**Due Date**

* Project 4 E is due lab time on Tuesday, 18 April 2017 (55 points for this part)

**Estimated time to complete --** a few hours

**Group project --** You may work in groups of 2 or 3

**What you need to do** --- Pages 10, 11, and 12

**Background material**

* Review Inheritance, Polymorphism, and Interfaces -- Chapters 8 and 9 in the textbook
* **Review all code within each class in the project before adding any new code**
* Review Graphical User Interfaces -- Chapter 6

**Learning Objectives**

After completing this project you should be able to:

• Design, implement, and test a small class hierarchy

• Know how to define and implement a java class interface

• Implement a GUI-based application

• Work with a group on a project.

**Project Description**

For Project 4 part E, you will develop a Graphical User Interface (GUI) for the Food Court Simulation that was developed in Project 4 part D, along with a ***continuousSteps*** method in the Clock class. The GUI will give the user the control to interact with the project model.

A second part of this project is to define Queue163 and PriorityQueue163 classes, where these two classes are to be used everywhere in the project, instead of the Java API ArrayList, LinkedList, and PriorityQueue classes.

A third part of this project is to change the Eatery class to accommodate a person according to the value in the attention field.

**Project Startup**

Project4E.zip is posted on Blackboard for you to download, unzip, and open as a BlueJ project.

|  |
| --- |
| **Food Court Simulation Flow Diagram** |
| **Figure 1. (Start on the right side)**  1 person at any given moment for any given eatery.  People leave  CheckOut area  Subway  X xxxxx  1 Q that services both checkouts, 'x' is the people  X  Taco Bell  X xxxxx  These are the 2 checkout areas. Once you checked out, you are done.  xxxxxxxxxx  Some Eatery  X xxxxx  1 Q for each eater and one person being serviced.  X  Some nth Eatery  X xxxxx  1 person being serviced and rest in Q waiting.  Legend: x is a person  Blue boxes are comments  Red boxes are part of simulation  Flow of simulation |

|  |
| --- |
| **UML diagram of Project 4D (BlueJ)** |
|  |

|  |
| --- |
| **UML diagram of Project 4E (BlueJ) using Java API LinkedList, ArrayList, & PriorityQueue** |
|  |

|  |
| --- |
| **UML diagram of Project 4E (BlueJ) using Queue163 and PriorityQueue163** |
|  |

|  |
| --- |
| **SimulationPanel centered on a JFrame with BorderLayout.CENTER** |
|  |
| **Project 4 A, B, C, and D Simulation Model** |
| **Window** |
| **RandomNormal** |
| **Event163** |
| **Person** |
| **ClockListener interface--** comparable to the java.awt.event ActionListener interface |
| **PersonProducer** |
| **FoodCourt** & **Eatery** |
| **Checkout** & **Cashier** |
| **Terminate** |
| **Clock** |
| **Simulation** |

|  |  |  |
| --- | --- | --- |
| **Project 4 E Simulation View – Graphical User Interface** | | **Simulation Model** |
| **SimulationPanel c**ontains a **Simulation reference** | | **Simulation** |
|  | **FoodCourtPanel** has a **FoodCourt reference** – top left panel | **FoodCourt** |
| **EateryPanel** has an **Eatery reference** – top right panels | **Eatery** |
| **CheckoutPanel** has a **Checkout reference** – left panel | **Checkout** |
| **CashierPanel** has a **Cashier reference** – right panels | **Cashier** |
| **SimulationControlPanel** – bottom panel  Contains a **Clock reference**  Contains a **PersonProducer reference**  Contains a **Terminate reference** | **Clock**  **PersonProducer**  **Terminate** |
| **SimulationGUI** main – makes a declaration for a JFrame | |  |

As given below, the Simulation includes four service providers within the simulation: producer, food court, checkout, and terminate.

