```
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();
}
}
```

Output

```
Deposito:  1000.0
Saldo despues del deposito; 1000.0
Deposito:  300.0
Saldo despues del deposito; 1300.0
Retiro: 150.0
Saldo despues del retiro:1150.0
Intento de retiro:1200.0
Saldo insuficiente, retiro cancelado.

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