REPORT

JAVA Programming and Labs

Student ID : 1771008
Name : Minjeong Kim

HW3_1.java : main method class

[Code]

```java
package HW3_1;

public class HW3_1 {

    public static void main(String[] args) {

        //creates a Bakery object
        Bakery bObj = new Bakery();

        //create Producer objects
        Producer pObj1 = new Producer(bObj, 1);
        Producer pObj2 = new Producer(bObj, 2);
        //create Consumer objects
        Consumer cObj1 = new Consumer(bObj, 1);
        Consumer cObj2 = new Consumer(bObj, 2);

        //start treads!
        pObj1.start();
        pObj2.start();
        cObj1.start();
        cObj2.start();
    }
}
```

[Output console]

```
Console  Problems  Javadoc  Declaration  Debug
<terminated> HW3_1 [Java Application] C:₩Program Files₩Java₩jre-10.0.
Producer 1 put: 8 (Total # of Breads = 8)
Consumer 1 got: 5 (Total # of Breads = 3)
Producer 2 put: 7 (Total # of Breads = 10)
Consumer 2 got: 5 (Total # of Breads = 5)
Producer 2 put: 5 (Total # of Breads = 10)
Producer 2 put: 4 (Total # of Breads = 14)
Consumer 2 got: 5 (Total # of Breads = 9)
Consumer 1 got: 5 (Total # of Breads = 4)
Producer 1 put: 4 (Total # of Breads = 8)
Producer 2 put: 6 (Total # of Breads = 14)
Consumer 2 got: 5 (Total # of Breads = 9)
Producer 1 put: 10 (Total # of Breads = 19)
Producer 1 put: 4 (Total # of Breads = 23)
Producer 2 put: 2 (Total # of Breads = 25)
Consumer 1 got: 5 (Total # of Breads = 20)
Producer 2 put: 3 (Total # of Breads = 23)
Producer 2 put: 2 (Total # of Breads = 25)
Consumer 2 got: 5 (Total # of Breads = 20)

:Done
=> Consumer 2 won the game! (first ate 20 breads)
```

[Code analysis]

the number of Bakery object = Bakery shop's number

in this simulation, the bakery is only one. therefore, creates a Bakery object

the number of Producer object = Producer's number

the number of Consumer object = Consumer's number

in this simulation, the number of each Producer and consumer is two.

therefore, creates two Consumer objects, and two Producer objects.

the Producer and Consumer class are extended by thread class. So, when the processing executing, to run the thread use start() method on each objects.

Consumer.java

[Code]

```java
package HW3_1;

public class Consumer extends Thread { //The Consumer class
    private Bakery bakery; // to use synchronize, using one bakery object
    private int number; // consumer id
    private int bread; // in processing, the number of total consuming breads, which is consumed by this consumer object

    public Consumer(Bakery bakery, int number) { //constructor. to initialize
        this.bakery = bakery; //to adjust thread, using one bakery object. (int this homework, the bakery parameter is same)
        this.number = number; // set consumer id
        this.bread = 0; // in init, the number of sum consuming breads must set zero. (because counting)
    }

    public int getBread() { //to access the bread element at the bakery class
        return bread; //get bread element
    }

    public void run() { //In the main function, if consumer objects start, execute this method and add thread.
        while(true) {//use infinite loop, but break the loop and process exit, when the satisfy the condition
                     //(reach the number of goal consuming breads)
            bread += bakery.get(number, this);//at a time, the number of sum consuming breads increase 5,
                                              //which is the return value of bakery.get() method.
            try {
                sleep((int)(Math.random()*100)); // if I don't set sleep randomly. the order of processing is same
                                                 //(like : p1 -> p2 -> c1 -> c2)
            }
            catch (InterruptedException ex) { // set exception
                System.out.println(ex); // to know content of exception, use print method.
            }
        }
    }
}
```

[Code analysis]

to create our own thread object, subclassing the thread class and instatntiating a new object of that class

<field>

- Bakery bakery = actually, the Bakery class controls thread by synchronized annotation. because I simulate one bakery's consuming and producing process, I make only one Bakery object in HW3_1.java.
- number = consumer id
- bread = one consumer object have their total number of consuming breads.

<constructor>

- Consumer(Bakery bakery, int number)
  in the bakery field, this is where the consumer object is located bakery. And to use number set the consumer id.

