**/\* Barber.java**

**\***

**\* The Barber is implemented as thread, and its main activities are sleeping for a random time,**

**\* and then call the methods in Barbershop class.**

**\***

**\***

**\* @author: Jie Zhang**

**\* Last Updated: 07/19/2002**

**\*/**

**public class Barber extends Thread{**

**private BarberShop shop;**

**private BarberShopApplet tapplet;**

**private int pid;**

**int delay = 2500;**

**int status = 0;**

**int customerID = 0;**

**public Barber(BarberShopApplet applet, BarberShop iq, int id){**

**shop = iq;**

**tapplet = applet;**

**pid = id;**

**}**

**public void run(){**

**while(true){**

**try{**

**status = 0;**

**tapplet.mc.println(status, "b", pid);**

**sleep((int)(Math.random()\*delay));**

**shop.cutHair(tapplet, pid);**

**sleep((int) (Math.random()\*delay));**

**shop.finishCut(tapplet, pid);**

**} catch(InterruptedException e){**

**System.err.println("Exception " + e.toString());**

**}**

**}**

**}**

**}**

**/\* BarberShop.java**

**\***

**\* This class is the program that controls all the activities of customers and barbers.**

**\* This barber shop has maximum capacity of 8 customers. There are 3 barbers in the**

**\* shop.**

**\* There are 4 sofa seats and 3 barber chairs in the shop. Three barbers will spend**

**\* their time on sleeping for a while, cutting hair and performing as a cashier while**

**\* he/she is not cutting hair.**

**\* The customer will wait outside of the shop if there are already 8 customers in the shop.**

**\* After the customer enters into the shop, he/she will wait in the standing area first, then**

**\* sit on sofa in the order that he/she arrives. He/she will sit on the barber chair if there**

**\* is any barber chair available. After the barber finishes the hair cut for him/her, this**

**\* customer will wait to pay, then exit from the barbershop.**

**\***

**\***

**\* @author: Jie Zhang**

**\* Last Updated: 07/19/2002**

**\*/**

**import java.awt.\*;**

**public class BarberShop extends Canvas**

**{**

**private int chairSize = 3;**

**private int sofaSize = 4;**

**private int frameDelay = 3560;**

**private int[] customerSofaQ; //the queue to hold the customers on the sofa**

**private int[] customerStandQ; //the queue to hold the customers on the standing area**

**private int[] customerChairQ; //the queue to hold the customers on Barber Chairs**

**private int[] customerPayQ; //the queue to hold the customers waiting for paying.**

**private int[] customerReady; //cutomerReady[i] = 1, customer i is ready for barber 1**

**private int[] finishedCustomerQ; //the array to hold the cut finish flags**

**private int[] paidCustomerQ;**

**private int[] exitArray; //the array hold all the customers in the order they exit shop**

**private int sofaTop, sofaBottom; //for customerSofaQ**

**private int chairTop, chairBottom; //for customerChairQ**

**private int payTop, payBottom; //for customerPayQ**

**private int customerTop, customerBottom;//for customersQ**

**private int customerOnSofa; //the count of customers on the sofa**

**private int customerOnChair; //the count of customers on the barber chairs**

**private int customerStandCount; //the count of customers standing**

**private int wantPayCount; //the count of customers waiting for paying**

**private int hasCashier;**

**private int cashierID; //the barber ID for who is performing as a cashier**

**private int exitID; //the customer ID who is leaving the barbershop**

**private int exitTop;**

**private int customerCount;**

**private int size;**

**private int[] customerOut;**

**private int outTop;**

**private int outBottom;**

**private int repaintFlag = 0;**

**private Font font;**

**private FontMetrics fm;**

**private int x; //customer consumed item**

**public BarberShop( ){**

**size = 4; //default buffer size**

**customerTop = customerBottom = 1;**

**payTop = payBottom = 1;**

**chairTop = chairBottom = 0;**

**sofaTop = sofaBottom = 0;**

**customerCount = 0;**

**customerOnSofa = 0;**

**customerOnChair = 0;**

