos modos permiten definir la manera en la que se pueden acceder. Los modos son:

- public (público): se puede acceder desde cualquier clase.
- private (privado): se puede acceder únicamente desde dentro de la misma clase.
- protected (protegido): es público para la misma clase o sus derivadas (herencia), pero para el resto de las clases son privadas
- static (estático): un atributo o método estático en realidad pertenece a la clase y no a los objetos por lo que no hace un objeto para que exista.

Constructores

Un constructor es un método especial de una clase que inicializa los atributos de un objeto al momento de su instanciación (asignación de memoria de la variable objeto). Puede haber muchos constructores dentro de una clase.

Las características de un constructor son:

- Mismo nombre de la clase
- No devuelve nada, tampoco se coloca void
- Se invoca (llama) al instanciar un objeto
- Si no se crea, el compilador crea uno por defecto

Destruyores

El destructor es el método con el cual el objeto se destruye y ya no se puede acceder a sus datos, en realidad lo que sucede es que se libera el objeto de la memoria.

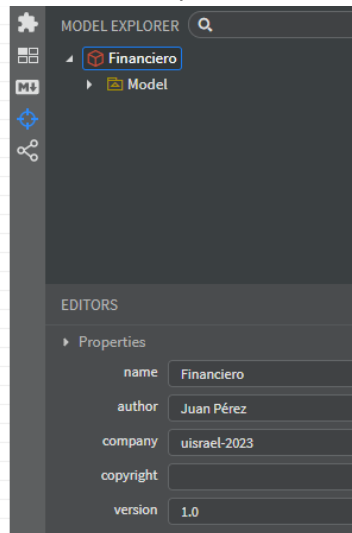
En lenguaje C++ es uno de los pocos lenguajes donde aún se utiliza la llamada al destructor para realizar la acción de liberar memoria de los objetos que ya no se utilizan.

Recolector de basura

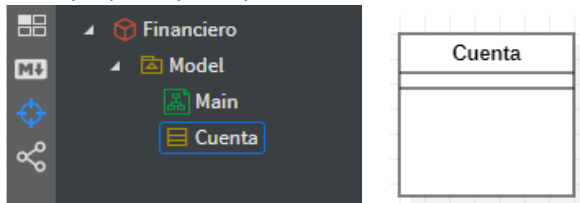
Actualmente los lenguajes orientados a objetos utilizan la técnica del recolector de basura (Garbage Collector) que realiza el proceso automático de liberar de memoria (destruir) los objetos que ya no se utilizan, esto es muy bueno ya que eso permite realizar el proceso sin tener que hacerlo específicamente y que el programador se olvide de hacerlo.

I Parte: Representación de la clase en UML

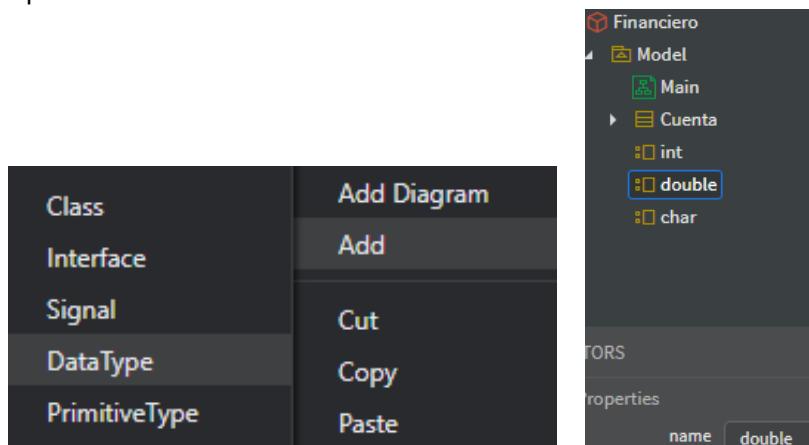
1. Abrir StarUML y colocar el nombre “Financiero” y sus datos en el modelo.

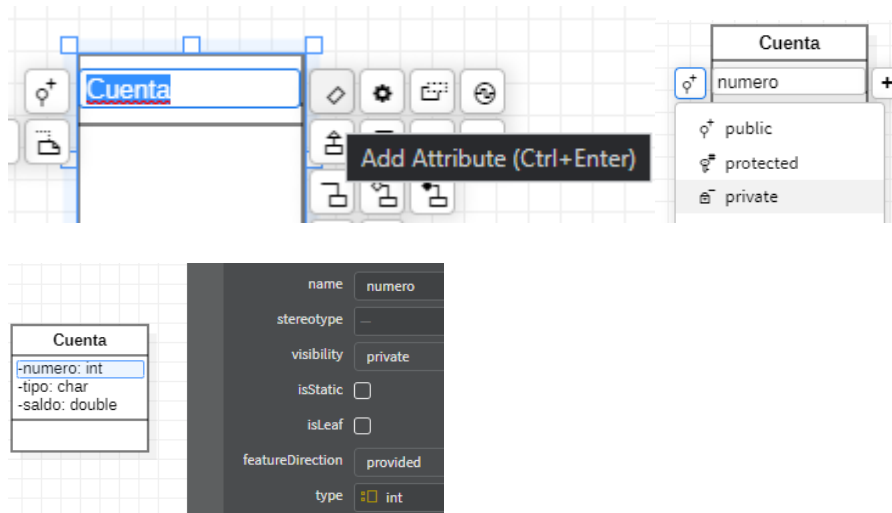


2. En el paquete principal “Model” abrir el modelo “Main” y crear la clase para la Cuenta

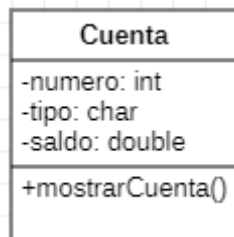


3. Definir los atributos que va a tener la cuenta (número de cuenta, tipo de cuenta, saldo de la cuenta), cada atributo debe definirse con su respectivo tipo de dato. Se recomienda crear los tipos de datos en UML.