<method>

- getBread() : to access the bread element at the bakery class, to analyze final condition.

- run()
  if the consumer objects started in main function, this run() method is executed. in the run method, get() method is executed. when the get method is executed, consumer consuming 5 breads at a time. So, have to add the bread value 5 (use get method's return value.). And, this run() method is executed randomly, since the behavior, consume the breads, is occurred randomly.

Producer.java

[Code]

```java
package HW3_1;

public class Producer extends Thread{//The Producer class

    private Bakery bakery; //to use synchronize, using one bakery object
    private int number; //producer id

    public Producer(Bakery bakery, int number) { //constructor. to initialize
        this.bakery = bakery; //to adjust thread, using one bakery object. (in this homework, the bakery parameter is same)
        this.number = number; //set producer id
    }

    public void run() {//In the main function. if producer objects start, execute this method and add thread.
        while(true) { //use infinite loop, but break the loop and process exit, when the satisfy the condition
                      //(reach the number of goal consuming breads)
            int bread = (int)(Math.random()*10+1); //at a time, the number of generating breads are random, which bound is 1 to 10.
            bakery.put(number, bread); // by using bakery class, I can adjust only one thread at a time (synchronize)
            try {
                sleep((int)(Math.random()*100)); // if I don't set sleep randomly. the order of processing is same
                                                 //(like : p1 -> p2 -> c1 -> c2)
            }
            catch (InterruptedException ex) { // set exception
                System.out.println(ex); // to know content of exception, use print method.
            }
        }
    }
}
```

[Code analysis]

to create our own thread object, subclassing the thread class and instatntiating a new object of that class

<field>

- Bakery bakery = actually, the Bakery class controls thread by synchronized annotation. because I simulate one bakery's consuming and producing process, I make only one Bakery object in HW3_1.java.
- number = producer id.

<constructor>

- Producer(Bakery bakery, int number)

  in the bakery field, this is where the producer object is located bakery. And to use number set the producer id.

<method>

- run()

  if the producer objects started in main function, this run() method is executed. in the run method, put() method is executed. before the put() method executed, I have to set the producing breads' number. this producing breads' number is random, because it is different that producer generate breads' number at a time. And, this breads variable's value, randomly assigned, is settled by put() methods parameter. Also, this run() method is executed randomly, since the behavior, produce the breads, is occurred randomly.

Bakery.java

[Code]

```java
package HW3_1;

public class Bakery { //actually, the bakery class controls threads
    public static int contents; // on bakery have one total breads num. so, I using class value, which have one data in one class
    private boolean available = false;
    //when available true, the time that consumer's can consume.
    //when available false, the time that consumer's can't consume.

    public synchronized int get(int who, Consumer con) {  //consumer
        while(available == false) { //when available false, it is the time that consumer's can't consume breads
            try {
                wait(); //because it is the time consumer's can't consume breads , the consumer object's thread must be wait.
                        //to doing this i use the get method's, since wait method must execute in synchronize
            }catch(InterruptedException ex) {//exception
                System.out.println(ex);//print exception content
            }
        }
        //available is true = the time that consumer's can consume
        contents -= 5; //consumer consuming 5 breads at a time. to update the total number of breads, assign the contents variable.
        System.out.println("Consumer "+ who + " got: " + 5 + " (Total # of Breads = " + contents + ")"); //print the console to satisfy the output format
        if(contents>=5) { //if the contents is greater or equal than 5, consumer could consuming the bread.
            available = true; //when available true, it is the time that consumer can consume breads
        }else { //if the contents is not greater than 5, consumer couldn't consuming the bread.
            available = false; //when available false, it is the time that consumer can consume breads.
        }
        if(con.getBread() + 5 == 20) { //since the getBread's return value is the number of consuming breads, before this get method executing. so set this condition.
            System.out.println("\n:Done"); //print the console to satisfy the output format.
            System.out.println("=> Consumer " + who + " won the game! (first ate 20 breads)"); //print the console to satisfy the output format.
            System.exit(0); //the process end condition is satisfied, the program exit.
        }else { //the case that consumer dosen't satisfy the number of the consuming breads
            notifyAll(); //because this tread is ended, this request lock and one of the other thread start
        }
        return 5; //to update the con object's breads number. return 5 (consuming breads at a time)
    }

    public synchronized void put(int who, int value) { //producer
        contents += value; //producer generate value(random) at a time. to update the total number of breads, assign the contents variable
        if(contents>=5) { //if the contents is greater or equal than 5, consumer could consuming the bread.
            available = true; //when available true, it is the time that consumer can consume breads
        }else { //if the contents is not greater than 5, consumer couldn't consuming the bread.
            available = false; //when available false, it is the time that consumer can consume breads.
        }
        System.out.println("Producer "+ who + " put: "+ value + " (Total # of Breads = " + contents + ")");//print the console to satisfy the output format
        notifyAll(); //because this thread is ended, this request lock and one of the other thread start
    }
}
```

