Labs

Lab 3.1: Synchronizing Threads

What is the purpose?

In this lab, you will write a program that launches 1,000 threads. Each thread adds 1 to a variable sum that is initially 0. You need to pass sum by reference to each thread. In order to pass it by reference, define an Integer wrapper object to hold the sum. Run the program with and without synchronization to see its effect.

Here is a sample of the output without synchronization:

```
At thread 1 sum = 0,
At thread 2 sum = 1,
At thread 3 sum = 2,
At thread 4 sum = 2,
At thread 5 sum = 4,
At thread 6 sum = 5,
At thread 7 sum = 6,
At thread 8 sum = 6,
At thread 9 sum = 8,
At thread 10 sum = 9,
At thread 11 sum = 10,
At thread 12 sum = 10,
……
……
At thread 995 sum = 991,
At thread 996 sum = 992,
At thread 997 sum = 993,
At thread 998 sum = 994,
At thread 999 sum = 995,
At thread 1000 sum = 996,
What is sum? 997
```

Figure 3-1-1: Sample Output 1

Here is a sample of the output *with* synchronization:

```
At thread 1 sum = 0,
At thread 2 sum = 1,
At thread 3 sum = 2,
At thread 4 sum = 3,
At thread 5 sum = 4,
At thread 6 sum = 5,
At thread 7 sum = 6,
At thread 8 sum = 7,
At thread 9 sum = 8,
At thread 10 sum = 9,
At thread 11 sum = 10,
At thread 12 sum = 11,
……
……
At thread 992 sum = 991,
```

```
At thread 993 sum = 992,
At thread 994 sum = 993,
At thread 995 sum = 994,
At thread 996 sum = 995,
At thread 997 sum = 996,
At thread 998 sum = 997,
At thread 999 sum = 998,
At thread 1000 sum = 999,
What is sum? 1000
```

Figure 3-1-2: Sample Output 2

What are the steps?

- **Task 1**

Procedure:

1. Create a java file named Thread.java that develops 1,000 threads, each of which adds 1 to a variable sum.
2. Complete the following code or create your own code:

```
import java.util.concurrent.*;
public class Threads {
private Integer sum = new Integer(0);
public static void main(String[] args) {

  Threads test = new Threads();
  System.out.println("What is sum ? " + test.sum);
}
public Threads() {
  ExecutorService executor = Executors.newFixedThreadPool(1000);
  ……..
  executor.shutdown();
  while(!executor.isTerminated()) {

  }
}

// without synchronization
class SumTask implements Runnable {
  public void run() {
    ……..
  }

}
```

Figure 3-1-3

3. Create a java file named Thread2.java that develops 1,000 threads with synchronization, each of which adds 1 to a variable sum.
4. Complete the following code or create your own code:

```
import java.util.concurrent.*;
public class Threads2 {
private Integer sum = new Integer(0);
public synchronized static void main(String[] args) {
```

```
   Threads2 test = new Threads2();

}
public Threads2() {
   ExecutorService executor = Executors.newFixedThreadPool(1000);
   ……..


  }
   executor.shutdown();
   while(!executor.isTerminated()) {


   }
}

// with synchronization
class SumTask2 implements Runnable {
   public synchronized void run() {
    ……..


  }
}
```

Figure 3-1-4

5. Submit your Java code and sample output to your instructor.

Did it work?

Were you able to:

- Create a Java program that can produce 1000 threads that adds 1 to a variable sum without synchronization?
- Create another Java program that can produce 1000 threads that adds 1 to a variable sum with synchronization?

Lab 3.2: Account Synchronization

What is the purpose?

In this lab, you will study Listing 29.9 on pages 950-951 of your textbook. Rewrite ThreadCooperation.java, using the wait() and notifyAll() methods of the object.