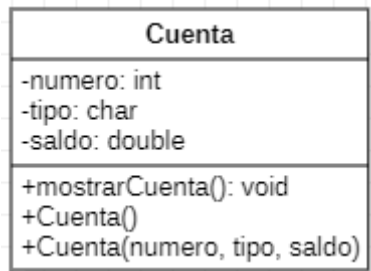


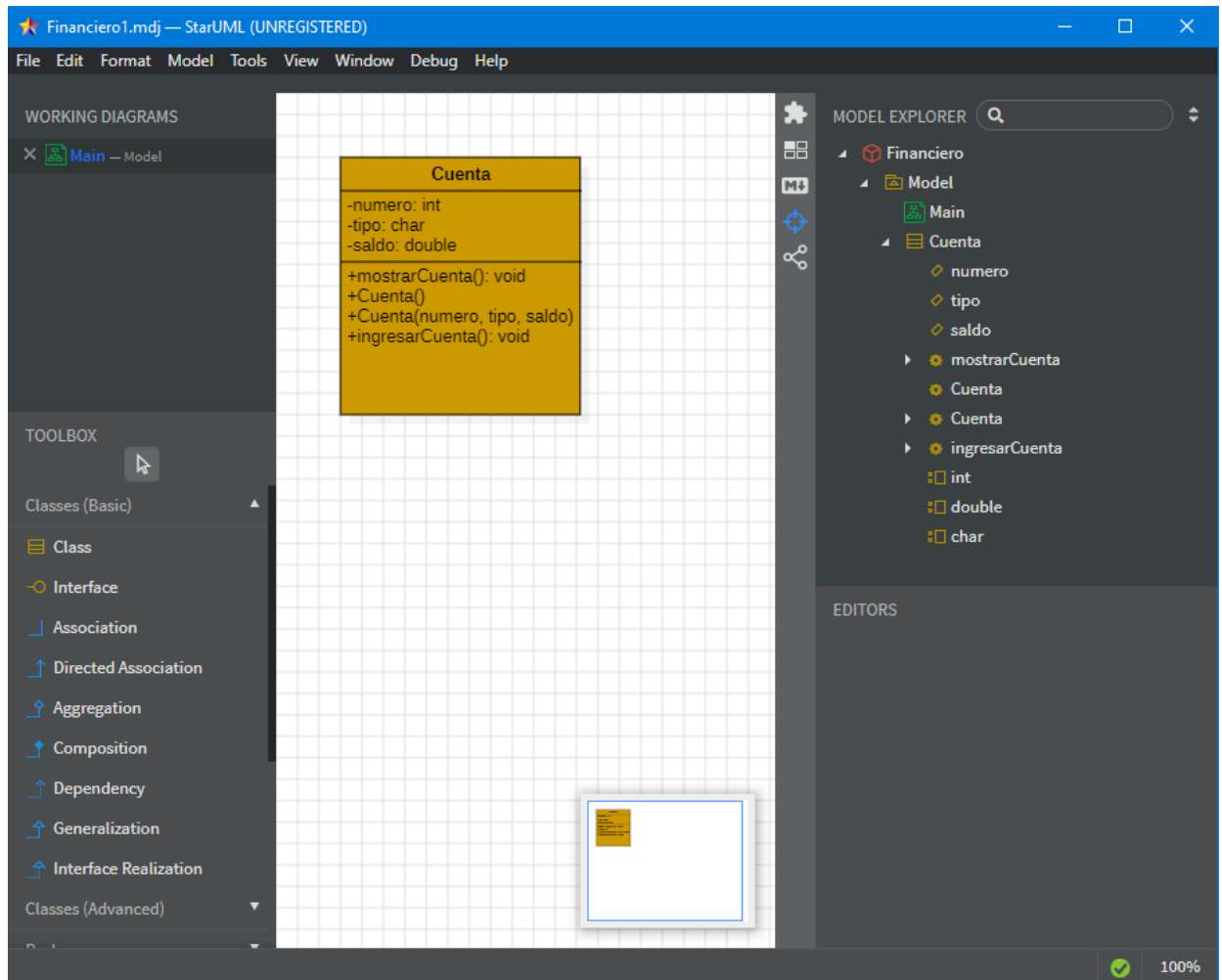


4. Crear los métodos



5. Crear los constructores

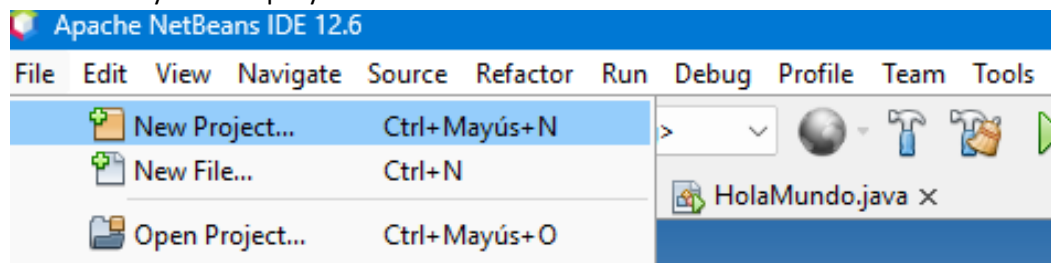




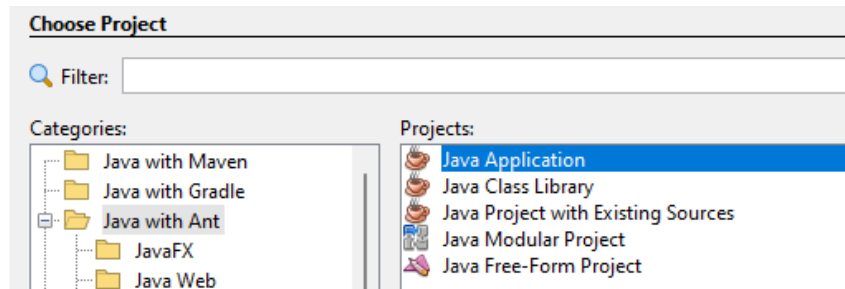
II Parte: Creación de un nuevo proyecto con del IDE (Netbeans o Eclipse)

Vamos a crear un proyecto para crear un programa para manipular los datos de cuentas de ahorros de una entidad financiera. Se utiliza Netbeans para el ejemplo.

