****

FACULTY *of* SCIENCE *and* ENGINEERING

Department of Computer Science

and Information Systems

**The Project for CS5741**

Concurrency and Parallelism

in Software Development

Yanjun Wang (15015882)

submitted:2015/11/20

Table of Content

Class Diagram 3

Classes Description 4

Test strategy 5

# Class Diagram



# Classes Description

**Customer**

boolean fast: if this customer is in fast checkout

Vector<Double> items: items of this customer

int numOfitems: the number of products for each trolley

double numOftime: the time for each product

int x,y: the location of this customer in the market

int numofCh: number of heckouts

Checkout[] Ch: queue of checkouts

Market belong: the market the customer belongs to

boolean stateofTh: if this customer thread started

Customer(): (Constructor) initialize items, numOfitems and numOftime of this customer

run(): thread of this customer, for changing checkout

numOfitems(): return the numOfitems of this customer

eachtime(): return the cost time of each items for scanning

**Checkout**

boolean stateofCh: if this checkout thread started

int numOfcustomers: number of customers in this checkout

int no: No. of this checkout in the market

Vector<Customer> queue: the queue of customers of this checkout

Market market: the market the customer belongs to

Checkout(): (Constructor) pass the parameters of numOfcustomers, queue, no and market

run(): thread of this checkout, for checkout

display(): print the information of customers of this checkout

check(): customers checkout

**Market**

Vector<Customer> allCustomers: all the customers in this market

Checkout[] newCh: all the checkouts in this market

Vector<Vector<Customer>> queue: all the queues in all checkouts

int sumOfcustomers: the number of customers in this market

int numofLost: the number of lost customers

int numOfch: the number of checkouts

boolean newflag: if this market uses new scanners

Market(): (Constructor) pass parameters of numOfch, sumOfcustomers, allCustomers and newflag

init(): initialize all the customers in all the checkouts

**Add**

run(): a thread for adding new customer

add(): add a new customer randomly

**GUI**

GUI(): (Constructor) initialize the fream

createFream(): create a fream for testing the program

compare(): compare 3 different situations

# Test strategy

Run GUI.main()

Click the button Start to simulate the process of customer check out (the new customers can be added)

Click the button Compare to compare the following different situations:

1. If we use six checkouts instead of five, how much does that reduce the waiting time for each customer?

Is it worth it?

1. Is it useful to open another "5 items or less" checkout? Or should it be "10 items or less"?
2. There is some new technology on the market:  scanners that would reduce the scanning time for each item (they are much more reliable). But three of the new scanners cost as much as five of the old ones.

Is it worth buying them?

Does that really make a difference?