**customerStandCount = 0;**

**wantPayCount = 0;**

**hasCashier = 0;**

**cashierID = 0;**

**exitID = 0;**

**exitTop = 0;**

**finishedCustomerQ = new int[11];**

**customerOut = new int[2];**

**outTop = outBottom = 0;**

**setSize(size);**

**resize(500, 300);**

**setBackground(Color.white);**

**font = new Font("TimesRoman", Font.BOLD, 18);**

**fm = getFontMetrics(font);**

**}**

**public void setSize(int s)**

**{**

**size = s;**

**if(size > 8) customerStandCount = 8;**

**else customerStandCount = size;**

**int tmpCount = 0;**

**if(size > 8)**

**{ tmpCount = size - 8;**

**System.out.println("the tmpCount is " + tmpCount);**

**for(int i = 0; i < tmpCount; i++)**

**{**

**customerOut[i] = 9+i;**

**}**

**}**

**outBottom = 0;**

**outTop = 1;**

**customerSofaQ = new int[sofaSize];**

**customerChairQ = new int[chairSize+1];**

**customerPayQ = new int[11];**

**customerReady = new int[size+1]; //the maximum customer size is 10**

**paidCustomerQ = new int[size+1];**

**exitArray = new int[size];**

**/\* Initialize the array\*/**

**for(int i = 1; i <=size ; i++)**

**{**

**customerReady[i] = 0;**

**}**

**repaint();**

**}**

**public synchronized boolean chairFull(){**

**return customerOnChair == chairSize;**

**}**

**public synchronized boolean sofaFull(){**

**return customerOnSofa == sofaSize;**

**}**

**/\*\***

**\* If there is no customer on this barber chair, then check if there is a cashier's job.**

**\* If he/she is doing a cashier's job, he/she will finish that first. Then the barber**

**\* will wait for the customer ready. After customer ready, he/she will cutting the hair**

**\* for a random time period.**

**\* The barber will just cutting hair for the customer sitting on**

**\* his/her chair. i.e. barber 1 will just cutting hair for the customer**

**\* sitting on the barber chair 1.**

**\*/**

**public synchronized void cutHair(BarberShopApplet applet, int id)**

**{**

**if(customerReady[id] == 0) getCashierLock(applet, id);**

**if(cashierID == id) performCashier(applet, id);**

**while(customerReady[id] == 0) //if there is no customer is waiting**

**{**

**updateBarberStatus(applet, id, 4);**

**try{ wait(); }catch(InterruptedException e){**

**System.err.println("Exception " + e.toString());**

**}**

**}**

**System.out.println("customerReady are: ");**

**for(int i = 0; i <= 3; i++)**

**{**

**System.out.println(Integer.toString(customerReady[i]));**

**}**

**int x = customerReady[id];**

**applet.b[id].customerID = x;**

**applet.c[x].barberID = id;**

**System.out.println("x is " + x);**

**applet.b[id].status = 1;**

**applet.mc.println(applet.b[id].status, "b", id, x);**

**updateCustomerStatus (applet, x, 1); //cutting Hair**

**//repaint();**

**notifyAll();**

**}**

**/\* When the barber finishes the haircutting, he/she will signal the customer finish flag**

**\* and update corresponding variables.**

**\*/**

**public synchronized void finishCut(BarberShopApplet applet, int id)**

**{**

**customerReady[id] = 0;**

**int y = applet.b[id].customerID;**

**/\* The following codes added to animate one of the unfair situation:**

**\* The process hold the resources unnecessarily.**

**\* The finished process has to wait until the prior process to finish**

**\* (the first process in this program).**

**\*/**

**if(applet.haltFlag == 1)**

**{**

**if(y != 1)**

**{**

**updateCustomerStatus(applet, y, 10);**

**updateBarberStatus(applet, id, 1);**

**}**

**else**

**{**

**while(true) /\* to keep the barber status in cutting hair \*/**

**{**

**try { wait(); } catch(InterruptedException e) {}**

**}**

**}**

**}**

**else if(applet.requestFlag == 1)**

**{**

**System.out.println("process is " + y);**

**if(y == 1)**

**{**

**while(finishedCustomerQ[2] != 1)**

**{ try { wait(); } catch(InterruptedException e) {}**

**}**

**updateCustomerStatus(applet, y, 11);**

**}**

**else if(y==3)**

**{**

**while(finishedCustomerQ[2] != 1)**

**{**

**try { wait(); } catch(InterruptedException e) {}**

**}**

**updateCustomerStatus(applet, y, 7);**