[Code analysis]

this bakery class's method is construct by synchronized method. by using this synchronized method, computer use only one synchronized method at a time. therefore, the HW3_2.java process multi thread, however active thread is one, and other threads wait status.

<field>

- contents = the total number of breads in this bakery objects. this variable value is settled by the consumers' consuming and producers' generating.
- available = to adjust synchronized status, use this value. the meaning of true is the state that consumer can consume breads, the get method's statement can occurred. And the meaning of false is the state that consumer can't consume breads, the get method's statement can't occurred.

<method>

- get()

the contents value decreases by 5 to update the total breads in bakery, since consumer consumes 5 breads at a time. and set the available, according to contents is not greater than 5. if the contents are greater or equal than 5, consumer can consume, so available set true. else if the contents are not greater than 5, consumer can't consume, so available set false. and this consumer's action is ended, to pass over other action, by calling the notifyAll() method. and return 5 to update this consumer object's total consuming breads number. And I set the exit condition by accessing this consumer object's total consuming breads value. so, compare the finish value with the total consuming breads, after this method is ended, value. If the condition is satisfied, print the winner and process exit. the reason why I compare the value, after this method is ended, next. If don't do that, can occur other objects thread running, so print the method message, despite of the ended condition already satisfied.

this method is actually executed when available true. in the other case, the status of object, executed this get() method, is wait() in thread. so, to check this using while loop, set wait() when available false.

- put()

the contents value increases by value(it is randomly generated value in producer class's run method) to update the total breads in bakery, since producer generates value at a time. And, also get method, check the contents are greater than 5, and according to condition, assign the true or false. By doing this, this poducer's actions is ended, so pass over other action call the notifyAll() method.

this method is actually executed always but it is controlled by the condition that consumer can consume or not. This is possible, because of the presence of wait() in synchronized method.