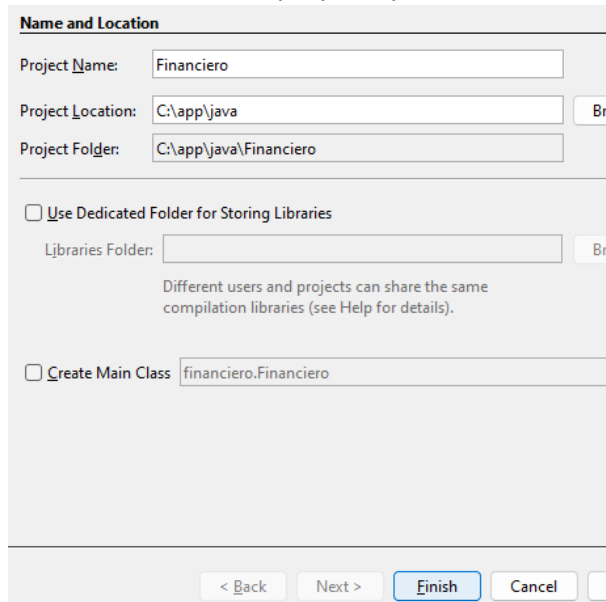
1. Abrir el IDE y crear el proyecto



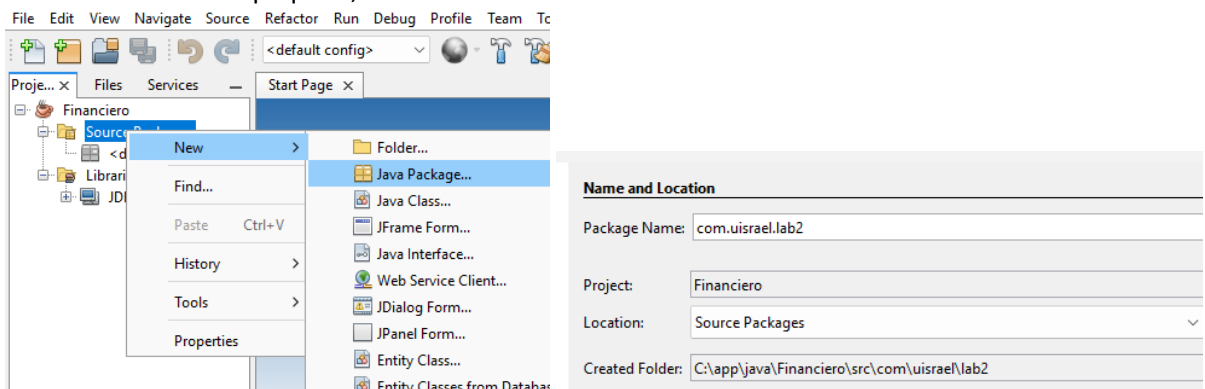
2. Escoger la categoría del proyecto



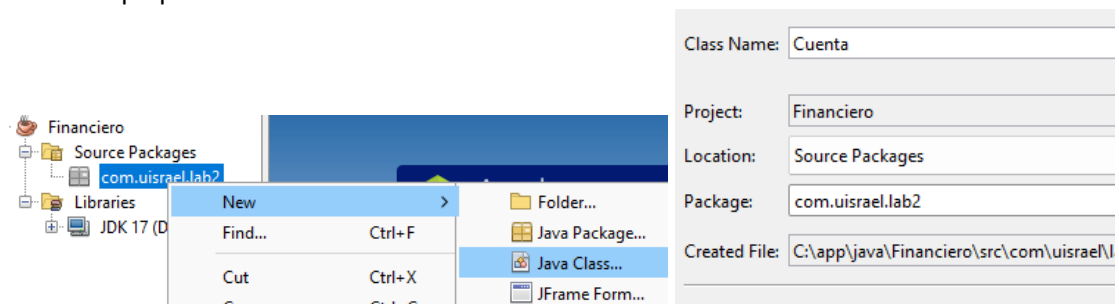
3. Colocar el nombre del proyecto y finalizar. Desactivar la creación de la clase principal.



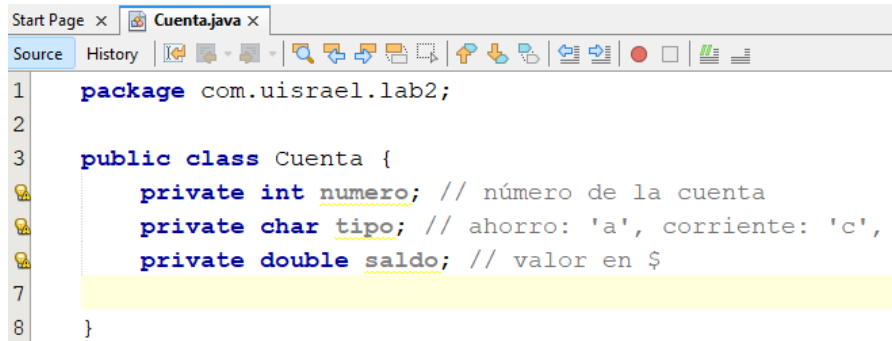
4. Crear un nuevo paquete, colocar el nombre: com.uisrael.lab2