**}**

**else**

**{**

**updateCustomerStatus(applet, y, 7); //waiting for pay**

**}**

**customerChairQ[id] = 0;**

**applet.b[id].customerID = 0;**

**applet.c[y].barberID = 0;**

**repaint();**

**finishedCustomerQ[y] = 1;**

**notifyAll();**

**wantPayCount ++;**

**customerPayQ[y] = y;**

**repaint();**

**customerOnChair --;**

**notifyAll();**

**if(wantPayCount > 0) getCashierLock(applet, id);**

**if(cashierID == id) performCashier(applet, id);**

**else updateBarberStatus(applet, id, 4);**

**}**

**else // To handle the processes in fair situation**

**{**

**updateCustomerStatus(applet, y, 7);**

**customerChairQ[id] = 0;**

**applet.b[id].customerID = 0;**

**applet.c[y].barberID = 0;**

**repaint();**

**System.out.println("customer " + y + " finish cutting");**

**finishedCustomerQ[y] = 1;**

**wantPayCount ++;**

**customerPayQ[payTop] = y;**

**payTop++;**

**repaint();**

**customerOnChair --;**

**notifyAll();**

**if(wantPayCount > 0) getCashierLock(applet, id);**

**if(cashierID == id) performCashier(applet, id);**

**else updateBarberStatus(applet, id, 4);**

**}**

**}**

**/\* The customer will first wait for his/her turn, then a available seat on the sofa.**

**\*/**

**public synchronized void sitSofa(BarberShopApplet applet, int id)**

**{**

**while(customerBottom != id)**

**{**

**System.out.println("customer " + id + " is waiting for the turn");**

**try{ wait(); } catch(InterruptedException e) {}**

**}**

**customerCount++;**

**notifyAll();**

**if(id > 8)**

**{ customerStandCount ++;**

**outBottom ++;**

**repaint();**

**}**

**while(sofaFull())**

**{**

**try { wait(); }catch(InterruptedException e) {}**

**}**

**customerBottom++;**

**notifyAll();**

**customerOnSofa ++;**

**customerStandCount --;**

**customerSofaQ[sofaTop] = id;**

**sofaTop =(sofaTop+1)%sofaSize;**

**repaint();**

**updateCustomerStatus(applet, id, 5); //sitting on sofa**

**notifyAll();**

**}**

**/\* The customer will first wait for his/her turn, then an available barber chair.**

**\* He/she will spend a random time on the chair before send the ready flag to the barber.**

**\*/**

**public synchronized void sitBarberChair(BarberShopApplet applet, int id)**

**{**

**while(customerSofaQ[sofaBottom] != id)**

**{**

**System.out.println("Customer " + id + "is waiting for the chair turn");**

**try{ wait(); } catch(InterruptedException e) { }**

**}**

**while(chairFull())**

**{**

**try { wait(); }catch(InterruptedException e) {}**

**}**

**customerSofaQ[sofaBottom] = 0;**

**sofaBottom =(sofaBottom+1)%sofaSize; //get up from sofa**

**customerOnSofa --;**

**customerOnChair ++;**

**for(int i = 1; i <= chairSize; i++)**

**{**

**if(customerChairQ[i] == 0)**

**{**

**customerChairQ[i] = id;**

**customerReady[i] = id;**

**i = chairSize; // get out of the loop**

**}**

**}**

**updateCustomerStatus(applet,id, 6);**

**repaint();**

**try{**

**applet.c[id].sleep((int) (Math.random()\*frameDelay));**

**}catch(InterruptedException e) { }**

**notifyAll();**

**}**

**/\* If there is a customer waiting and no cashier, a barber will be a cashier when he/she**

**\* is not cutting hair.**

**\*/**

**public synchronized void getCashierLock(BarberShopApplet applet, int bid)**

**{**

**if((wantPayCount > 0) && (hasCashier!= 1))**

**{**

**hasCashier= 1;**

**cashierID = bid;**

**//updateBarberStatus(applet, bid, 9); // a cashier right now**

**repaint();**

**System.out.println("Barber " + bid + " got the cashier Lock right now");**

**notifyAll();**

**}**

**}**

**public synchronized void performCashier(BarberShopApplet applet, int bid)**

**{**

**while(wantPayCount > 0)**

**{**

**System.out.println("Barber " + bid + " is a cashier right now");**

**updateBarberStatus(applet, bid, 2);**

**try{ wait(); } catch(InterruptedException e) {}**

**}**

**cashierID = 0;**

**hasCashier= 0;**

**notifyAll();**

**}**

**/\* The customer will wait for a cashier first, then wait for his/her turn to pay.**

**\* It will take random time to get the receipt, the customer then will leave the shop.