

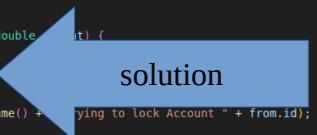
Report: Deadlock Simulation & Solution

Part 1: Deadlock Simulation

The screenshot shows a Java development environment with two open files: `DeadlockSimulation1.java` and `DeadlockSimulation2.java`. The code implements a simple banking system with accounts and a transfer function. It uses `Semaphore` to manage locks on accounts. The terminal window below shows the execution of the code, where two threads are attempting to lock accounts simultaneously, leading to a deadlock.

```
1 import java.util.concurrent.Semaphore;
2
3 class Account {
4     public int id;
5     public double balance;
6     public Semaphore lock = new Semaphore(permits: 1);
7
8     public Account(int id, double balance) {
9         this.id = id;
10        this.balance = balance;
11    }
12 }
13
14 public class DeadlockSimulation1 {
15
16     public static void transfer(Account from, Account to, double amount) {
17         try {
18             System.out.println(Thread.currentThread().getName() + " trying to lock Account " + from.id);
19             from.lock.acquire();
20             System.out.println(Thread.currentThread().getName() + " locked Account " + from.id);
21
22             System.out.println(Thread.currentThread().getName() + " processing... (holding lock on " + from.id + ")");
23             Thread.sleep(millis: 1000);
24
25             System.out.println(Thread.currentThread().getName() + " trying to lock Account " + to.id);
26             to.lock.acquire();
27
28             System.out.println(Thread.currentThread().getName() + " locked Account " + to.id);
29
30             if (from.balance >= amount) {
31                 from.balance -= amount;
32                 to.balance += amount;
33                 System.out.println("Success! Transferred " + amount + " from " + from.id + " to " + to.id);
34             } else {
35                 System.out.println("Insufficient funds for " + Thread.currentThread().getName());
36             }
37
38             to.lock.release();
39             from.lock.release();
40
41         } catch (InterruptedException e) {
42
43     }
44 }
45
46 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Activity\ 7 ; /usr/bin/env /usr/lib/jvm/jdk-25.0.1-oracle-x64/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /home/dararith/.config/Code/User/workspaceStorage/60bf605d48194d21dddec1c02fa863b8/redhat.java/jdt_ws/Class\ Activity\ 7 743ee8d3/bin DeadlockSimulation1
(base) dararith@dararith-ASUS-Vivobook-S-14:~/Documents/Operating\ System/Class\ Activity\ Class\ Activity\ 7$ cd /home/dararith/Documents/Operating\ System/Class\ Activity\ Class\ Activity\ 7/743ee8d3/bin DeadlockSimulation1
Activity\ 7 ; /usr/bin/env /usr/lib/jvm/jdk-25.0.1-oracle-x64/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /home/dararith/.config/Code/User/workspaceStorage/60bf605d48194d21dddec1c02fa863b8/redhat.java/jdt_ws/Class\ Activity\ 7 743ee8d3/bin DeadlockSimulation1
Thread-0 trying to lock Account 2
Thread-0 trying to lock Account 1
Thread-1 locked Account 1
Thread-1 locked Account 2
Thread-0 processing... (holding lock on 1)
Thread-1 processing... (holding lock on 2)
Thread-1 trying to lock Account 1
Thread-0 trying to lock Account 2
[]
```

Part 2: Deadlock Solution (Lock Ordering)



```
1 import java.util.concurrent.Semaphore;
2
3 class Account {
4     public int id;
5     public double balance;
6     public Semaphore lock = new Semaphore(permits: 1);
7
8     public Account(int id, double balance) {
9         this.id = id;
10        this.balance = balance;
11    }
12 }
13
14 public class DeadlockSimulation2 {
15
16     public static void transfer(Account from, Account to, double amount) {
17
18         Account firstLock = from.id < to.id ? from : to;
19         Account secondLock = from.id < to.id ? to : from;
20
21         try {
22             System.out.println(Thread.currentThread().getName() + " trying to lock Account " + from.id);
23             firstLock.lock.acquire();
24             System.out.println(Thread.currentThread().getName() + " locked Account " + from.id);
25
26             System.out.println(Thread.currentThread().getName() + " processing... (holding lock on " + from.id + ")");
27             Thread.sleep(1000);
28
29             System.out.println(Thread.currentThread().getName() + " trying to lock Account " + to.id);
30             secondLock.lock.acquire();
31
32             System.out.println(Thread.currentThread().getName() + " locked Account " + to.id);
33
34             if (from.balance >= amount) {
35                 from.balance -= amount;
36                 to.balance += amount;
37                 System.out.println("Success! Transferred " + amount + " from " + from.id + " to " + to.id);
38             } else {
39                 System.out.println("Insufficient funds for " + Thread.currentThread().getName());
40             }
41
42             secondLock.lock.release();
43
44         } catch (InterruptedException e) {
45             e.printStackTrace();
46         }
47     }
48 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) dararith@dararith-ASUS-Vivobook-S-14:~/Documents/Operating System/Class Activity/Class Activity 7$ cd /home/darith/Downloads/Operating System/Class Activity/Class Activity 7 ; /usr/bin/env /usr/lib/jvm/jdk-25.0.1-oracle-x64/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /home/darith/.config/Code/User/workspaceStorage/68bf690de48194d21dddec1c62fa863d8/redhat.java/jdt ws/ClassActivity/7 7436e8d3/bin DeadlockSimulation2
Thread-1 trying to lock Account 1
Thread-1 locked Account 1
Success! Transferred 100.0 from 2 to 1
Thread-0 locked Account 1
Thread-0 processing... (holding lock on 1)
Thread-0 trying to lock Account 2
Thread-0 locked Account 2
Success! Transferred 100.0 from 1 to 2
Transfer Successfully
(base) dararith@dararith-ASUS-Vivobook-S-14:~/Documents/Operating System/Class Activity/Class Activity 7$ 
```

Run: DeadlockSimulation2 Run: DeadlockSimulation2 Run: DeadlockSimulation2

Dararith (1 minute ago) Ln 12, Col 2 Spaces: 4 UTF-8 LF () Java Finish Setup Prettier