



“SUBPROCESOS Y PROCESOS”

Montserrat Olan Lopez

Ingeniería en sistemas computacionales y diseño de software Instituto Universitario de Yucatán

2303040768: Sistemas Operativos

Docente Ing.Perla Alejandra Landero Heredia

10 De Agosto del 2025

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class BankAccount {
    private double balance;
    private Lock lock;

    public BankAccount(){
        balance = 0.0;
        lock = new ReentrantLock();
    }

    public void deposit(double amount){
        lock.lock();
        try {
            balance += amount;
            System.out.println("Deposito: " + amount);
            System.out.println("Saldo despues del deposito: " + balance);
        } finally {
            lock.unlock();
        }
    }

    public void withdraw(double amount){
        lock.lock();
        try {
            if(balance >= amount){
                balance -= amount;
                System.out.println("Retiro: " + amount);
                System.out.println("Saldo despues del retiro:" + balance);
            } else {
                System.out.println("Intento de retiro:" + amount);
                System.out.println("Saldo insuficiente, retiro cancelado.");
            }
        }
    }
}
```

```
    }  
  } finally {  
    lock.unlock();  
  }  
}
```

```
public static void main(String[] args){  
  BankAccount account = new BankAccount();  
  
  Thread depositThread1 = new Thread(() -> account  
    .deposit(1000));  
  Thread depositThread2 = new Thread(() -> account  
    .deposit(300));  
  Thread withdrawalThread1 = new Thread(() ->  
    account.withdraw(150));  
  Thread withdrawalThread2 = new Thread(() ->  
    account.withdraw(1200));  
  
  depositThread1.start();  
  depositThread2.start();  
  withdrawalThread1.start();  
  withdrawalThread2.start();  
}  
}
```



BankAccount.java



Share



Run

Output

Clear



```
34     }
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
44         .deposit(1000));
45     Thread depositThread2 = new Thread() -> account
46         .deposit(300));
47     Thread withdrawalThread1 = new Thread() ->
48         account.withdraw(150));
49     Thread withdrawalThread2 = new Thread() ->
50         account.withdraw(1200));
51
52     depositThread1.start();
53     depositThread2.start();
54     withdrawalThread1.start();
55     withdrawalThread2.start();
56 }
```

```
Deposito: 1000.0
Saldo despues del deposito; 1000.0
Deposito: 300.0
Saldo despues del deposito; 1300.0
Retiro: 1200.0
Saldo despues del retiro:100.0
Intento de retiro:150.0
Saldo insuficiente, retiro cancelado.
```

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



35°C
Mayorm. soleado



Buscar



ENG
US

02:47 p. m.
10/08/2025

Explicacion del código

Balance: representa el saldo de la cuenta.

- ReentrantLock: asegura que solo un hilo acceda al saldo a la vez.
- deposit(double amount): suma dinero al saldo.
- withdraw(double amount): resta dinero si hay suficiente saldo

Se crean 4 hilos:

- 2 para depositar: \$1000 y \$300.
- 2 para retirar: \$150 y \$1200.
- Todos se ejecutan simultáneamente, pero el Lock garantiza que las operaciones sean seguras y ordenadas.