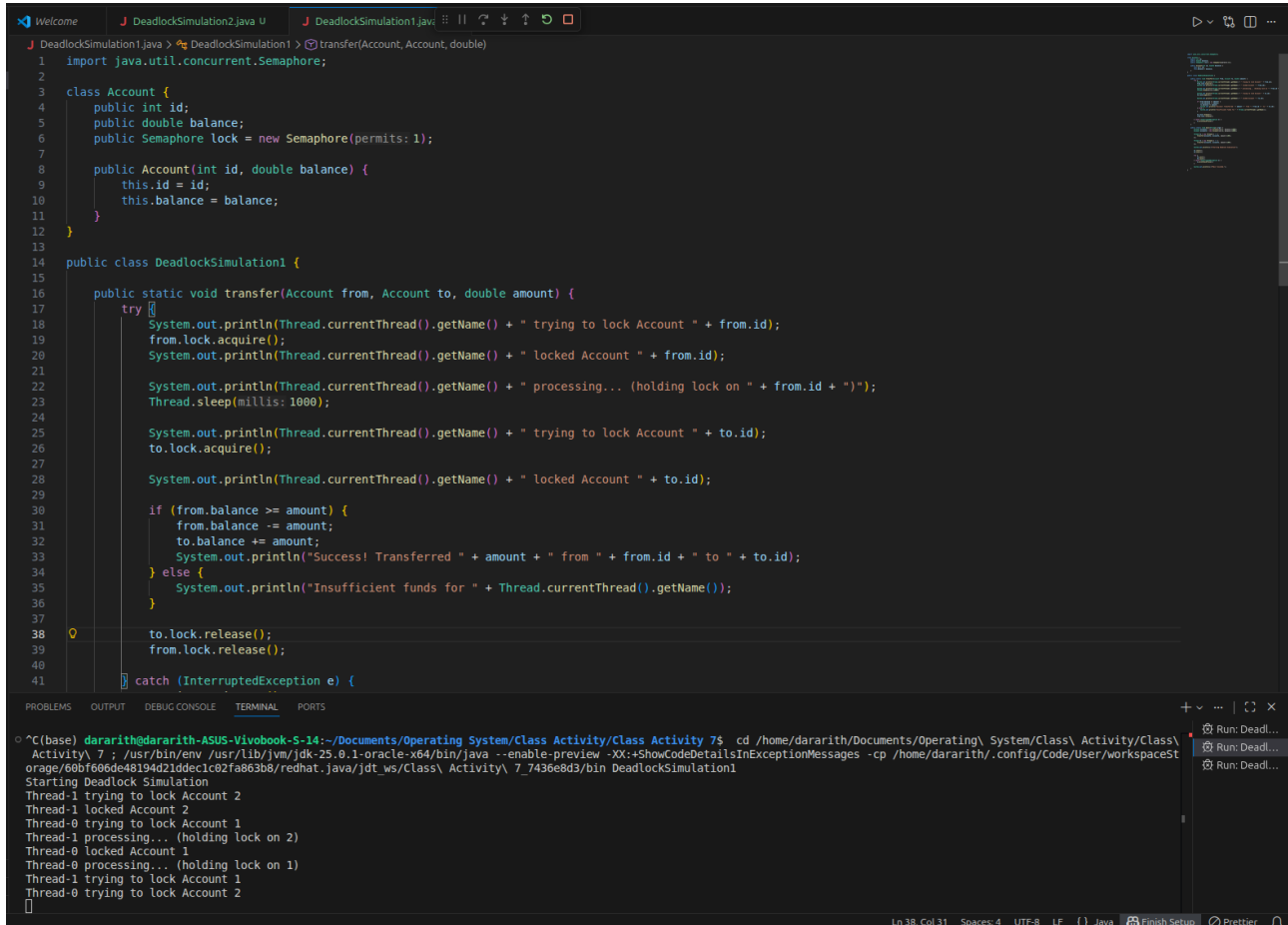


Report: Deadlock Simulation & Solution

Part 1: Deadlock Simulation



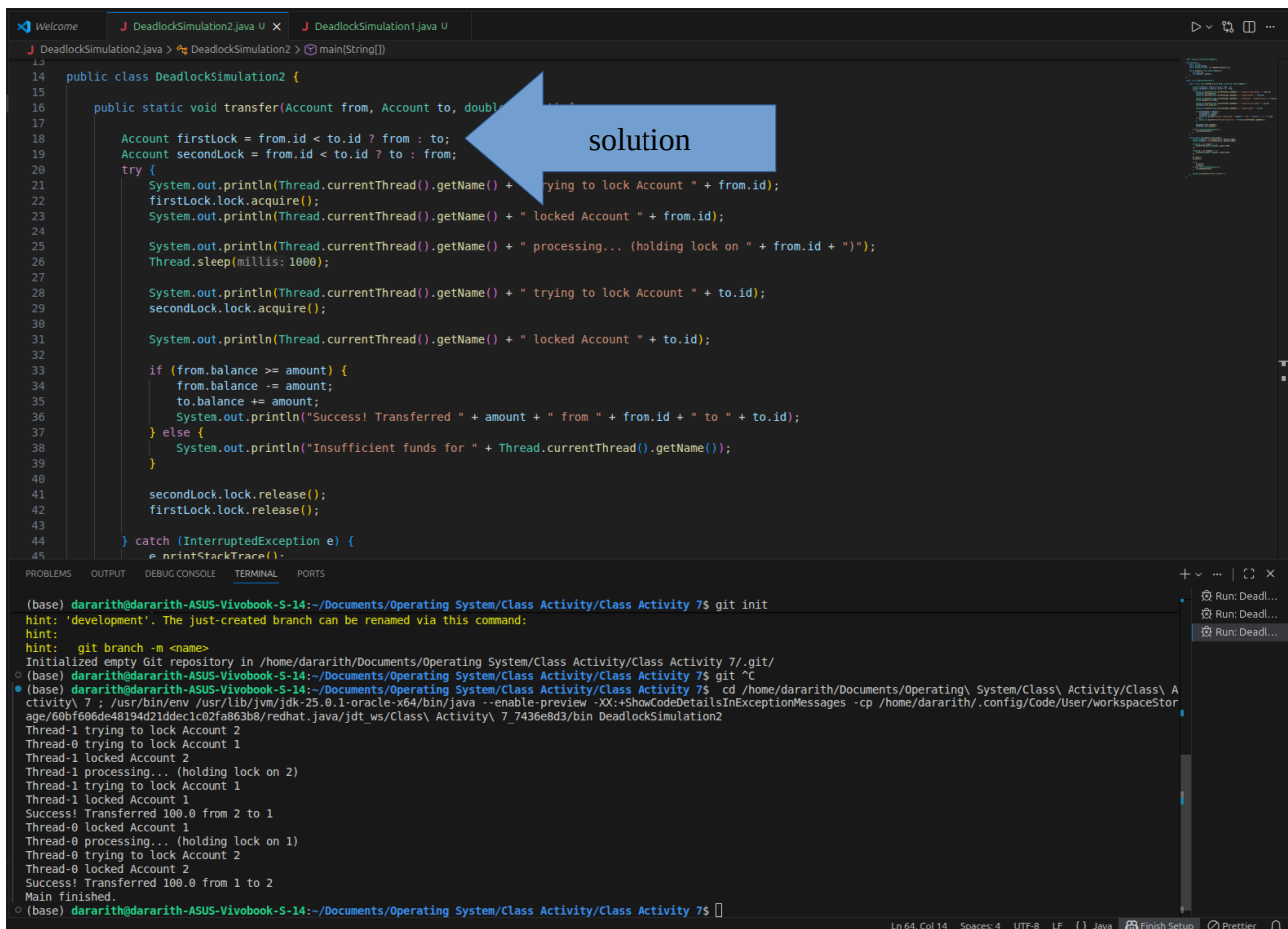
The screenshot displays an IDE with a Java file named `DeadlockSimulation1.java`. The code defines an `Account` class with a `lock` semaphore and a `transfer` method. The `transfer` method attempts to acquire locks on two accounts, `from` and `to`, to transfer an amount. It includes a sleep to simulate processing time. The `main` method calls `transfer` with two accounts and an amount of 10.

```
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 }
```

The terminal output shows the execution of the simulation. It starts with "Starting Deadlock Simulation". Thread-1 attempts to lock Account 2, succeeds, and then attempts to lock Account 1. Thread-0 attempts to lock Account 1, succeeds, and then attempts to lock Account 2. Both threads are now holding locks on each other's accounts, leading to a deadlock state where neither can complete the transfer.

```
Starting Deadlock Simulation
Thread-1 trying to lock Account 2
Thread-1 locked Account 2
Thread-0 trying to lock Account 1
Thread-0 locked Account 1
Thread-1 processing... (holding lock on 2)
Thread-0 processing... (holding lock on 1)
Thread-1 trying to lock Account 1
Thread-0 trying to lock Account 2
```

Part 2: Deadlock Solution (Lock Ordering)



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

terminal

```
(base) dararith@dararith-ASUS-Vivobook-S-14:~/Documents/Operating System/Class Activity/Class Activity 7$ git init
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint: git branch -m <name>
Initialized empty Git repository in /home/dararith/Documents/Operating System/Class Activity/Class Activity 7/.git/
(base) dararith@dararith-ASUS-Vivobook-S-14:~/Documents/Operating System/Class Activity/Class Activity 7$ git ^C
(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 ; /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/60b6f606de48194d21d0c1c02fa863b8/redhat.java/jdt_ws/Class Activity\ 7_7436e8d3/bin DeadlocksSimulation2
Thread-1 trying to lock Account 2
Thread-0 locked Account 1
Thread-1 locked Account 2
Thread-1 processing... (holding lock on 2)
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
Main finished.
(base) dararith@dararith-ASUS-Vivobook-S-14:~/Documents/Operating System/Class Activity/Class Activity 7$
```