Write a multithreaded program that implements the ***banker’s algorithm*** discussed in class and Section 7.5.3. Create *n* threads that request and release resources from the bank. The banker will grant the request only if it leaves the system in a safe state. This program should use Java threads.

The following programs are included in the package (You *only* need to complete **BankImpl** class):

1. class **SleepUtilities**: utilities for causing a thread to sleep.
2. interface **Bank**: This is the interface for BankImpl class.
3. class **BankImpl**: **This is the only program you need to complete.**

This class implements the Bank interface. It provides methods needed in Banker’s algorithms. **You need to fill in the blanks in the following methods:**

private boolean isSafeState (int threadNum, int[] request)

public synchronized boolean requestResources(int threadNum, int[] request)

public synchronized void releaseResources(int threadNum, int[] release)

Note: need is always maximum ***minus*** allocation, including in the realeaseResources method.

1. class **Customer**: In this implementation, customers are “processes” in the description of Banker’s algorithm. A customer requests, uses, and then releases resources. From the field of this class, you can see that we assume 5 customers (threads).
2. class **Factory**: this is the main class that you should run from.

To run the application, use command:

java Factory <one or more resources>

e.g. java Factory 10 5 7

The package also includes an input file (infile.txt). It keeps a record the maximum resources that each customer demands. The Factory class reads from this file.