|  |
| --- |
| Simulation consists of a clock and four clock listeners |
| /\*\*  \* The clock removes Event163 events from the futureEvent priority queue and calls the  \* appropriate listener to performAction with that event, where the event contains the  \* listener block index, as well as the server index as needed.  \*  \* Coordinated by events on a futureEvents priority queue, instances of a Person class move  \* through the simulation blocks.  \*/  public class Simulation  {    public Clock **clock;** // removes events from a futureEvent queue  public PersonProducer **producer**; // A Clock listener  public FoodCourt **foodCourt**; // A Clock listener  public Checkout **checkout**; // A Clock listener  public Terminate **terminate**; // A Clock listener  public Simulation()  {  this.clock = new Clock( );  this.producer = new PersonProducer( this.clock );  this.foodCourt = new FoodCourt( this.clock );  this.checkout = new Checkout( this.clock );  this.terminate = new Terminate( this.clock );  this.producer.nextBlock = this.foodCourt;  this.foodCourt.nextBlock = this.checkout;  this.checkout.nextBlock = this.terminate;  **clock**.add(this.**producer**); // as simulation block **0**  **clock**.add(this.**foodCourt**); // as simulation block **1**  **clock**.add(this.**checkout**); // as simulation block **2**  **clock**.add(this.**terminate**); // as simulation block **3**  }  } |

An instance of the SimulationPanel, given below, contains a field for a **Simulation** reference. The SimulationPanel constructor invokes a method to create the **simulation** object. The Food Court simulation starts with one eatery and one cashier.

|  |
| --- |
| SimulationPanel contains a **Simulation** reference to an existing **simulation** object, |
| public class SimulationPanel extends JPanel  {  **private Simulation simulation;**  public SimulationPanel( . . . )  {    **this.createSimulation();**    . . .  }  /\*\*  \* The createSimulation method creates an instance for a food court Simulation., and by default  \* it includes one Eatery and one Cashier.  \*/  private void createSimulation()  {  **simulation = new Simulation();**  double estimateOrderTime1 = 24;  Eatery food = new Eatery(simulation.clock, "Flavors", estimateOrderTime1 );  **simulation.foodCourt.add( food );**  double estimateCheckoutTime1 = 12;  **simulation.checkout.addCashier(estimateCheckoutTime1);**  }  } |

|  |
| --- |
| Before you do TODO 1, and certainly before TODO 23, please make the following 3 changes, marked in RED, to your Terminate class. Your version in the startup inadvertently has the changes already made that you need to do as TODOs 45-47. |
| import java.awt.\*;import java.text.DecimalFormat;import java.util.\*;/\*\*\* @author \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\* @version date for today\*/public class Terminate implements ClockListener {public Window window;public Clock clock; // the clock is shared, clock.time is current simulation timeprivate int blockIndex; // this block indexprivate Person person; // the person yp DEPART the simulation// TODO 45 Comment out the next two lines.private ArrayList<Person> PQ; // a person queue for waiting in linepublic ArrayList<Event163> EQ; // an event queue of Event163 events// TODO 45 Uncomment out the next two lines.// private Queue163<Person> PQ; // a person queue for waiting in line// public Queue163<Event163> EQ; // an event queue of Event163 eventspublic int departures; // a count of persons that DEPART from the simulationprivate int maxPQlength;public Terminate(Clock clock) {this.window = new Window("Terminate", 350, 25, 1125, 200, Color.BLUE, false);this.clock = clock;// TODO 46 Comment out the next two lines.this.PQ = new ArrayList<Person>();this.EQ = new ArrayList<Event163>();// TODO 46 Uncomment out the next two lines.// this.PQ = new Queue163<Person>();// this.EQ = new Queue163<Event163>();this.maxPQlength = 0;this.person = null;}public void add (Person person){this.PQ.add(person);if (this.PQ.size() > this.maxPQlength){this.maxPQlength = this.PQ.size();}}public void performAction(Event163 e){this.blockIndex = e.blockIndex;if (e.eventType.equals( "DEPART" )){if (this.person != null){this.EQ.add( e ); //add the event to the selected server EQ}else if ( this.PQ.size() >= 1) {this.person = this.PQ.remove(0);this.person.blockIndex = this.blockIndex;this.report( "DEPARTS ", Color.BLUE, this.person );this.person = null;this.departures++;if ( this.PQ.size() >= 1){this.performAction( this.EQ.remove( 0 ) );}}}}// TODO 47 Comment out the following method.public String PQtoString(){String str = " PQ "+ this.PQ.size() + ": " ;for (Person p: this.PQ){str += p.trackingNumber + " ";}return str;}// TODO 47 Uncomment out the following method.// public String PQtoString()// {// String str = " PQ "+ this.PQ.size() + ": " ;// PQ.begin();// while ( PQ.hasNext())// {// str += PQ.next().trackingNumber + " ";// }// return str;// }private void report( String eventType, Color color, Person person ){String trn = person.trackingNumber + person.attention + " ";String evt = eventType + " ";String clk = Clock.inSeconds(this.clock.time) + " ";if (this.window != null){this.window.setColor( color );this.window.println( trn + evt + clk + person.toString());}}public void showStatistics(){this.window.println();this.window.println( "Terminate Statistics" );this.window.println( "Simulation block: " + "B" + this.blockIndex );this.window.println( "Clock time:\t" + Clock.inSeconds(this.clock.time) );this.window.println( "Departures:\t" + this.departures );this.window.println();} } |