What I learned from this homework3_1

<Thread and Synchronized>

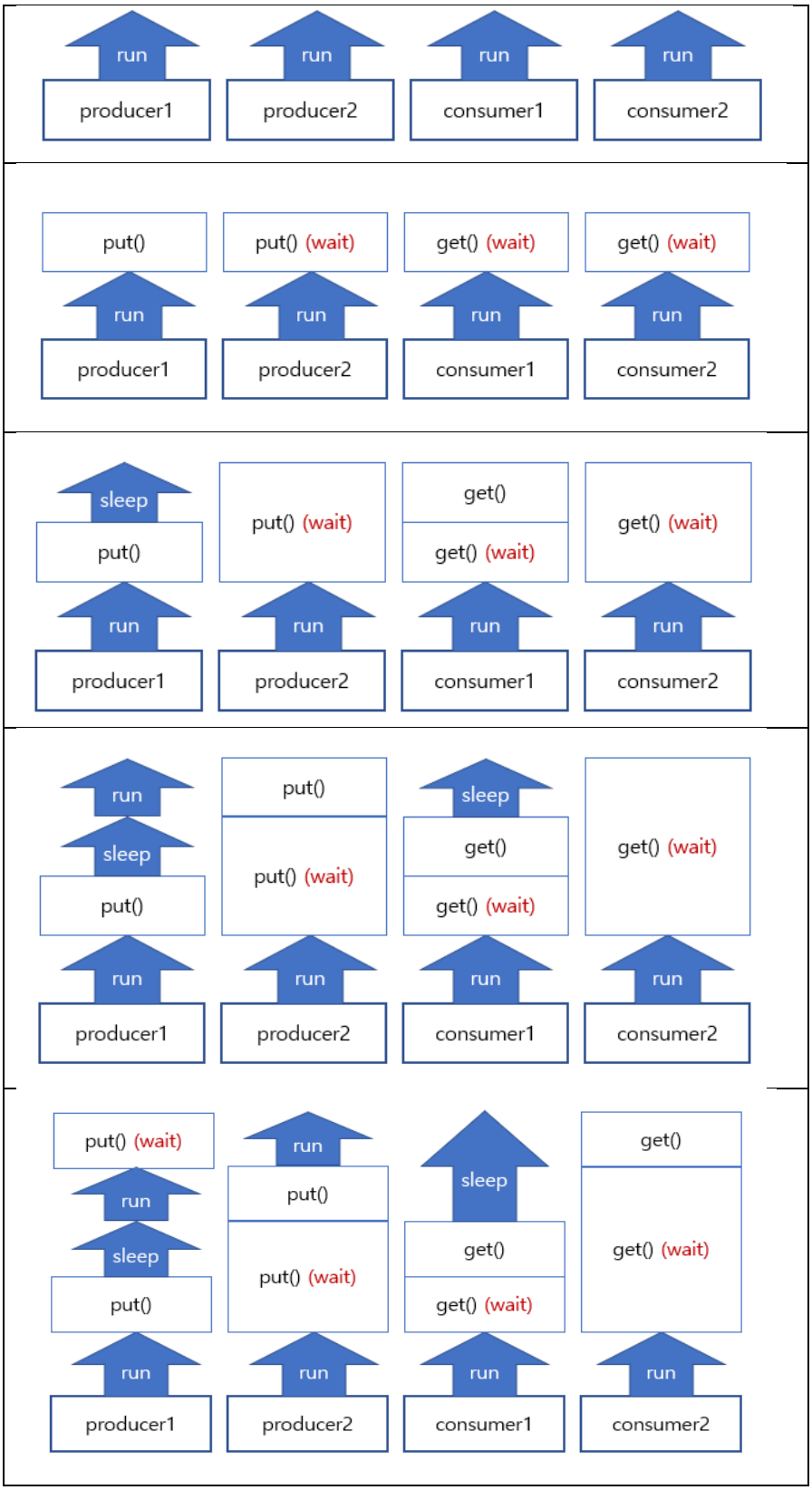
Before the explaining, set concept clearly.
process is running program. A process contains multiple threads : several tasks occur at the same time.

First, I really confusing two concept, thread's sleep() and synchronized method. however. by doing this homework, I set the concept of these clearly.

Thread start() : the threads that executed start() method are all in process. (by doing run() method)
Thread sleep() : the threads that executed sleep() method are is in process, but its task are not running.
Thread died : if the thread's run() method is return, this threads is out of process.

synchronized method
all the thread is running (not died), however they wait() the order that, to process their method running.

to know easy, view the image.



like this the wait() is occur in running time, because the synchronize method is running in run() method.
and when the sleep method, the thread is don't do anything. (it is also not status that wait)

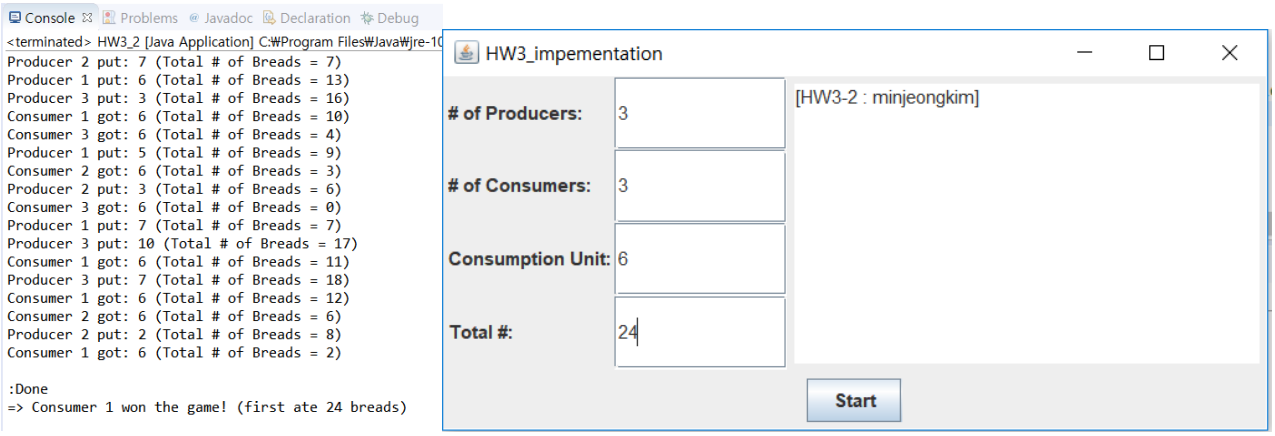
To draw easy, I participate by block, however in real, the sleep time is not block, also the put(). get() method.
so, each tasks have occur disorderly time. (not block time > like my draw) (actually have clock, but ignore).
Because of this feature, threads and synchronizations are suitable for simulation.

