

## **Part 1: Depuració del Codi**

**Objectiu:** *entendre com es gestionen les operacions de dipositar i retirar, com es modifiquen les variables i com es capturen les excepcions en cas d'errors.*

Documenteu els següents passos:

### **1. Descripció inicial del codi:**

- **Què fa el codi? (Explicar breument).**

El código implementa un sistema bancario básico a través de una clase Account, que permite al usuario manipular el saldo mediante las operaciones de ingreso y retirada de dinero. La clase Main es la que construye una cuenta de cliente y realiza operaciones con ella, comprobando las posibles excepciones.

- **Quins són els mètodes més importants i què fan?**

`depositAmount(double amount)`

- Afegeix un import al saldo del compte.
- Llença una excepció si l'import és negatiu.

`withdrawAmount(double amount)`

- Resta un import del saldo si hi ha prou diners.
- Llença excepcions si l'import és negatiu o si el saldo és insuficient.

`getBalance()`

- Retorna el saldo actual del compte.

- **Quin és el valor inicial del saldo (balance) abans de realitzar qualsevol operació?**

El compte `myAccount` es crea amb un saldo inicial de **2500**.

**2. Posar punts de control (Breakpoints):** Per depurar el codi, utilitza els punts de control (breakpoints). Això permet aturar l'execució del codi en determinats punts i examinar l'estat de les variables. Per afegir un punt de control, fes clic a la barra de l'esquerra de la línia on vols aturar el codi.

- On has col·locat els punts de control (breakpoints) i per què?
- Inclou una captura de pantalla de Eclipse amb els breakpoints activats abans de començar la depuració.

Línia 9 de **Main.java** → Just abans de cridar **withdrawAmount(2300)** per veure l'estat inicial del compte.

Línia 16 de **Main.java** → Abans de fer el dipòsit per verificar l'estat després de la retirada.

Línia 22 de **Main.java** → Abans d'imprimir el saldo final per veure si s'ha actualitzat correctament.

```
1 package SimpleBankingSystem;
2 /**
3  *
4  * @author Flor Martinez
5  */
6 public class Main {
7     public static void main(String[] args) {
8         Account myAccount;
9
10        myAccount = new Account("Flor Martinez", "1000-1234-56-123456789", 2500);
11
12        try {
13            myAccount.withdrawAmount(2300);
14        } catch (Exception e) {
15            System.err.println(e.getMessage());
16            System.out.println("Error al retirar");
17        }
18
19        try {
20            System.out.println("Ingrés al compte");
21            myAccount.depositAmount(1695);
22        } catch (Exception e) {
23            System.err.println(e.getMessage());
24            System.out.println("Error en l'ingrés");
25        }
26
27        System.out.println("El saldo actual es " + myAccount.getBalance());
28    }
29 }
30
31
32
```

### 3. Examina les variables i el flux d'execució:

- A mesura que el codi s'atura a cada punt de control, observa el valor de les variables `name`, `account` i `balance`. Inclou una captura de pantalles dels valors de les variables a mesura que avancen les operacions.

The first screenshot shows the `Main.java` file with a breakpoint at line 6. A tooltip for the `myAccount` variable is displayed, showing its value: `myAccount= Account (id=24)`. The tooltip also shows the values of the `account` and `balance` attributes: `account= "1000-1234-56-123456789" (id=26)` and `balance= 2500.0`. The `name` attribute is also visible: `name= "Flor Martinez" (id=33)`.

The second screenshot shows the `Main.java` file with a breakpoint at line 12. The `Variables` panel on the right displays the current state of the variables:

Name	Value
<code>&lt;init&gt;() returned</code>	(No explicit return value)
<code>args</code>	<code>String[] (id=20)</code>
<code>myAccount</code>	<code>Account (id=27)</code>

#### 4. Explora les excepcions:

- Feu els canvis necessaris al Main per fer saltar les excepcions. Inclou la captura de pantalla d'un missatge d'error generat per una excepció i com es visualitza al terminal o a la consola de Eclipse.

```
Console × Problems Debug Shell
<terminated> Main [Java Application] C:\Users\alumnat\p2\pool\p
No hi ha suficient saldo
Error al retirar
Ingrés al compte
No es pot ingressar una quantitat negativa.
Error en l'ingrés
El saldo actual es 2500.0
```

```
1 /**
2  *
3  * @author Flor Martinez
4  */
5 public class Main {
6     public static void main(String[] args) {
7         Account myAccount;
8
9         myAccount = new Account("Flor Martinez", "1000-1234-56-123456789", 2500);
10
11         try {
12             myAccount.withdrawAmount(5000);
13         } catch (Exception e) {
14             System.err.println(e.getMessage());
15
16             System.out.println("Error al retirar");
17         }
18
19         try {
20             System.out.println("Ingrés al compte");
21             myAccount.depositAmount(-100);
22
23         } catch (Exception e) {
24             System.err.println(e.getMessage());
25             System.out.println("Error en l'ingrés");
26         }
27
28         System.out.println("El saldo actual es " + myAccount.getBalance());
29     }
30 }
31
```

```
Console × Problems Debug Shell
erminated> Main [Java Application] C:\Users\alumnat\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
```