



“BankAccount”

Jonathan Torres Centeno

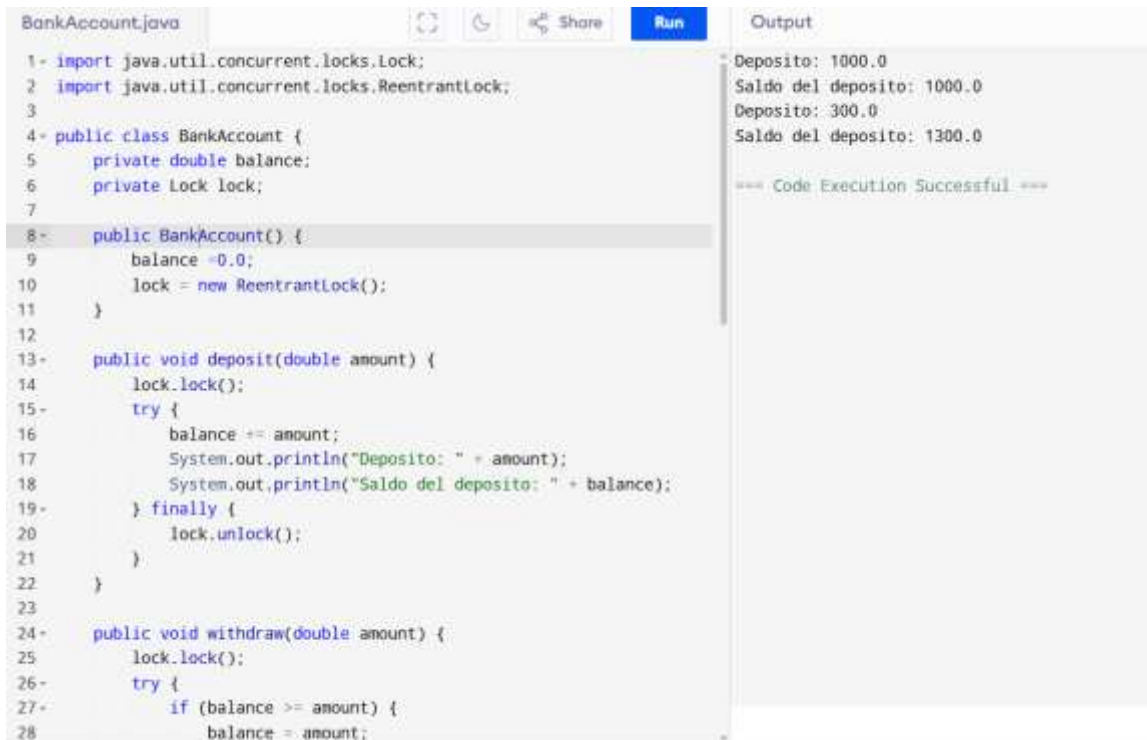
Licenciatura en Sistemas de Computación y Diseño de Software, Instituto Universitario de Yucatán

46220517343718H: Sistemas Operativos

Mtro. Perla Alejandra Landero Heredia

10 De Agosto Del 2025

IMAGEN 1:



The screenshot shows an IDE window titled "BankAccount.java". The code defines a `BankAccount` class with a `balance` attribute and a `lock` of type `ReentrantLock`. It includes a constructor, a `deposit` method, and a `withdraw` method. The `deposit` method uses a `try-finally` block to ensure the lock is released. The `withdraw` method checks if the balance is sufficient before withdrawing. The output window on the right shows the results of a successful code execution.