5. En el paquete crear una clase llamada Cuenta.

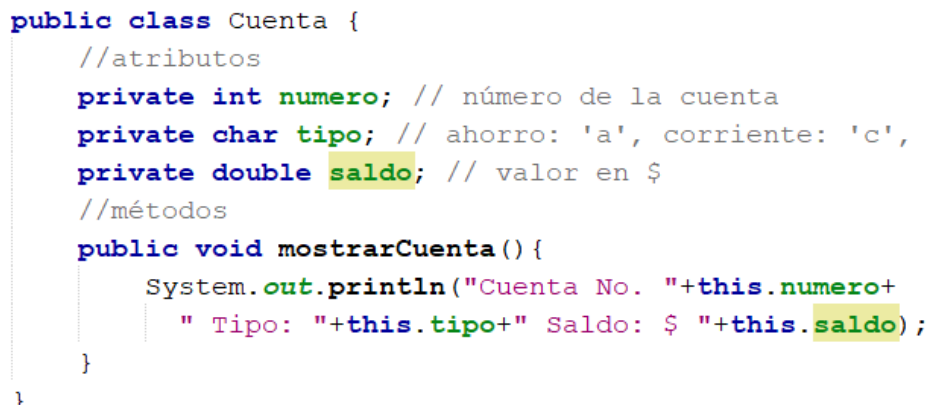


6. En la clase definir los atributos (características de la Clase) según las necesidades del programa.



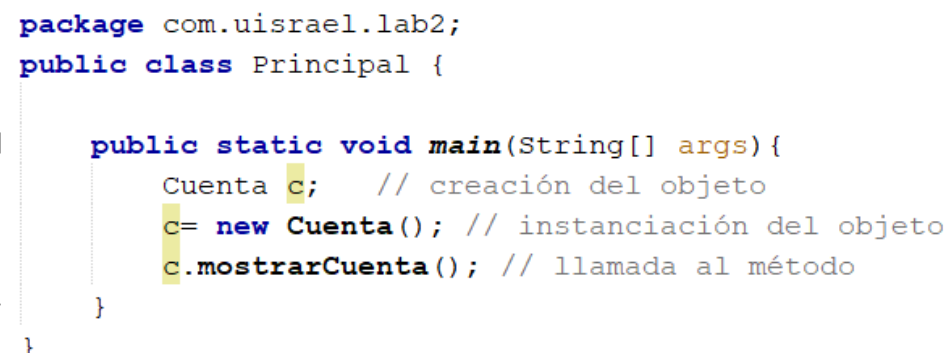
```
1 package com.uisrael.lab2;
2
3 public class Cuenta {
4     private int numero; // número de la cuenta
5     private char tipo; // ahorro: 'a', corriente: 'c',
6     private double saldo; // valor en $
7
8 }
```

7. Crear un método para mostrar por consola los datos de los objetos.



```
public class Cuenta {
    //atributos
    private int numero; // número de la cuenta
    private char tipo; // ahorro: 'a', corriente: 'c',
    private double saldo; // valor en $
    //métodos
    public void mostrarCuenta() {
        System.out.println("Cuenta No. "+this.numero+
            " Tipo: "+this.tipo+" Saldo: $ "+this.saldo);
    }
}
```

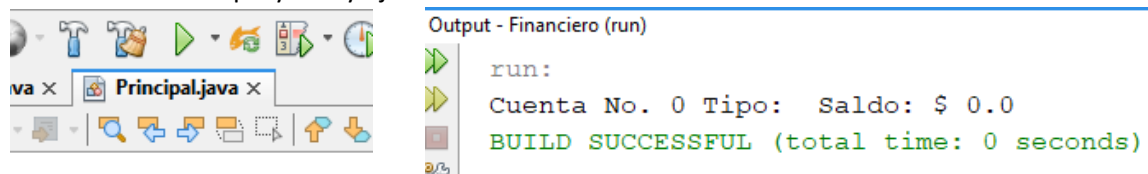
8. Crear una nueva clase “Principal” que tenga el método main. En el main crear un objeto a utilizar.



```
package com.uisrael.lab2;
public class Principal {

    public static void main(String[] args){
        Cuenta c; // creación del objeto
        c= new Cuenta(); // instanciación del objeto
        c.mostrarCuenta(); // llamada al método
    }
}
```

9. Construir el proyecto y Ejecutarlo



```
Output - Financiero (run)
run:
Cuenta No. 0 Tipo: Saldo: $ 0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

III Parte: Objetos, Constructores

Crear e instanciar objetos a través de constructores.

1. Creación de constructores

```
// constructor sin argumentos
public Cuenta() {
    this.numero=1000;
    this.tipo='a';
    this.saldo=50;
}
//constructor con 3 argumentos
public Cuenta(int numero, char tipo, double saldo) {
    this.numero = numero;
    this.tipo = tipo;
    this.saldo = saldo;
}
```

2. Instanciación de objetos y llamada a los constructores

```
public class Principal {

    public static void main(String[] args){
        // instancia un objeto c1 y llama al constructor()
        Cuenta c1= new Cuenta();
        // instancia un objeto c2 y llama al constructor(int, char, double)
        Cuenta c2= new Cuenta(101,'a',100);
        c1.mostrarCuenta(); //llamada al método con el objeto c1
        c2.mostrarCuenta(); // llamada al método con el objeto c2
    }
}
```

3. Ejecutar y analizar los resultados

Output - Financiero (run)

```
run:
Cuenta No. 1000 Tipo: a Saldo: $ 50.0
Cuenta No. 101 Tipo: a Saldo: $ 100.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

IV Parte: Reto

1. Investigar sobre la clase Scanner para lectura de datos desde teclado.
2. Crear un método llamado "ingresarCuenta" en la que aplique la clase Scanner para ingresar los datos de la cuenta desde teclado.
3. Llamar al método creado y luego el método mostrarCuenta()