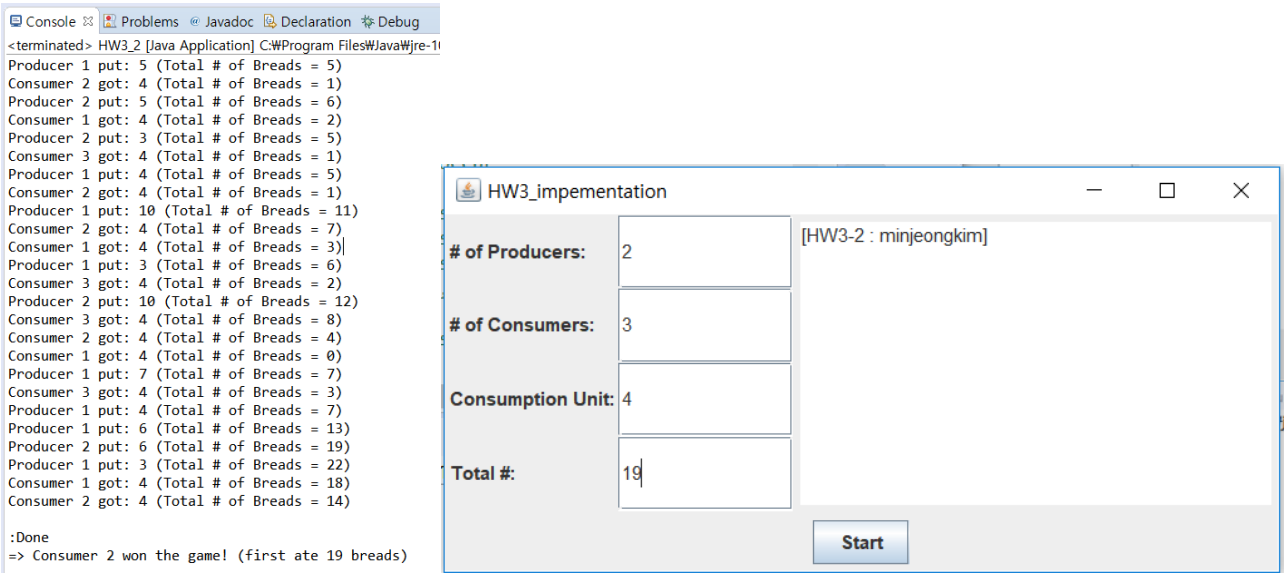
HW3_2.java : main method class

[Code]

```java
package HW3_2;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HW3_2 {

    public static int numOfPro, numOfCon, unitOfCon, total; //save the input value of TextField in this variable

    public static void main(String[] args) {
        JFrame frame = new JFrame(); //GUI frame
        JPanel p1 = new JPanel(); //input TextField's Panel
        p1.setLayout(new GridLayout(4,2)); //the Panel's format is grid 4X2

        JLabel label1 = new JLabel(" # of Producers: "); //information of first TextField
        p1.add(label1); // row 1 col 1
        JTextField field1 = new JTextField(10); // first TextField : number of Producers
        p1.add(field1); // row 1 col 2

        JLabel label2 = new JLabel(" # of Consumers: "); //information of first TextField
        p1.add(label2); // row 2 col 1
        JTextField field2 = new JTextField(10);  // second TextField : number of Consumers
        p1.add(field2); // row 2 col 2

        JLabel label3 = new JLabel(" Consumption Unit: "); //information of first TextField
        p1.add(label3); // row 3 col 1
        JTextField field3 = new JTextField(10); // Third TextField : Consumption Unit
        p1.add(field3); // row 3 col 2

        JLabel label4 = new JLabel(" Total #: "); //information of first TextField
        p1.add(label4); // row 4 col 1
        JTextField field4 = new JTextField(10); // Forth TextField : Total number of bread (the winning condition)
        p1.add(field4); // row 4 col 2

        JPanel p2 = new JPanel(); // button Panel
        JButton button = new JButton("Start"); //button init, the button value is Start
        p2.add(button); //the button is in panel

        JPanel p3 = new JPanel(); // TextArea Panel
        JTextArea textA1 = new JTextArea(10,30); //set TextArea. size is in the bracket
        textA1.setText("[HW3-2 : minjeongkim]"); //setting message in this area, before user typing,
        p3.add(textA1); //the textArea is in panel

        frame.setLayout(new BorderLayout()); //set the frame layout type
        frame.add(p1,BorderLayout.WEST); // set the panel1's location left
        frame.add(p2,BorderLayout.SOUTH); // set the panel2's location right