**

**\*/**

**public synchronized void waitPay(BarberShopApplet applet, int cid)**

**{**

**while(customerPayQ[payBottom] != cid)**

**{**

**if((applet.requestFlag == 1) && (cid == 3))**

**repaintFlag = 1;**

**try{ wait(); } catch(InterruptedException e) { }**

**}**

**if(applet.requestFlag == 1) // for 1st process in unfair situation**

**{**

**while(true)**

**{**

**try{ applet.c[cid].sleep((int) (Math.random()\*frameDelay)); }**

**catch(InterruptedException e) { }**

**}**

**}**

**while(hasCashier!= 1)**

**{**

**try{ wait(); } catch(InterruptedException e) { }**

**}**

**try{ applet.c[cid].sleep((int) (Math.random()\*frameDelay)); }**

**catch(InterruptedException e) {}**

**updateCustomerStatus(applet, cid, 9);**

**payBottom++;**

**wantPayCount --;**

**exitID = cid;**

**exitArray[exitTop] = cid;**

**exitTop ++;**

**customerCount --;**

**repaint();**

**notifyAll();**

**}**

**public synchronized void updateCustomerStatus(BarberShopApplet applet, int cid, int status)**

**{**

**applet.c[cid].status = status;**

**applet.mc.println(status, "c", cid);**

**}**

**public synchronized void updateBarberStatus(BarberShopApplet applet, int bid, int status)**

**{**

**applet.b[bid].status = status;**

**applet.mc.println(status, "b", bid);**

**}**

**public void clear()**

**{**

**size = 4; //default buffer size**

**customerCount = 0;**

**payTop = payBottom = 1;**

**chairTop = chairBottom = 0;**

**sofaTop = sofaBottom = 0;**

**customerTop= customerBottom= 1;**

**outTop = outBottom = 0;**

**finishedCustomerQ = new int[11];**

**customerOut = new int[2];**

**customerOnSofa = 0; //the count of customers on the sofa**

**customerOnChair = 0; //the count of customers on the barber chairs**

**customerStandCount = 0; //the count of customers standing**

**wantPayCount = 0; //the count of customers waiting for paying**

**hasCashier = 0;**

**cashierID = 0; //the barber ID for who is performing as a cashier**

**exitID = 0;**

**exitTop = 0;**

**repaintFlag = 0;**

**}**

**/\***

**\* Draw the barber shop on the canvas**

**\*/**

**public void paint(Graphics g){**

**g.setFont(new Font("TimesRoman", Font.BOLD, 12));**

**g.setColor(Color.blue);**

**int xpos = 120;**

**int ypos = 10;**

**g.setFont(new Font("TimesRoman", Font.BOLD, 18));**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Draw Barber Chairs on the canvas \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**g.drawString("Barber Chairs", xpos+150, ypos+5);**

**for(int i = 1; i <= chairSize; i++)**

**{**

**g.draw3DRect(xpos+100+70\*(i-1), ypos+20, 28, 28, true);**

**if(i != cashierID) g.drawString("B"+i, xpos+103+70\*(i-1), ypos+70);**

**}**

**g.setColor(Color.red);**

**for(int j=1; j <= chairSize; j ++)**

**{**

**if(customerChairQ[j] != 0)**

**{**

**g.drawString(Integer.toString(customerChairQ[j]), xpos + 105 + 70\*(j-1), ypos+35);**

**g.draw3DRect(xpos+100+70\*(j-1), ypos+20, 28, 28, true);**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Draw Cashier's waiting queue \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**g.setColor(Color.blue);**

**g.drawString("Cashier", xpos+410, ypos+45);**

**g.setFont(new Font("TimesRoman", Font.BOLD, 14));**

**if(cashierID != 0)**

**{**

**g.drawString("B "+cashierID, xpos+430, ypos+20);**

**}**

**g.draw3DRect(xpos+410, ypos+60, 60, 20, true);**

**g.setFont(new Font("TimesRoman", Font.BOLD, 12));**

**int b = payBottom;**

**System.out.println("wantPaycount is " + wantPayCount);**

**if(repaintFlag == 1)**

**{**

**for(int i = 0; i < 3; i++)**

**{**

**if(customerPayQ[i+b] != 0)**

**g.drawString("C"+customerPayQ[i+b], xpos+430, ypos+100);**

**ypos += 20;**

**}**

**}**

**else**

**{**

**for(int i = 0; i < wantPayCount; i++)**

**{**

**if(customerPayQ[i+b] != 0)**

**g.drawString("C"+customerPayQ[i+b], xpos+430, ypos+100);**

**ypos += 20;**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Draw standing room on canvas \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**g.setFont(new Font("TimesRoman", Font.BOLD, 12));**