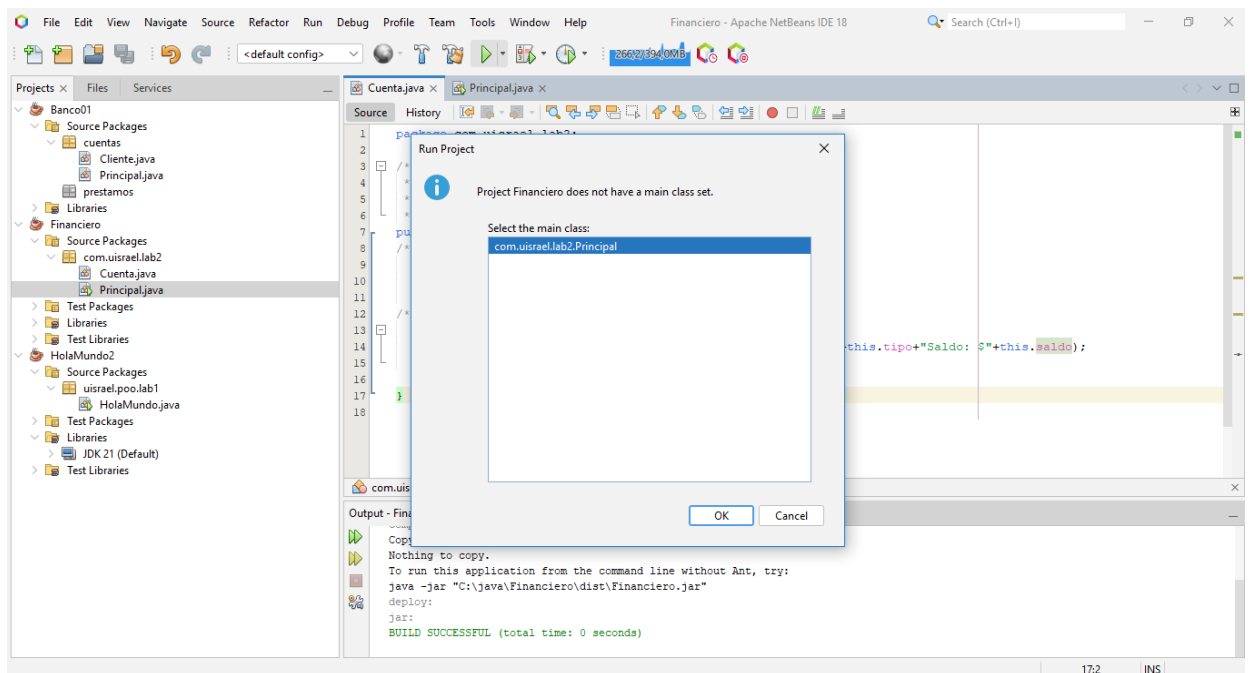
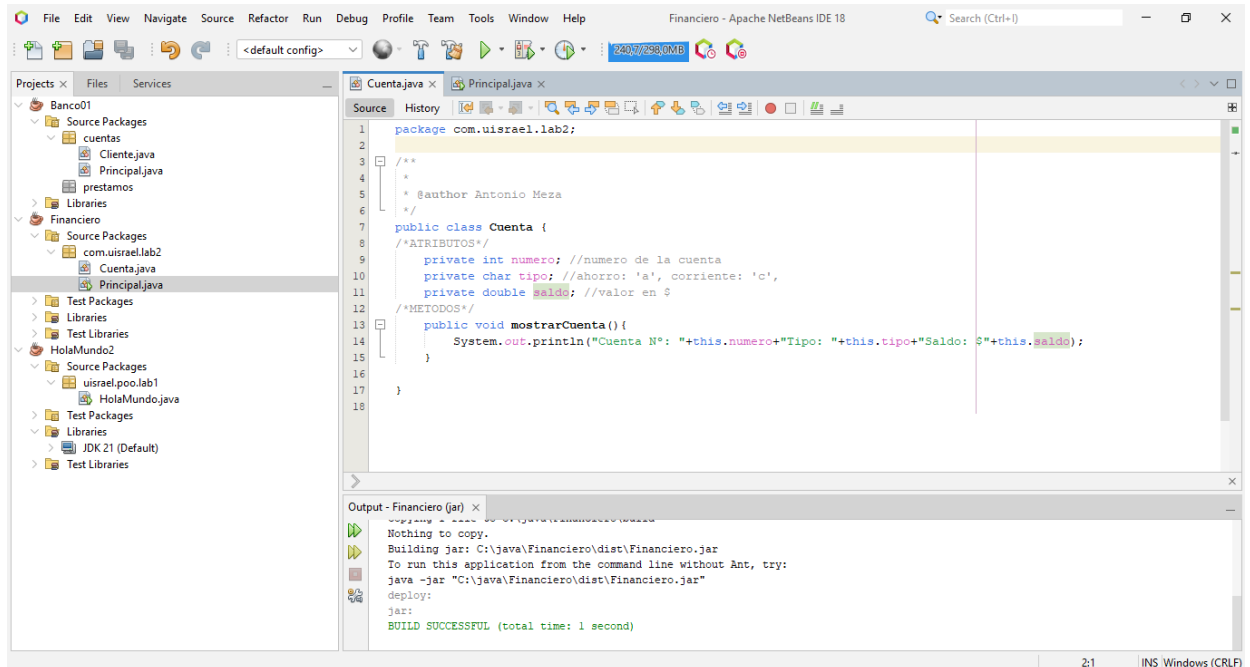
V Parte: Envío

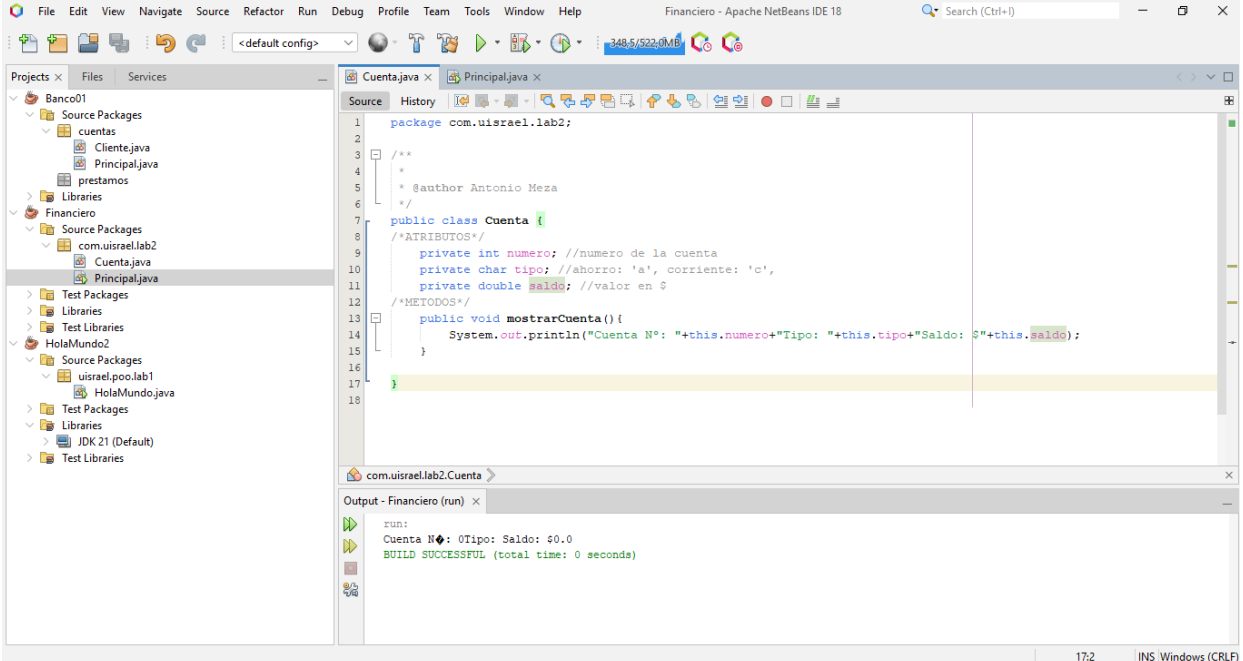
- Colocar en al menos 10 líneas de código comentarios apropiados a cada línea.

