# Threads Homework Part II

Due Friday, April 18, 2008 at 9:00am in class

1. Variable balance is a static class variable of type AtomicInteger. Why is the following code **not** properly synchronized?

| // Thread 1: | // Thread 2: |
|---|---|
| int x = balance.get(); | int y = balance.get(); |
| x = x + 100; | y =y – 100; |
| balance.set(x); | balance.set(y); |

2. Variable balance is a static class variable of type AtomicInteger. Why is the following code **not** properly synchronized?

| // Thread 1: | // Thread 2: |
|---|---|
| int x = balance.get(); | int y = balance.get(); |
| synchronized (balance) { x = x + 100; } | synchronized (balance) { y =y – 100; } |
| balance.set(x); | balance.set(y); |

3. Variable balance is a static class variable of type **Integer**. Why is the following code **not** properly synchronized?

| // Thread 1: | // Thread 2: |
|---|---|
| synchronized (balance) { | synchronized (balance) { |
| int x = balance.intValue(); | int y = balance.intValue(); |
| x = x + 100; | y =y – 100; |
| balance = new Integer (x); | balance = new Integer(y); |
| } | } |

4. MyInteger is a class that stores a single integer accessible through get and set methods that are not synchronized. What bad situation could arise when the following code is run? (This is not something we've discussed yet…) Fix the code so that situation does not arise.

```
public static MyInteger balance0 = new MyInteger(100);
public static MyInteger balance1 = new MyInteger(40);
```

```
// Thread 1: (money transfer)
synchronized (balance1) {
   synchronized (balance 0) {
     balance1.set( balance0.get() +
                    balance1.get());
     balance0.set(0);
   }
}
```

```
// Thread 2: (find larger account)
synchronized (balance0) {
   synchronized (balance1) {
     if (balance0.get() > balance1.get()) {
          System.out.println("Person 0 is richer");
     }
   }
}
```