作業(水餃店銷售模擬器 ) (A1103343\_童筱文)

丁丁退休之後打算開一間水餃店，店裡每天預計準備5000顆豬肉水餃，3000顆牛肉水餃以及1000顆蔬菜水餃，請您幫丁丁的水餃店設計一個銷售的模擬器。

(1) 請使用多執行緒來設計此模擬器，並考慮同步的問題。

(2) 使用者可以輸入同時光顧的顧客數目。

(3) 每位顧客每次可同時點10到50顆水餃，模擬點水餃數請以亂數產生。

(4) 請亂數選取顧客點餐的水餃種類(豬肉水餃, 牛肉水餃或蔬菜水餃)。

(5) 水餃售完為止。

(6) 當顧客點餐後，在下一位顧客開始點餐前必須有等待服務生的時間，此等待的暫停時間為3秒。

ANSWER:

以下是使用Java程式語言設計的水餃店銷售模擬器：

import java.util.Random;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

class DumplingShopSimulator {

private static final int PORK\_DUMPLINGS = 5000;

private static final int BEEF\_DUMPLINGS = 3000;

private static final int VEGETABLE\_DUMPLINGS = 1000;

private static final int MIN\_DUMPLINGS\_ORDER = 10;

private static final int MAX\_DUMPLINGS\_ORDER = 50;

private static final Object lock = new Object();

private int porkDumplingsRemaining = PORK\_DUMPLINGS;

private int beefDumplingsRemaining = BEEF\_DUMPLINGS;

private int vegetableDumplingsRemaining = VEGETABLE\_DUMPLINGS;

private Random random = new Random();

private class Customer implements Runnable {

private String name;

private int dumplingsOrdered;

private String dumplingType;

public Customer(String name) {

this.name = name;

}

private void placeOrder() {

dumplingsOrdered = random.nextInt(MAX\_DUMPLINGS\_ORDER - MIN\_DUMPLINGS\_ORDER + 1) + MIN\_DUMPLINGS\_ORDER;

int randomDumplingType = random.nextInt(3);

if (randomDumplingType == 0 && porkDumplingsRemaining > 0) {

dumplingType = "豬肉水餃";

porkDumplingsRemaining -= dumplingsOrdered;

} else if (randomDumplingType == 1 && beefDumplingsRemaining > 0) {

dumplingType = "牛肉水餃";

beefDumplingsRemaining -= dumplingsOrdered;

} else if (randomDumplingType == 2 && vegetableDumplingsRemaining > 0) {

dumplingType = "蔬菜水餃";

vegetableDumplingsRemaining -= dumplingsOrdered;

} else {

dumplingsOrdered = 0; // No dumplings available

}

}

private void serveOrder() {

if (dumplingsOrdered > 0) {

System.out.println(name + " 點了 " + dumplingsOrdered + " 顆 " + dumplingType);

} else {

System.out.println(name + " 離開了，水餃售完了！");

}

}

@Override

public void run() {

synchronized (lock) {

try {

Thread.sleep(3000); // 等待服務生

} catch (InterruptedException e) {

e.printStackTrace();

}

placeOrder();

serveOrder();

}

}

}

public void simulate(int numCustomers) {

ExecutorService executorService = Executors.newFixedThreadPool(numCustomers);

for (int i = 1; i <= numCustomers; i++) {

executorService.execute(new Customer("顧客" + i));

}

executorService.shutdown();

}

public static void main(String[] args) {

DumplingShopSimulator simulator = new DumplingShopSimulator();

int numCustomers = 5; // 使用者可以輸入同時光顧的顧客數目

simulator.simulate(numCustomers);

}

}