- Colocar al final de este documento capturas de pantalla completa de la ejecución del programa de cada parte.
- Comprimir todo el proyecto y subirlo en la tarea correspondiente.

EVIDENCIAS

- Construcción y ejecución del proyecto Financiero / Class Cuenta





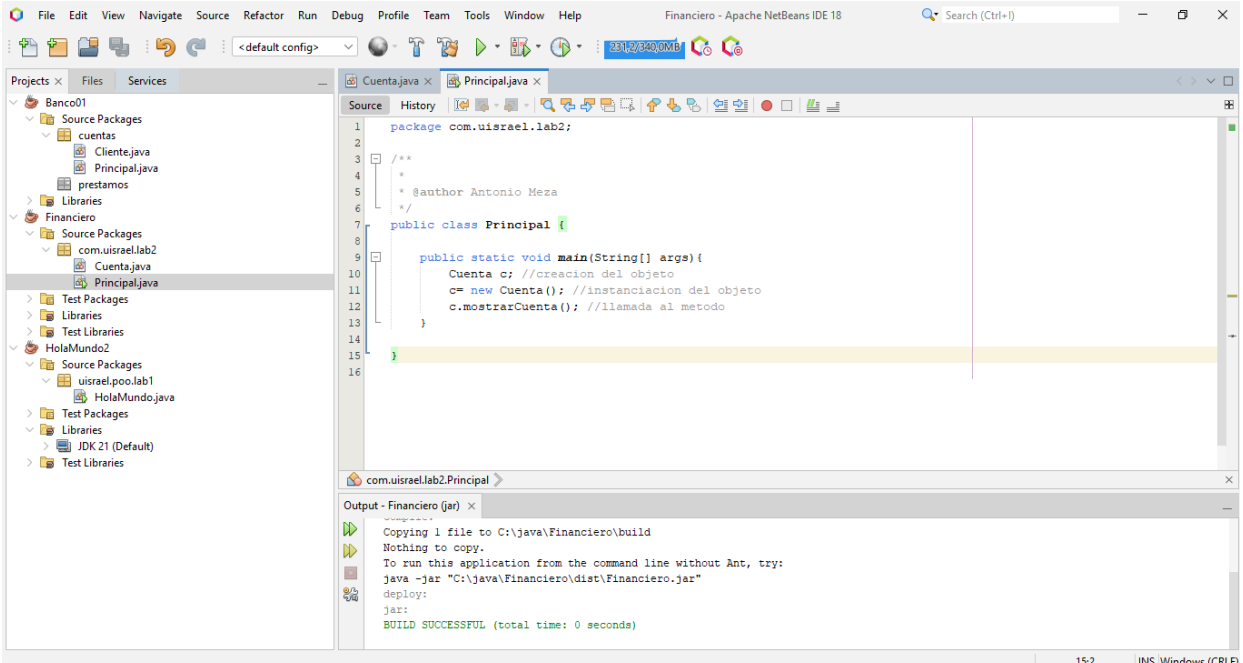
The screenshot shows the NetBeans IDE with the 'Financiero' project open. The 'Cuenta.java' file is selected in the 'Files' view. The 'Source' view displays the code for the 'Cuenta' class, which includes attributes 'numero', 'tipo', and 'saldo', and a method 'mostrarCuenta()'. The 'Output' view shows the successful execution of the class, displaying the output: 'Cuenta N°: 0Tipo: Saldo: \$0.0' and 'BUILD SUCCESSFUL (total time: 0 seconds)'.

```
1 package com.uisrael.lab2;
2
3 /**
4  *
5  * @author Antonio Meza
6  */
7 public class Cuenta {
8     /**ATRIBUTOS*/
9     private int numero; //numero de la cuenta
10    private char tipo; //ahorro: 'a', corriente: 'c',
11    private double saldo; //valor en $
12    /**METODOS*/
13    public void mostrarCuenta() {
14        System.out.println("Cuenta N°: "+this.numero+"Tipo: "+this.tipo+"Saldo: $" +this.saldo);
15    }
16 }
17
18
```