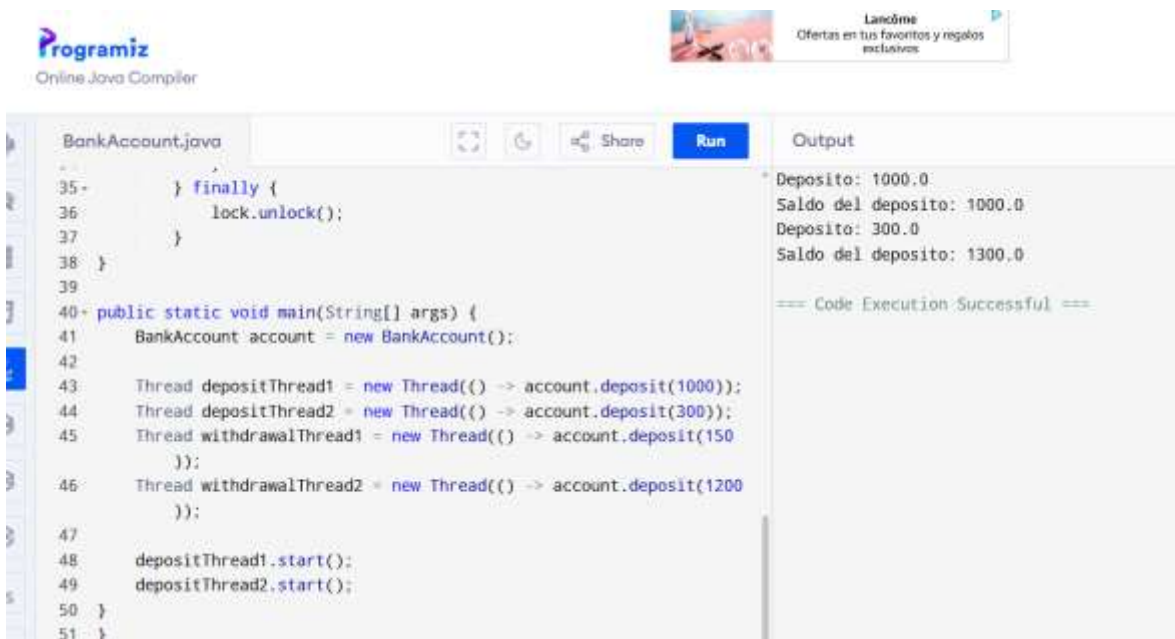
```
1- import java.util.concurrent.locks.Lock;
2- import java.util.concurrent.locks.ReentrantLock;
3-
4- public class BankAccount {
5-     private double balance;
6-     private Lock lock;
7-
8-     public BankAccount() {
9-         balance = 0.0;
10-        lock = new ReentrantLock();
11-    }
12-
13-    public void deposit(double amount) {
14-        lock.lock();
15-        try {
16-            balance += amount;
17-            System.out.println("Deposito: " + amount);
18-            System.out.println("Saldo del deposito: " + balance);
19-        } finally {
20-            lock.unlock();
21-        }
22-    }
23-
24-    public void withdraw(double amount) {
25-        lock.lock();
26-        try {
27-            if (balance >= amount) {
28-                balance = amount;
```

Output

```
Deposito: 1000.0
Saldo del deposito: 1000.0
Deposito: 300.0
Saldo del deposito: 1300.0

=== Code Execution Successful ===
```

IMAGEN 2:



The screenshot shows an IDE window titled "BankAccount.java". The code defines a `BankAccount` class and a `main` method. The `main` method creates a `BankAccount` instance and starts three threads: `depositThread1`, `depositThread2`, and `withdrawalThread1`. The `depositThread1` and `depositThread2` threads call the `deposit` method with amounts of 1000 and 300 respectively. The `withdrawalThread1` thread calls the `deposit` method with an amount of 150. The `withdrawalThread2` thread calls the `deposit` method with an amount of 1200. The output window on the right shows the results of a successful code execution.

```
35-     } finally {
36-         lock.unlock();
37-     }
38- }
39-
40- public static void main(String[] args) {
41-     BankAccount account = new BankAccount();
42-
43-     Thread depositThread1 = new Thread(() -> account.deposit(1000));
44-     Thread depositThread2 = new Thread(() -> account.deposit(300));
45-     Thread withdrawalThread1 = new Thread(() -> account.deposit(150));
46-     Thread withdrawalThread2 = new Thread(() -> account.deposit(1200));
47-
48-     depositThread1.start();
49-     depositThread2.start();
50- }
51- }
```

Output

```
Deposito: 1000.0
Saldo del deposito: 1000.0
Deposito: 300.0
Saldo del deposito: 1300.0

=== Code Execution Successful ===
```