Java y Servicios Web I
Máster en Ingeniería Matemática

Hoja de Ejercicios – Tema 9
29 de Mayo de 2012

Ejercicio 9.1 (0.5 puntos)

Retomemos la clase del Ejercicio 4.2 para manejar cuentas bancarias:

```
public class CuentaBancaria {
  private int saldo = 0;

  public CuentaBancaria() { saldo = 0; }
  public void ingresarDinero(int cantidad) { … }
  public void retirarDinero(int cantidad) { … }

  public void transferirDinero(int cantidad, CuentaBancaria destino) {
   this.retirarDinero(cantidad);
   destino.ingresarDinero(cantidad);
  }
}
```

Crear una clase `Transferidor`, que implemente la interfaz `Runnable`, y que contenga dos atributos: `origen` y `destino`, ambos objetos de la clase `CuentaBancaria`. El método `run()` de cada transferidor ha de realizar un número grande de transferencias de 1€ desde la cuenta origen hacia la cuenta destino (por ejemplo, 100 000 transferencias).

El programa principal ha de crear dos cuentas, A y B, y dos transferidores, que se han de ejecutar simultáneamente en hilos distintos. El primero de ellos transferirá dinero de A a B, y el segundo de ellos lo hará de B a A. El programa principal habrá de esperar a que ambos transferidores terminen (mediante el método `join` de la clase `Thread`) y mostrar los saldos disponibles en cada cuenta una vez que los transferidores han terminado.

Ejecuta el programa. ¿Acaban ambas cuentas con saldo cero? ¿Por qué no?

Ahora declara los métodos `ingresarDinero`, `retirarDinero` y `transferirDinero` como `synchronized`. ¿Soluciona esto el problema? ¿Qué ha pasado?

Indicación: Se trata de un problema de *interbloqueo*. Los siguientes enlaces pueden ayudarte a explicar lo sucedido:

http://es.wikipedia.org/wiki/Bloqueo_mutuo
http://docs.oracle.com/javase/tutorial/essential/concurrency/deadlock.html