**ypos = 10;**

**g.drawString("Standing Room Area", xpos-100, ypos+160);**

**b = customerBottom;**

**for(int i = 0; i < customerStandCount; i++)**

**{**

**g.drawString("C"+(i+b), xpos+80-25\*i, ypos+120);**

**}**

**g.setColor(Color.green);**

**g.drawString("Entrance", xpos-110, ypos+100);**

**g.drawString("--------->", xpos-110, ypos+105);**

**g.setColor(Color.red);**

**System.out.println("outTop is: " + outTop);**

**for(int i = outBottom; i <= outTop; i++)**

**{**

**if(customerOut[i] > 0)**

**g.drawString("C "+customerOut[i], xpos-80, ypos+80-20\*i);**

**}**

**g.setColor(Color.red);**

**g.drawString("Exit", xpos+530, ypos+10);**

**for(int i = 0; i < exitTop; i++)**

**{**

**if(exitArray[i] != 0)**

**g.drawString("C" + exitArray[i], xpos+530, ypos+25+15\*i);**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Draw waiting Sofa on canvas \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**xpos = 100;**

**ypos = 10;**

**g.setColor(Color.blue);**

**g.drawString("Sofa", xpos+225, ypos+185);**

**for(int i = 0; i < sofaSize; i++)**

**{**

**g.draw3DRect(xpos+180+28\*(i), ypos+140, 28, 28, true);**

**}**

**g.setColor(Color.red);**

**int k = sofaBottom;**

**for(int j=0; j < customerOnSofa; j++)**

**{**

**g.drawString(Integer.toString(customerSofaQ[k]), xpos + (190+28\*3) -28\*(j), ypos+155);**

**g.draw3DRect(xpos+180+28\*3 - 28\*(j), ypos+140, 28, 28, true);**

**k = (k+1)%sofaSize;**

**}**

**}**

**}**

**/\* File: BarberShopApplet.java**

**\***

**\* This is a Java applet file for Barbershop problem animation. The GUI**

**\* of this applet contains three parts: animation canvas, message canvas**

**\* and a button panel.**

**\* The animation canvas is where the Barbershop animation is displayed.**

**\* The message canvas is where the statues of barbers and customers are displayed.**

**\* The button panel has 6 basic buttons: START, STOP, PAUSE, CONTINUE, FASTER,**

**\* SLOWER.**

**\***

**\* This applet will allow user to choose from a fair barbershop or unfair barbershop.**

**\* In the fair barbershop, unless the user choose the number of customers, default**

**\* number of the customers is 4.**

**\* The number of customers in the unfair barbershop is also 4. And there are two**

**\* unfair situations that the user can choose from.**

**\***

**\***

**\* @author: Jie Zhang**

**\* Last Updated: 07/19/2002**

**\*/**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.util.\*;**

**import java.applet.Applet;**

**import java.lang.\*;**

**public class BarberShopApplet extends Applet**

**{**

**private BarberShopApplet applet = this;**

**private BarberShop myShop;**

**private Button fastButton, slowButton, stopButton, startButton,pauseButton, continueButton;**

**private Panel buttonPanel, namePanel, namePanel2;**

**private Checkbox haltResource, requestResource;**

**private Choice customer;**

**private int customerN = 4; //default number of customer and barber**

**private int barberN = 3;**

**private Thread at;**

**MessageCanvas mc;**

**Customer[] c;**

**Barber[] b;**

**int haltFlag = 0;**

**int requestFlag = 0;**

**synchronized void startPushed() {notify();}**

**synchronized void stopPushed() {notify();}**

**public void init() {**

**resize(800, 600);**

**setLayout(new GridLayout(3, 1));**

**myShop = new BarberShop();**

**mc = new MessageCanvas();**

**add(myShop); //add BarberShop canvas at the top**

**add(mc); //add message box canvas in the middle**

**buttonPanel = new Panel();**

**Panel namePanel = new Panel();**

**Panel bPanel = new Panel(); // to hold all buttons and the labels**

**bPanel.setFont(new Font("TimesRoman", Font.BOLD, 14));**

**bPanel.setLayout(new GridLayout(3, 1));**

**buttonPanel.add(startButton = new Button("START"));**

**buttonPanel.add(stopButton = new Button("STOP"));**

**buttonPanel.add(pauseButton = new Button("PAUSE"));**

**buttonPanel.add(continueButton = new Button("CONTINUE"));**

**buttonPanel.add(fastButton = new Button("FAST"));**

**buttonPanel.add(slowButton = new Button("SLOW"));**

**Label titleLabel = new Label("Fair Barbershop", Label.CENTER);**

**titleLabel.setFont(new Font("TimesRoman", Font.BOLD, 16));**

**titleLabel.setForeground(Color.blue);**

**Label