Exercise Playing with coroutines

The following program has two co-operating coroutines:

```
class MainActivity : AppCompatActivity() {
   class Account {
       private var amount: Double = 0.0

       suspend fun deposit_coroutine(amount: Double) {
           val x = this.amount
           delay(1)    // simulates processing time
           this.amount = x + amount
       }

       fun saldo(): Double = amount
   }

   /* Approximate measurement of the given block's execution time */
   fun withTimeMeasurement(title:String, isActive:Boolean=true, code:() -> Unit) {
       if(!isActive) return

       val timeStart=System.currentTimeMillis()
       code()
       val timeEnd=System.currentTimeMillis()

       println("operation in '$title' took ${(timeEnd- timeStart)} ms")
   }

   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
       setContentView(R.layout.activity_main)

       val tili2 = Account();
       withTimeMeasurement("Single coroutine deposit 1000 times") {
           runBlocking {
               launch {
                   for (i in 1..1000)
                       tili2.deposit_coroutine(0.0)
               }
           }
           findViewById<TextView>(R.id.idSaldo2).text = "Saldo2: ${tili2.saldo()}"
       }

       withTimeMeasurement("Two coroutines together", isActive = true) {
           runBlocking {
               launch {
                   for (i in 1..1000) tili2.deposit_coroutine(1.0)
               }
               launch {
                   for (i in 1..1000) tili2.deposit_coroutine(1.0)
               }
               findViewById<TextView>(R.id.idSaldo2).text = "Saldo2: ${tili2.saldo()}"
           }
       }
   }
}
```

runBlocking {} is a coroutine function. By not providing any context, it will get run on the main thread. Runs a new coroutine and blocks the current thread interruptible until its completion. This function should not be used from a coroutine. It is designed to bridge regular blocking code to libraries that are written in suspending style, to be used in main functions and in tests. Note that it is not recommended to use in Android, because it will block the UI thread. In this case we use it only to allow easy execution time measurements.

launch {} is a coroutine builder. It launches a new coroutine concurrently with the rest of the code, which continues to work independently.

There is a problem in the given code. Two coroutines increase the value (saldo) of the account by 1000. Therefore the R.id.idSaldo TextField should have the value of 2000. But when you run the code, the value will be 0 or 1000.

Fix the program, so that the final saldo of the account will be 2000, and you are still using those two independent coroutines (and the deposit() operation reads the saldo to the variable x, delays 1ms, and the writes the incremented value back to the saldo). Do not use withContex() in your solution.

Hint 1: Remember to add the Kotlin coroutine library to the build.gradle.

Hint 2: There are two separate errors in the program.

Hint 3: Mutex may be a useful tool.