//package com.concurrency;

**import** java.util.concurrent.Semaphore;

**import** java.util.concurrent.locks.Lock;

**import** java.util.concurrent.locks.ReentrantLock;

**public** **class** CigaretteSmoker {

/\*\*

\* Boolean variables indicate whether or not an ingredient is on the table.

\*/

**boolean** isTobacco = **false**;

**boolean** isPaper = **false**;

**boolean** isMatch = **false**;

/\*\*

\* The pushers use tobaccoSem to signal the smoker with tobacco, and the other semaphores likewise.

\*/

**public** **static** Semaphore *tobaccoSem* = **new** Semaphore(0);

**public** **static** Semaphore *paperSem* = **new** Semaphore(0);

**public** **static** Semaphore *matchSem* = **new** Semaphore(0);

/\*\*

\* Semaphore for signaling ingredients are available

\*/

**public** **static** Semaphore *tobacco* = **new** Semaphore(0);

**public** **static** Semaphore *paper* = **new** Semaphore(0);

**public** **static** Semaphore *match* = **new** Semaphore(0);

/\*\*

\* All the agents will wait on agentSem and each time agentSem is signaled, one

\* of the Agents wakes up and provides ingredients by signaling two semaphores

\*/

**public** **static** Semaphore *agentSem* = **new** Semaphore(1);

**public** **static** Lock *mutex* = **new** ReentrantLock();

/\*\*

\* This method will initiate all the 3 Pushers

\*

\* Description from http://www.greenteapress.com/semaphores/downey05semaphores.pdf

\*

\* The solution proposed by Parnas uses three helper threads called “pushers” that

\* respond to the signals from the agent, keep track of the available ingredients,

\* and signal the appropriate smoker.

\*/

**public** **void** initPushers() {

Thread pusherA = **new** Thread() {

**public** **void** run() {

**while**(**true**) {

**try** {

*tobacco*.acquire();

System.***out***.println("Pusher A for tobacco is active");

*mutex*.lock();

**try** {

**if**(isPaper) {

isPaper = **false**;

*matchSem*.release();

} **else** **if**(isMatch) {

isMatch = **false**;

*paperSem*.release();

} **else** {

isTobacco = **true**;

}

} **finally** {

*mutex*.unlock();

}

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

};

};

Thread pusherB = **new** Thread() {

**public** **void** run() {

**while**(**true**) {

**try** {

*paper*.acquire();

System.***out***.println("Pusher B for Paper is active");

*mutex*.lock();

**try** {

**if**(isTobacco) {

isTobacco = **false**;

*matchSem*.release();

} **else** **if**(isMatch) {

isMatch = **false**;

*tobaccoSem*.release();

} **else** {

isPaper = **true**;

}

} **finally** {

*mutex*.unlock();

}

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

};

};

Thread pusherC = **new** Thread() {

**public** **void** run() {

**while**(**true**) {

**try** {

*match*.acquire();

System.***out***.println("Pusher C for Match is active");

*mutex*.lock();

**try** {

**if**(isPaper) {

isPaper = **false**;

*tobaccoSem*.release();

} **else** **if**(isTobacco) {

isTobacco = **false**;

*paperSem*.release();

} **else** {

isMatch = **true**;

}

} **finally** {

*mutex*.unlock();

}

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

};

};

pusherA.start();

pusherB.start();

pusherC.start();

}

/\*\*

\* This method will initialize all the 3 smokers. Smoker will perform following task:

\*

\* 1> Try to acquire the ingredient semaphore so that smoker can start only when the necessary ingredients are present. This will be signaled by the Pushers.

\* 2> Make Cigarette

\* 3> Release the agentSem semaphore so that Agent can place the ingredients again on the table.

\* 4> Start smoking

\*/

**public** **void** initSmokers() {

Thread tobaccoSmoker = **new** Thread() {

@Override

**public** **void** run() {

**while**(**true**) {

**try** {

*tobaccoSem*.acquire();

makeCigarette();

*agentSem*.release();

smoke();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

**public** **void** makeCigarette() {

System.***out***.println("tobaccoSmoker is making cigratte");

**try** {

*sleep*(5000);

} **catch** (InterruptedException ex) {

}

System.***out***.println("tobaccoSmoker is cigratte making completed");

}

**public** **void** smoke() {

System.***out***.println("tobaccoSmoker is smoking");

**try** {

*sleep*(5000);

} **catch** (InterruptedException ex) {

}

}

};

Thread matchSmoker = **new** Thread() {

@Override

**public** **void** run() {

**while**(**true**) {

**try** {

*matchSem*.acquire();

makeCigarette();

*agentSem*.release();

smoke();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

**public** **void** makeCigarette() {

System.***out***.println("matchSmoker is making cigratte");

**try** {

*sleep*(5000);

} **catch** (InterruptedException ex) {

}

System.***out***.println("matchSmoker is cigratte making completed");

}

**public** **void** smoke() {

System.***out***.println("matchSmoker is smoking");

**try** {

*sleep*(5000);

} **catch** (InterruptedException ex) {

}

}

};

Thread paperSmoker = **new** Thread() {

@Override

**public** **void** run() {

**while**(**true**) {

**try** {

*paperSem*.acquire();

makeCigarette();

*agentSem*.release();

smoke();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

**public** **void** makeCigarette() {

System.***out***.println("paperSmoker is making cigratte");

**try** {

*sleep*(5000);

} **catch** (InterruptedException ex) {

}

System.***out***.println("paperSmoker is cigratte making completed");

}

**public** **void** smoke() {

System.***out***.println("paperSmoker is smoking");

**try** {

*sleep*(5000);

} **catch** (InterruptedException ex) {

}

}

};

tobaccoSmoker.start();

matchSmoker.start();

paperSmoker.start();

}

/\*\*

\* This method will initialize all the 3 agents. Agents will perform following task:

\*

\* 1> Try to acquire agentSem Semaphore so that they release the ingredients

\* 2> Places the ingredients on the table. This is done by releasing the respective ingredients semaphore thereby signaling the Pushers to takeover.

\*/

**public** **void** initAgents() {

Thread agentA = **new** Thread() {

@Override

**public** **void** run() {

**while**(**true**) {

**try** {

*agentSem*.acquire();

System.***out***.println("Agent A is active and will release provide Tobacco & Paper ingredients.");

*tobacco*.release();

*paper*.release();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

};

Thread agentB = **new** Thread() {

@Override

**public** **void** run() {

**while**(**true**) {

**try** {

*agentSem*.acquire();

System.***out***.println("Agent B is active and will release provide Match & Paper ingredients.");

*match*.release();

*paper*.release();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

};

Thread agentC = **new** Thread() {

@Override

**public** **void** run() {

**while**(**true**) {

**try** {

*agentSem*.acquire();

System.***out***.println("Agent C is active and will release provide Tobacco & Match ingredients.");

*tobacco*.release();

*match*.release();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

};

agentA.start();

agentB.start();

agentC.start();

}

**public** **static** **void** main(String[] args) {

CigaretteSmoker cs = **new** CigaretteSmoker();

cs.initAgents();

cs.initPushers();

cs.initSmokers();

}

}