textLabel = new Label("Maximum Shop Capacity is 8 Customers", Label.CENTER);**

**Label titleLabel2 = new Label("Unfair Barbershop ", Label.CENTER);**

**Label textLabel2 = new Label("4 Customers In The Shop", Label.CENTER);**

**titleLabel2.setFont(new Font("TimesRoman", Font.BOLD, 16));**

**titleLabel2.setForeground(Color.blue);**

**namePanel.setLayout(new GridLayout(2,1));**

**namePanel.add(titleLabel);**

**namePanel.add(textLabel);**

**namePanel2 = new Panel();**

**namePanel2.setLayout(new GridLayout(2,1));**

**namePanel2.add(titleLabel2);**

**namePanel2.add(textLabel2);**

**Panel titlePanel = new Panel();**

**titlePanel.setLayout(new GridLayout(1,2));**

**titlePanel.add(namePanel);**

**titlePanel.add(namePanel2);**

**Panel choicePanel = new Panel(); //to hold all the choice boxes**

**choicePanel.setLayout(new GridLayout(1,2));**

**customer = new Choice();**

**for(int i = 1; i <=10; i++)**

**{**

**customer.addItem(Integer.toString(i));**

**}**

**customer.select("4");**

**Label customerLabel = new Label("Number of Customers", 2);**

**customerLabel.setBackground(Color.lightGray);**

**Panel customerPanel = new Panel();**

**customerPanel.add(customerLabel);**

**customerPanel.add(customer);**

**Panel unfairPanel = new Panel();**

**unfairPanel.setLayout(new GridLayout(2,1));**

**CheckboxGroup g = new CheckboxGroup();**

**unfairPanel.add(haltResource = new Checkbox("Request finished, but resources are held unnecessarily", g, false));**

**unfairPanel.add(requestResource = new Checkbox("Request not finished, but resources are released", g, false));**

**choicePanel.add(customerPanel);**

**choicePanel.add(unfairPanel);**

**bPanel.add(titlePanel);**

**bPanel.add(choicePanel);**

**bPanel.add(buttonPanel);**

**add(bPanel);**

**}**

**public boolean action(Event evt, Object arg)**

**{**

**if(evt.target == customer)**

**{**

**customerN = Integer.parseInt(arg.toString());**

**haltResource.setEnabled(false);**

**requestResource.setEnabled(false);**

**return true;**

**}**

**else if(evt.target == haltResource)**

**{**

**startButton.setEnabled(false);**

**customer.setEnabled(false);**

**stopButton.setEnabled(true);**

**haltResource.setEnabled(false);**

**requestResource.setEnabled(false);**

**haltFlag = 1;**

**System.out.println("HaltResource");**

**customerN = 4;**

**myShop.setSize(customerN);**

**c = new Customer[customerN+1]; //Customer[0] is a dummy slot**

**b = new Barber[barberN+1];**

**mc.setMessage(barberN, customerN);**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i] = new Customer(applet, myShop, i);**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i] = new Barber(applet, myShop, i);**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i].start();**

**}**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i].start();**

**}**

**return true;**

**}**

**else if(evt.target == requestResource)**

**{**

**startButton.setEnabled(false);**

**stopButton.setEnabled(true);**

**customer.setEnabled(false);**

**haltResource.setEnabled(false);**

**requestResource.setEnabled(false);**

**System.out.println("RequestResource");**

**requestFlag = 1;**

**customerN = 4;**

**myShop.setSize(customerN);**

**c = new Customer[customerN+1]; //Customer[0] is a dummy slot**

**b = new Barber[barberN+1];**

**mc.setMessage(barberN, customerN);**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i] = new Customer(applet, myShop, i);**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i] = new Barber(applet, myShop, i);**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i].start();**

**}**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i].start();**

**}**

**return true;**

**}**

**else if(arg.equals("PAUSE"))**

**{ for(int i = 1; i <= customerN; i++)**

**{**

**if(c[i].isAlive()) c[i].suspend();**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**if(b[i].isAlive()) b[i].suspend();**

**}**

**fastButton.setEnabled(false);**

**slowButton.setEnabled(false);**

**return true;**

**}**

**else if(arg.equals("CONTINUE"))**

**{**

**for(int i = 1; i <= customerN; i++)**

**{**

**if(c[i].isAlive()) c[i].resume();**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**if(b[i].isAlive()) b[i].resume();**

