**package** com.example.java;  
  
  
**import** java.io.File;  
**import** java.util.Collections;  
**import** java.util.Comparator;  
**import** java.util.LinkedList;  
**import** java.util.Scanner;  
  
**public class** Main {  
 **public static void** main(String[] args) **throws** Exception {  
 System.***out***.println(**"\*\*\* Welcome to Burrito Brothers \*\*\*"** + **"\n"**);  
  
 **int** custId;  
 **int** numBurritos;  
  
 *//Creating a linked list to hold waiting customers* LinkedList<Customer> customersWaiting = **new** LinkedList<Customer>();  
  
 *//Obtaining customers from text file and storing their information in string array* Scanner fileToRead = **new** Scanner(**new** File(**"customers.txt"**));  
 **while** (fileToRead.hasNext()) {  
 String[] line = fileToRead.nextLine().split(**" "**);  
 custId = Integer.*parseInt*(line[0]);  
 numBurritos = Integer.*parseInt*(line[1]);  
  
 **if** ((numBurritos > 0) && (customersWaiting.size() < 15)) {  
  
 *//Creating new Customer object* Customer customer = **new** Customer(customersWaiting, custId, numBurritos);  
  
 *//Adding customers to linked list* customersWaiting.add(customer);  
  
 *//Sorting waiting customers* Collections.*sort*(customersWaiting, **new** Comparator<Customer>() {  
  
 **public int** compare(Customer c1, Customer c2) {  
 **if** (c1.getNumBurritos() > c2.getNumBurritos()) {  
 **return** 1;  
 } **else if** (c1.getNumBurritos() < c2.getNumBurritos()) {  
 **return** -1;  
 }  
 **return** 0;  
 }  
  
 });  
  
 **new** Thread(customer).start();  
 }  
 }  
  
 **final** String[] SERVERS = {**"Anne"**, **"Joe"**, **"Bob"**};  
  
 *//Creation of server threads* **for** (**int** i = 0; i < SERVERS.**length**; i++) {  
 BurritoServer server = **new** BurritoServer(customersWaiting, SERVERS[i]);  
 **new** Thread(server).start();  
  
 }  
  
 }  
}

**package** com.example.java;  
  
**import** java.util.LinkedList;  
**import** java.util.List;  
**import** java.util.concurrent.Semaphore;  
  
**import static** java.lang.Thread.sleep;  
  
*/\*\*  
 \* Created by AnabetsyRivero on 10/31/16.  
 \*/***public class** Customer **implements** Runnable {  
  
 LinkedList<Customer> **customersWaiting**;  
  
 **public int custId**;  
 **public int numBurritos**;  
 **private boolean notServed** = **true**;  
 **private int freeSeats** = 15;  
  
 **private** Semaphore **waitingArea** = **new** Semaphore(15);  
 **private** Semaphore **customers** = **new** Semaphore(1);  
 **private** Semaphore **counters** = **new** Semaphore(3);  
 **private** Semaphore **register** = **new** Semaphore(1);  
  
  
 **public** Customer(List<Customer> customersWaiting, **int** custId, **int** numBurritos) {  
 **this**.**customersWaiting** = (LinkedList<Customer>) customersWaiting;  
 **this**.**custId** = custId;  
 **this**.**numBurritos** = numBurritos;  
 }  
  
  
 **public int** getCustId(){  
 **return custId**;  
 }  
  
  
 **public int** getNumBurritos(){  
 **return numBurritos**;  
 }  
  
  
 @Override  
 **public** String toString(){  
 **return this**.getCustId() + **": "** + **this**.getNumBurritos() + **" burritos"** + **"\n"**;  
 }  
  
  
  
 *//this method simulates getting a burrito* **public void** get\_burrito(){  
 **try** {  
 *sleep*(1000);  
 } **catch** (InterruptedException ex) {}  
 }  
  
 *//this method simulates paying for burritos* **public void** pay\_burritos(){  
 **try** {  
 **register**.acquire();  
 *sleep*(2000);*//just added 6:17* System.***out***.println(**"Customer "** + **this**.getCustId() + **" is paying for his order... "** + **"\n"**);  
 **register**.release();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 *//this method simulates leaving the shop* **public void** leave\_shop(){  
 System.***out***.println(**"Customer "** + **this**.getCustId() + **" is leaving the shop. "** + **"\n"** );  
 }  
  
  
  
  
 @Override  
 **public void** run() {  
 **while** (**notServed**) { *// as long as the customer is not served  
  
 //tries to get access to the chairs* **if** (**customersWaiting**.size() < 15) {  
 **try** {  
 **waitingArea**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(**"Customer "** + **this**.getCustId() + **" with an order of "** + **this**.getNumBurritos() + **" burritos just sat down."** + **"\n"**);  
  