Output - Financiero (run) ×

```
run:
Cuenta N°: 0Tipo: Saldo: $0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Construcción y ejecución de la Class Principal

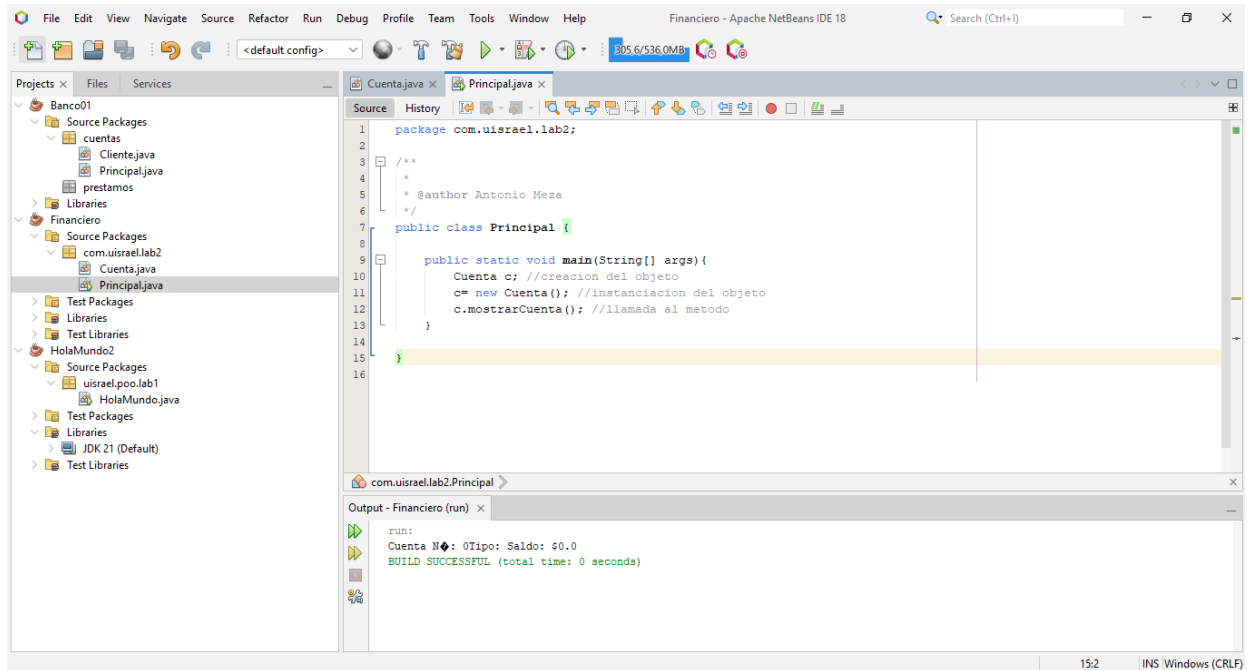


The screenshot shows the NetBeans IDE with the 'Financiero' project open. The 'Principal.java' file is selected in the 'Files' view. The 'Source' view displays the code for the 'Principal' class, which includes a static method 'main()' that creates a 'Cuenta' object and calls its 'mostrarCuenta()' method. The 'Output' view shows the successful execution of the class, displaying the output: 'Copying 1 file to C:\java\Financiero\build', 'Nothing to copy.', 'To run this application from the command line without Ant, try: java -jar "C:\java\Financiero\dist\Financiero.jar"', 'deploy: jar:', and 'BUILD SUCCESSFUL (total time: 0 seconds)'.

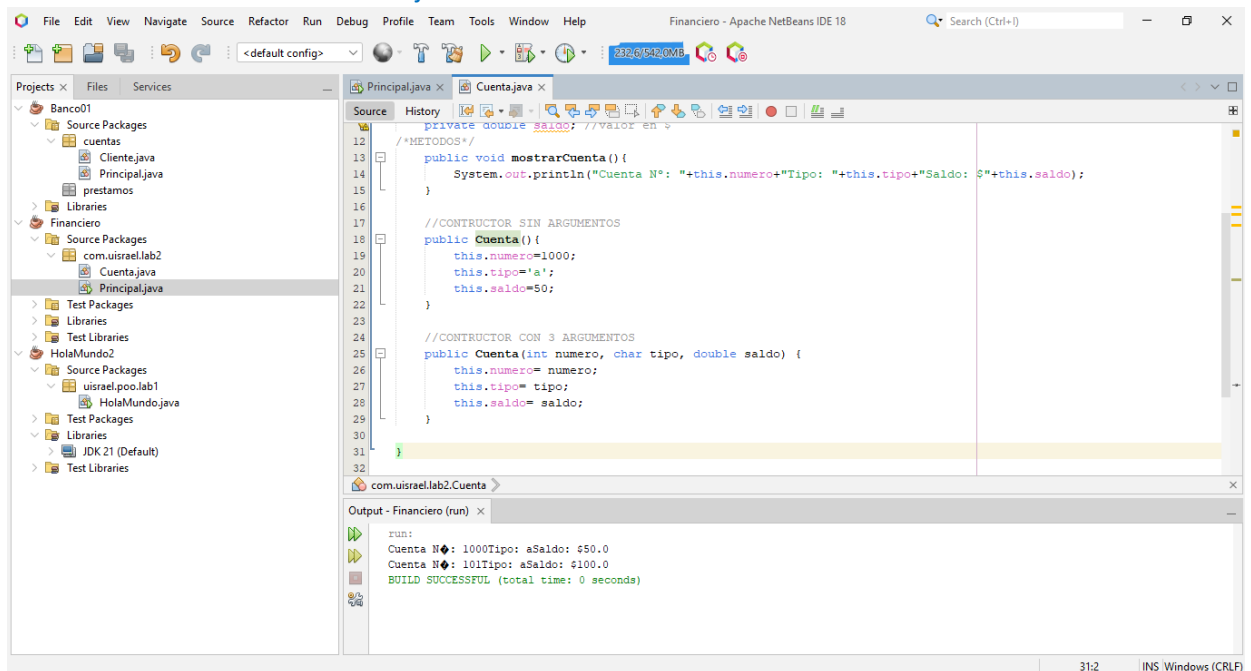
```
1 package com.uisrael.lab2;
2
3 /**
4  *
5  * @author Antonio Meza
6  */
7 public class Principal {
8
9     public static void main(String[] args) {
10        Cuenta c; //creacion del objeto
11        c= new Cuenta(); //instanciacion del objeto
12        c.mostrarCuenta(); //llamada al metodo
13    }
14 }
15
16
```

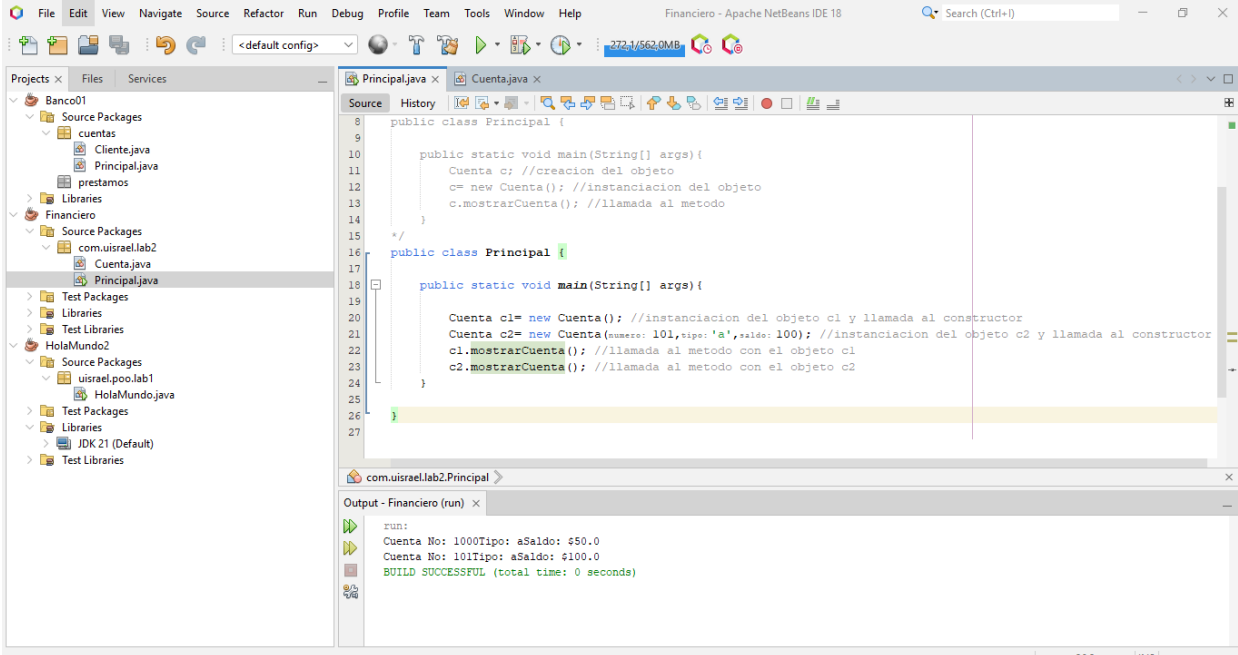
Output - Financiero (jar) ×