**}**

**fastButton.setEnabled(true);**

**slowButton.setEnabled(true);**

**return true;**

**}**

**else if(arg.equals("FASTER"))**

**{**

**int newDelay = b[1].delay;**

**newDelay /= 2;**

**newDelay = newDelay < 100 ? 100: newDelay;**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i].delay = newDelay;**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i].delay = newDelay;**

**}**

**return true;**

**}**

**else if(arg.equals("SLOWER"))**

**{**

**int newDelay = b[1].delay;**

**newDelay \*= 2;**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i].delay = newDelay;**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i].delay = newDelay;**

**}**

**return true;**

**}**

**else if(arg.equals("START"))**

**{**

**myShop.setSize(customerN);**

**c = new Customer[customerN+1]; //Customer[0] is a dummy slot**

**b = new Barber[barberN+1];**

**mc.setMessage(barberN, customerN);**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i] = new Customer(applet, myShop, i);**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i] = new Barber(applet, myShop, i);**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**b[i].start();**

**}**

**for(int i = 1; i <= customerN; i++)**

**{**

**c[i].start();**

**}**

**applet.startPushed();**

**stopButton.setEnabled(true);**

**startButton.setEnabled(false);**

**fastButton.setEnabled(true);**

**slowButton.setEnabled(true);**

**customer.setEnabled(false);**

**haltResource.setEnabled(false);**

**requestResource.setEnabled(false);**

**return true;**

**}**

**else if(arg.equals("STOP"))**

**{**

**try{**

**for(int i = 1; i <= customerN; i++)**

**{**

**if(c[i].isAlive()) c[i].stop();**

**c[i] = null;**

**}**

**for(int i = 1; i <= barberN; i++)**

**{**

**if(b[i].isAlive()) b[i].stop();**

**b[i] = null;**

**}**

**}catch(Exception e) {}**

**myShop.clear();**

**applet.stopPushed();**

**haltFlag = 0;**

**requestFlag = 0;**

**startButton.setEnabled(true);**

**customer.setEnabled(true);**

**haltResource.setEnabled(true);**

**requestResource.setEnabled(true);**

**fastButton.setEnabled(true);**

**slowButton.setEnabled(true);**

**if(at != null) at.stop();**

**at = null;**

**return true;**

**}**

**else{ return false;}**

**}**

**}**

**/\* Customer.java**

**\***

**\* The Customer Thread's main activities are call the methods in Barbershop class.**

**\***

**\***

**\* @author: Jie Zhang**

**\* Last Updated: 07/19/2002**

**\*/**

**public class Customer extends Thread{**

**private BarberShopApplet tapplet;**

**private BarberShop shop;**

**private int cid;**

**int delay = 2500;**

**int status = 0;**

**int cutFinish = 0;**

**int barberID = 0;**

**int paid = 0;**

**public Customer(BarberShopApplet applet, BarberShop iq, int id){**

**shop = iq;**

**tapplet = applet;**

**cid = id;**

**}**

**public void run(){**

**try{**

**status = 0;**

**tapplet.mc.println(status, "c", cid);**

**shop.sitSofa(tapplet, cid);**

**sleep(delay);**

**shop.sitBarberChair(tapplet, cid);**

**shop.waitPay(tapplet, cid);**

**} catch(InterruptedException e){**

**System.err.println("Customer Exception " + e.toString());**

**}**

**}**

**}**

**/\* File: MessageCanvas.java**

**\***

**\* This class provides message canvas for the applet GUI.**

**\* It will print the statuses of customers and barbers on the GUI.**

**\***

**\* @author: Jie Zhang**

**\* Last Updated: 07/18/2002**

**\*/**

**import java.awt.\*;**

**class MessageCanvas extends Canvas**

**{**

**private Font font;**

**private FontMetrics fm;**

**private int[] barberStatus;**

**private int[] customerStatus;**

**private int[] serviceStatus; //store the customer id that is cutting hair for each barber**

**//serviceStatus[i]=j, Barber i is cutting hair for customer j**

**private int msgHeight;**

**private int msgWidth;**

**private int bn, cn;**

**private int frameDelay = 256;**

**public MessageCanvas( )**

**{**

**resize(size().width, 50);**

**setBackground(Color.green);**

**font = new Font("TimesRoman", 1, 18);**

**fm = getFontMetrics(font);**

**msgHeight = fm.getHeight();**

**}**

**public void setMessage(int barberN, int customerN)**

**{**

**bn = barberN;**

**cn = customerN;**

**barberStatus = new int[bn+1];**

**customerStatus = new int[cn+1];**

**serviceStatus = new int[bn+1];**

**repaint();**

**}**

