Chelsea Zackey
Erik Rosales
Albert Belardino

**Lab 5: Task 1**

Race conditions occur within the methods of the Account class, namely for withdraw() and deposit(). If two threads both called either of these functions for the same account, then they would both see the global variable “balance” of that account. However, say thread1 saw the current balance, and prior to adjusting this balance, thread2 is scheduled to see the current balance. Then whichever thread is scheduled to finish the method last is the thread that ultimately adjusts the global variable “balance”, as if the first completed transaction never happened (since each thread adjusted the same global variable).

Moreover, race conditions occur whenever we call the Bank class methods shouldTest() and test(), since these methods are simultaneously called within each of the threads associated with each of the given accounts. Namely, if multiple threads are calling test(), which calls the Account method getBalance(), it is possible that other threads are simultaneously performing their own transactions on that specific account (and changing the balance), thus compromising the result of our test.