## What you need to do

**SimulationGUI** class. Study to understand. Otherwise, done.

**SimulationPanel** class. Study to understand. Otherwise, done.

**FoodCourtPanel** class. **TODOs 1-5**, mostly in the ActionListener class. Study to understand. Review the Grid layout. Observe that this class has a **reference to foodCourt.**

**EateryPanel** class. **TODOs 6-9**, mostly in the ActionListener. Study to understand. Review the Grid layout. Observe that this class has a **reference to eatery**.

**CheckoutPanel** class. **TODOs 10-14**, mostly in the ActionListener. Study to understand. Review the Grid layout. Observe that this class has a **reference to checkout.**

**CashierPanel** class. **TODOs 15-18**, mostly in the ActionListener. Study to understand. Review the Grid layout. Observe that this class has a **reference to cashier.**

**SimulationControlPanel** class. **TODOs19-21**, mostly in the ActionListener. Study to understand. Review the Grid layout. Observe that this class has controls that are declared as fields for proper functionality. These controls are added to internal JPanels, which are used to organize of the view. Study how the javax swing timer is used to implement continuous event simulation, verses discrete event simulation.

Observe that this class has a **reference to simulation**, and hence **simulation.clock**,

**simulation.clock.window**, **simulation.producer**, and **simulation.terminate**.

**Clock** class. **TODO 22 & 23 The SimulationGUI should be fully functional.**

**LinkedList163**  class. **TODO 24 - 29** The SimulationGUI should be fully functional.

**Queue163**  class. **TODO 30 & 31** The SimulationGUI should be fully functional.

**PriorityQueue163**  class. **TODO 32 - 35** The SimulationGUI should be fully functional.

**Clock** class. **TODO 36 & 37**

**Eatery**  class. **TODO 38 - 40**

**Checkout**  class. **TODO 41 - 44**

**Terminate class. TODO 45 - 47**

**The SimulationGUI should be fully functional.**

**Before you do the following, create a backup copy of your Project 4 E!**

**Make a change in the Eatery class.**

First of all observe that in the PersonProducer class, the attention field for each person was checked as special:

10 % as Person.ASSIST , i.e. "A";

20% as Person.BUSINESS,i.e. "B";

70% as Person.COACH, i.e. "C";

Modify your performAction method in the Eatery class so that a person checked as Person.ASSIST will have timeToOder calculated by randomNormal with 2.0 \* estimateServiceTime and a person checked as Person.BUSINESS will have their timeToOder calculated by randomNormal with 0.5 \* estimateServiceTime.

**The SimulationGUI should be fully functional.**

# What to hand in.

* Submit you solution to Backboard as a zipped file.
* Be ready to demonstration your solution to Project 4E in lab on 18 April 2017.
* **One** printed copy per group of the following classes, with all names in the group listed as the authors:

1. Event163
2. Person
3. ClockListener
4. PersonProducer
5. FoodCourt
6. Eatery
7. Checkout
8. Cashier
9. Terminate
10. Clock
11. Simulation
12. SimulationPanel
13. FoodCourtPanel
14. EateryPanel
15. CheckoutPanel
16. CashierPanel
17. SimulationControlPanel
18. SimulationGUI