```
Copying 1 file to C:\java\Financiero\build
Nothing to copy.
To run this application from the command line without Ant, try:
java -jar "C:\java\Financiero\dist\Financiero.jar"
deploy:
jar:
BUILD SUCCESSFUL (total time: 0 seconds)
```



- Creación e instancia de objetos a través de constructores.





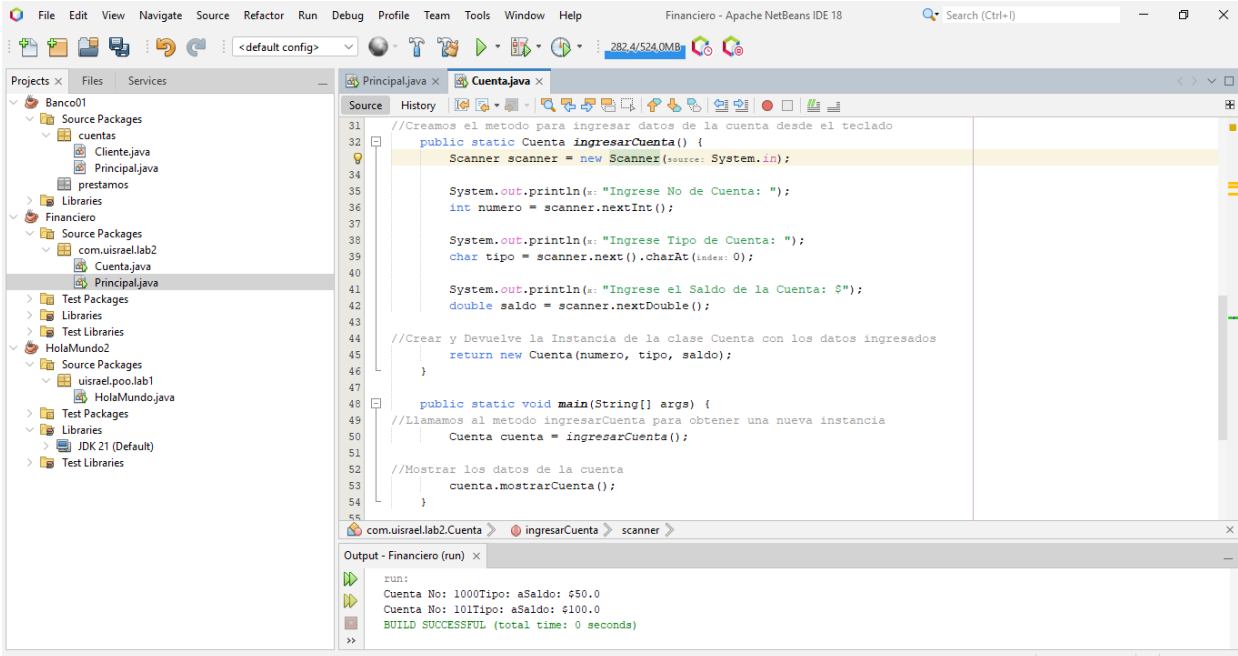
```
public class Principal {  
    public static void main(String[] args) {  
        Cuenta c; //creacion del objeto  
        c= new Cuenta(); //instanciacion del objeto  
        c.mostrarCuenta(); //llamada al metodo  
    }  
}  
  
public class Principal {  
    public static void main(String[] args) {  
        Cuenta c1= new Cuenta(); //instanciacion del objeto c1 y llamada al constructor  
        Cuenta c2= new Cuenta(numero: 101, tipo: 'a', saldo: 100); //instanciacion del objeto c2 y llamada al constructor  
        c1.mostrarCuenta(); //llamada al metodo con el objeto c1  
        c2.mostrarCuenta(); //llamada al metodo con el objeto c2  
    }  
}
```

Output - Financiero (run) x

```
run:  
Cuenta No: 1000Tipo: aSaldo: $50.0  
Cuenta No: 101Tipo: aSaldo: $100.0  
BUILD SUCCESSFUL (total time: 0 seconds)
```

VI Parte: Reto

- Investigar sobre la clase Scanner para lectura de datos desde teclado.
- Crear un método llamado “ingresarCuenta” en la que aplique la clase Scanner para ingresar los datos de la cuenta desde teclado.
- Llamar al método creado y luego el método mostrarCuenta()



```
//Creamos el metodo para ingresar datos de la cuenta desde el teclado  
public static Cuenta ingresarCuenta() {  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.println("Ingrese No de Cuenta: ");  
    int numero = scanner.nextInt();  
  
    System.out.println("Ingrese Tipo de Cuenta: ");  
    char tipo = scanner.next().charAt(0);  
  
    System.out.println("Ingrese el Saldo de la Cuenta: $");  
    double saldo = scanner.nextDouble();  
  
    //Crear y Devuelve la Instancia de la clase Cuenta con los datos ingresados  
    return new Cuenta(numero, tipo, saldo);  
}  
  
public static void main(String[] args) {  
    //Llamamos al metodo ingresarCuenta para obtener una nueva instancia  
    Cuenta cuenta = ingresarCuenta();  
  
    //Mostrar los datos de la cuenta  
    cuenta.mostrarCuenta();  
}
```

Output - Financiero (run) x

```
run:  
Cuenta No: 1000Tipo: aSaldo: $50.0  
Cuenta No: 101Tipo: aSaldo: $100.0  
BUILD SUCCESSFUL (total time: 0 seconds)
```

UML : Financiero