Here is a sample of the output:

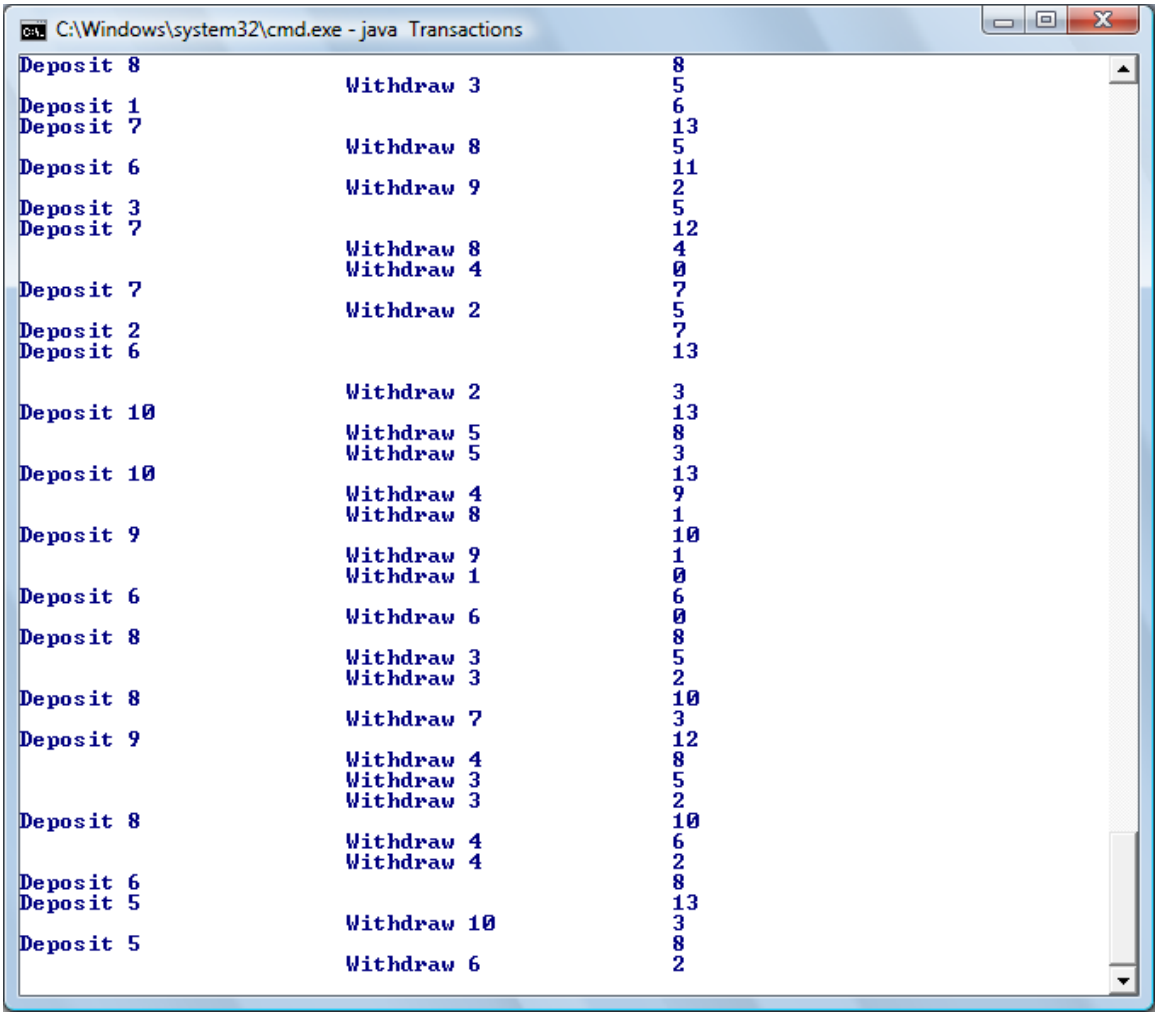

Figure 3-2-1

What are the steps?

- **Task 1**

  Procedure:

  1. Modify the code for ThreadCooperation.java on pages 950-951 or create your own code.
  2. Add the wait() method in the public synchronized void withdraw(int amount) to cause the current thread to wait until another thread invokes the notifyAll() method for the object account.
  3. Add the notifyAll() method in the public synchronized void deposit(int amount) to wake up all the threads that are waiting on the object account.
  4. Re-run the ThreadCooperation class to see the difference.
  5. Submit your Java code and sample output to your instructor.

Did it work?

Were you able to:

- Rewrite ThreadCooperation.java by using wait() method instead of await() method?
- Rewrite ThreadCooperation.java by using notifyAll() method instead of signalAll() method?