 **freeSeats**--;  
 **try** {  
 **customers**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 } *//just did at 11:33* **waitingArea**.release();  
 **try** {  
 **counters**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **try** {  
 **counters**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 **this**.get\_burrito();  
  
 **customers**.release();*//just now* **counters**.release();  
 **this**.pay\_burritos();  
 **this**.leave\_shop();  
  
  
 **if**(**numBurritos** > 3){  
 **numBurritos** -= 3;  
 **try** {  
 **waitingArea**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 *// System.out.println("Customer lol " + this.getCustId() + " with an order of " + this.getNumBurritos() + " burritos just sat down." + "\n");* **freeSeats**--;  
  
 **try** {  
 **customers**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 */\* Customer customer1 = new Customer(customersWaiting, custId, numBurritos);  
 customersWaiting.add(customer1);  
  
 Collections.sort(customersWaiting, new Comparator<Customer>() {  
  
 public int compare(Customer c1, Customer c2) {  
 if (c1.getNumBurritos() > c2.getNumBurritos()) {  
 return 1;  
 } else if (c1.getNumBurritos() < c2.getNumBurritos()) {  
 return -1;  
 }  
 return 0;  
 }  
  
 }); \*/* **waitingArea**.release();  
 **customers**.release();  
  
 **try** {  
 **counters**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **this**.get\_burrito();  
 **counters**.release();  
  
 **this**.pay\_burritos();  
 **this**.leave\_shop();  
  
  
 } **else if**((**numBurritos** > 0) && (**numBurritos** <= 3)) {  
 **numBurritos** -= 3;  
 **if** (**numBurritos** <= 0) {  
 **notServed** = **false**;  
 } **else** {  
 **try** {  
 **waitingArea**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **freeSeats**--;  
  
  
 **try** {  
 **customers**.acquire(); *//just added at 2:55pm* } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 */\* Customer customer2 = new Customer(customersWaiting, custId, numBurritos);  
 customersWaiting.add(customer2);  
  
 Collections.sort(customersWaiting, new Comparator<Customer>() {  
  
 public int compare(Customer c1, Customer c2) {  
 if (c1.getNumBurritos() > c2.getNumBurritos()) {  
 return 1;  
 } else if (c1.getNumBurritos() < c2.getNumBurritos()) {  
 return -1;  
 }  
 return 0;  
 }  
  
 }); \*/* **waitingArea**.release();  
 **customers**.release();  
  
 **try** {  
 **counters**.acquire();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **this**.get\_burrito();  
 **counters**.release();  
 **this**.pay\_burritos();  
 **this**.leave\_shop();  
 }  
 }  
  
 }  
 **else** { *// there are no free seats* System.***out***.println(**"There are no free seats. Customer "** + **this**.getCustId() + **" has left Burrito Brothers."** + **"\n"**);  
 **notServed**=**true**; *// the customer will leave since there are no spots left in the seating area* }  
 }  
 }  
  
}

**package** com.example.java;  
  
**import** java.util.LinkedList;  
**import** java.util.List;  
**import** java.util.concurrent.Semaphore;  
  
**import static** java.lang.Thread.sleep;  
  
*/\*\*  
 \* Created by AnabetsyRivero on 10/31/16.  
 \*/***public class** BurritoServer **implements** Runnable {  
 **private final** String **name**;  
 Semaphore **customers** = **new** Semaphore(1);  
  
  
 LinkedList<Customer> **customersWaiting**;  
  
 **public** BurritoServer(List<Customer> customersWaiting, String name) {  
 **this**.**customersWaiting** = (LinkedList<Customer>) customersWaiting;  
 **this**.**name** = name;  
 **this**.**customersWaiting** = (LinkedList<Customer>) customersWaiting;  
 }  
  
 @Override  
 **public** String toString(){  
 **return "Server name: "** + **name**;  
 }  
  
  
 @Override  
 **public void** run() {  
 **while**(**customersWaiting**.size() > 0) {  
 **try** {  
 **customers**.acquire();  
 **this**.makeBurrito(); *//making burritos...* } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **customers**.release();  
 }  
 }  
  
 *// this method will simulate making burritos* **public void** makeBurrito() **throws** InterruptedException {  
  
 System.***out***.println(**" Server "** + **name** + **" is serving customer "** + **customersWaiting**.pollFirst());  
 *sleep*(500);  
 }  
  
}

customers.txt

1 5  
2 3  
3 1  
4 6  
5 2  
6 4  
7 1  
8 3  
9 5  
10 1  
11 2  
12 3  
13 4  
14 1  
15 2