**void println(String s)**

**{**

**msgWidth = fm.stringWidth(s);**

**repaint();**

**}**

**void println(int s, String st, int id)**

**{**

**if(st.equals("b"))**

**barberStatus[id] = s;**

**else**

**customerStatus[id] = s;**

**repaint();**

**}**

**void println(int s, String st, int id, int cid)**

**{**

**if(st.equals("b"))**

**{ barberStatus[id] = s;**

**serviceStatus[id] = cid;**

**}**

**else**

**customerStatus[id] = s;**

**repaint();**

**}**

**public void paint(Graphics g)**

**{**

**g.setFont(font);**

**int xpos = 40;**

**int ypos = 30;**

**g.drawString("Status of Customers: ", 60, 20);**

**g.drawString("Status of Barbers: ", 380, 20);**

**g.setFont(new Font("TimesRoman", 1, 12));**

**for(int i=1; i<=cn;i++)**

**{**

**g.setColor(Color.black);**

**g.drawString("C" + i, xpos, ypos+(12\*i+5\*(i-1)));**

**if(customerStatus[i] == 0)**

**{**

**g.setColor(Color.yellow);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Standing ...", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}**

**else if (customerStatus[i] == 1)**

**{**

**g.setColor(Color.gray);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Cutting Hair...", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}**

**else if (customerStatus[i] == 2)**

**{**

**g.setColor(Color.blue);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Waiting for Cashier", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}**

**else if (customerStatus[i] == 3)**

**{**

**g.setColor(Color.red);**

**g.fillOval(xpos+40,ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Finished the Hair Cut", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}**

**else if (customerStatus[i] == 4)**

**{**

**g.setColor(Color.red);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Waiting to pay", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}**

**else if (customerStatus[i] == 5)**

**{**

**g.setColor(Color.blue);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Sitting on the Sofa", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}else if (customerStatus[i] == 6)**

**{**

**g.setColor(Color.red);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Sitting on the BarberChair", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}else if (customerStatus[i] == 7)**

**{**

**g.setColor(Color.red);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Waiting to pay", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}else if (customerStatus[i] == 9)**

**{**

**g.setColor(Color.green);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Left ", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}else if (customerStatus[i] == 10)**

**{**

**g.setColor(Color.gray);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("Finished, but hold the resources unnecessarily", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}else if (customerStatus[i] == 11)**

**{**

**g.setColor(Color.gray);**

**g.fillOval(xpos+40, ypos+(2\*i+15\*(i-1)), 14, 14);**

**g.drawString("C1 hasn't finished but has to leave", xpos+80, ypos+(12\*i + 5\*(i-1)));**

**}**

**}**

**xpos = 380;**

**ypos = 40;**

**for(int i=1; i<=bn; i++)**

**{**

**g.setColor(Color.black);**

**g.drawString("B" + i, xpos, ypos+(15\*i+10\*(i-1)));**

**if(barberStatus[i] == 0)**

**{**

**g.setColor(Color.yellow);**

**g.fillOval(xpos+60, ypos+(2\*i+22\*(i-1)), 18, 18);**

**g.drawString("Sleeping ...", xpos+120, ypos+(15\*i + 10\*(i-1)));**

**}**

**else if (barberStatus[i] == 1)**

**{**

**g.setColor(Color.gray);**

**g.fillOval(xpos+60, ypos+(2\*i+22\*(i-1)), 18, 18);**

**g.drawString("Cutting Hair for C"+serviceStatus[i], xpos+120, ypos+(15\*i + 10\*(i-1)));**

**}**

**else if (barberStatus[i] == 2)**

**{**

**g.setColor(Color.blue);**

**g.fillOval(xpos+60, ypos+(2\*i+22\*(i-1)), 18, 18);**

**g.drawString("Accepting Payment...", xpos+120, ypos+(15\*i + 10\*(i-1)));**

**}**

**else if (barberStatus[i] == 3)**

**{**

**g.setColor(Color.red);**

**g.fillOval(xpos+60, ypos+(2\*i+22\*(i-1)), 18, 18);**

**g.drawString("Finished the Hair Cut", xpos+120, ypos+(15\*i + 10\*(i-1)));**

**}**

**else if (barberStatus[i] == 4)**

**{**

**g.setColor(Color.gray);**

**g.fillOval(xpos+60, ypos+(2\*i+22\*(i-1)), 18, 18);**

**g.drawString("Waiting ... ", xpos+120, ypos+(15\*i + 10\*(i-1)));**

**}**

**}**

**}**

**}**