Task 4: Synchronized Blocks and Methods
Write a program that simulates a bank account being accessed by multiple threads to perform deposits and withdrawals using synchronized methods to prevent race conditions.
ANS:

```
package com.Day23;
public class Task4 {
  private double balance;
  public Task4(double initialBalance) {
     this.balance = initialBalance;
  }
  public synchronized void deposit(double amount) {
     balance += amount;
     System.out.println(Thread.currentThread().getName() + " deposited " + amount + ", balance: " +
balance);
  }
  public synchronized void withdraw(double amount) {
     if (balance >= amount) {
        balance -= amount;
        System.out.println(Thread.currentThread().getName() + " withdrew " + amount + ", balance: " +
balance);
     } else {
        System.out.println(Thread.currentThread().getName() + " - Insufficient funds for withdrawal");
     }
  }
  public static void main(String[] args) {
     Task4 account = new Task4(1000);
     Thread[] threads = new Thread[5];
     for (int i = 0; i < threads.length; i++) {
        threads[i] = new Thread(() -> {
           for (int j = 0; j < 3; j++) {
              account.deposit(100);
              account.withdraw(200);
           }
        });
        threads[i].start();
     }
     // Wait for all threads to finish
     for (Thread thread : threads) {
        try {
           thread.join();
        } catch (InterruptedException e) {
           e.printStackTrace();
        }
     }
     System.out.println("Final balance: " + account.balance);
  }
}
```