Bushra Hameed

**Project 2 Design**

**List of Semaphores:**

1. **public** **static** Semaphore *maxCustomers* = **new** Semaphore(5, **true**);
2. **public** **static** Semaphore *queueNotEmpty* = **new** Semaphore(0, **true**);
3. **public** **static** Semaphore *queue1NotEmpty* = **new** Semaphore(0, **true**);
4. **public** **static** Semaphore[] *banktellerRequest* = **new** Semaphore[] {**new** Semaphore(0), **new** Semaphore(0)};
5. **public** **static** Semaphore[] *customerDepositReceipt* = **new** Semaphore[] {**new** Semaphore(0), **new** Semaphore(0)};
6. **public** **static** Semaphore[] *customerDepositComplete* = **new** Semaphore[] {**new** Semaphore(0), **new** Semaphore(0)};
7. **public** **static** Semaphore[] *withdrawalReceipt* = **new** Semaphore[] {**new** Semaphore(0), **new** Semaphore(0)};
8. **public** **static** Semaphore[] *withdrawalComplete* = **new** Semaphore[] {**new** Semaphore(0), **new** Semaphore(0)};
9. **public** **static** Semaphore[] *tellerReady* = **new** Semaphore[]{ **new** Semaphore(0), **new** Semaphore(0), **new** Semaphore(0), **new** Semaphore(0),**new** Semaphore(0)};
10. **public** **static** Semaphore[] *loanOfficerReady* = **new** Semaphore[]{ **new** Semaphore(0), **new** Semaphore(0), **new** Semaphore(0), **new** Semaphore(0),**new** Semaphore(0)};
11. **public** **static** Semaphore *loanOfficerRequest* = **new** Semaphore(0, **true**);
12. **public** **static** Semaphore *loanOfficerReceipt* = **new** Semaphore(0, **true**);
13. **public** **static** Semaphore *loanTransactionComplete* = **new** Semaphore(0, **true**);
14. **public** **static** Semaphore *mutex*= **new** Semaphore(1, **true**);
15. **public** **static** Semaphore *mutex1*= **new** Semaphore(1, **true**);

**Purpose of Each Semaphore:**

1. max number of customers
2. signify that the first queue is not empty
3. signify that the second queue is not empty
4. signifies the bank teller request
5. signifies the customer deposit receipt
6. signifies that the customer’s deposit is complete
7. signifies the withdrawal receipt
8. signifies that the withdrawal is complete
9. signifies that the teller is ready
10. signifies that the Loan Officer is ready
11. signifies the Loan Officer request has been made
12. signifies the Loan Officer receipt
13. signifies that loan transaction is complete
14. ensures only one thread can execute in the critical section
15. ensures only one thread can execute in the critical section

**Pseudocode:**

public class Customer implements Runnable{

//list of global semaphores

//list of global queues

//list of global arrays

Customer(int num){

//initialize customer instances

}

Public void run(){

for() //for loop runs up to 3 times {

try{

//limit the number of max threads

if( task is equal to 1) {

process customer deposit

move onto the next customer

}

if(task is equal to 2) {

a withdrawal is made

}

if(task is equal to 3) {

a loan is being processed

}

catch {

interrupt

}

//tasks are randomly assigned

private int randTaskAssign() {

//use random function

}

public static void main(String args[]) {

//list of variables

Loan Officer thread – use for loop

Bank Teller thread – use for loop

Customers thread – use for loop

Join Customers thread – use for loop

Print summary

}

//bank teller class implements the thread

class Bankteller implements Runnable{

//initialize variables

//call Bankteller()

Bankteller(int num) {}

public void run() {

while loop set to true {

try block {

if statements to handle deposits and withdrawals

}

catch {interrupts}

}

}

}

}

//Loan Officer implements the thread

class LoanOfficer implements Runnable {

//list of variables

public void run() {

while loop set to true {

try block {

if block to execute loan processing

}

catch (interrupt) {

//print

}

}

}

}