        frame.add(p3,BorderLayout.EAST); // set the panel3's location bottom

        frame.pack(); //the task that fit the window frame with layout of subcomponents.
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //if the GUI closed, the system is exit
        frame.setTitle("HW3_impementation"); //GUI window's title setting
        frame.setLocationRelativeTo(null); //open the GUI window in the center of screen
        frame.setVisible(true); //if don't insert this statement, the GUI window is not visible (like not open)

        button.addActionListener(new ActionListener(){ //set event listener //if the button click, this event occur
            public void actionPerformed(ActionEvent e) { //when the button click, execute this method
                numOfPro = Integer.parseInt(field1.getText()); //since this value is widely used in other classes, be settled by class variable.
                numOfCon = Integer.parseInt(field2.getText()); //since this value is widely used in other classes, be settled by class variable.
                unitOfCon = Integer.parseInt(field3.getText()); //since this value is widely used in other classes, be settled by class variable.
                total = Integer.parseInt(field4.getText()); //since this value is widely used in other classes, be settled by class variable.

                Bakery bObj = new Bakery(); //since the simulation occurred by one bakery, the bakery object is only one.
                for(int i=1; i<=numOfPro; i++) { // generate producer objects by numOfPro value
                    Producer pObj = new Producer(bObj,i); //Producer object construct
                    pObj.start(); //producer object's thread is started (the tread method run() execute)
                }
                for(int i=1; i<=numOfCon; i++) { // generate consumer objects by numOfCon value
                    Consumer cObj = new Consumer(bObj,i); //Consumer object construct
                    cObj.start(); //consumer object's thread is started (the tread method run() execute)
                }
            }
        });
    }
}
```

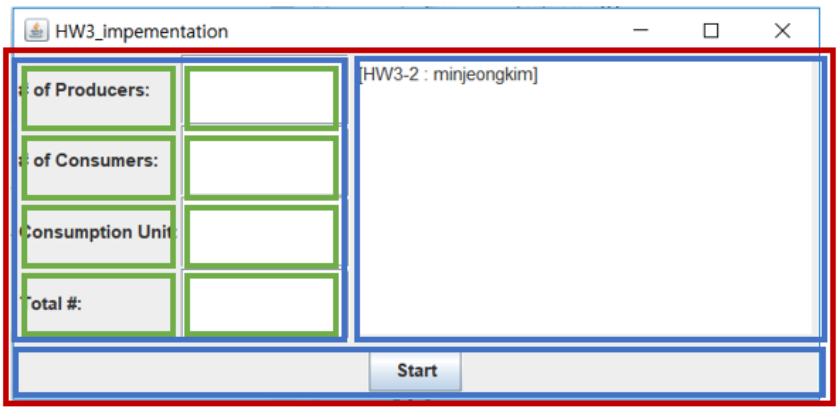
[Output console]

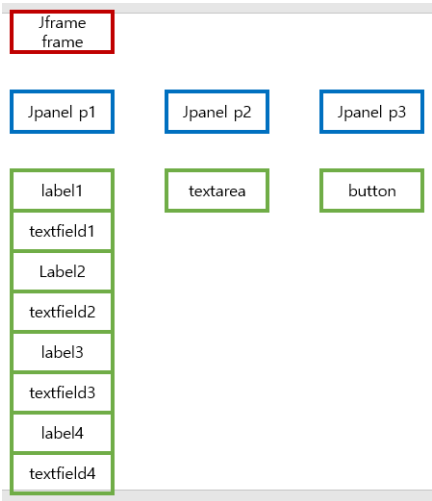


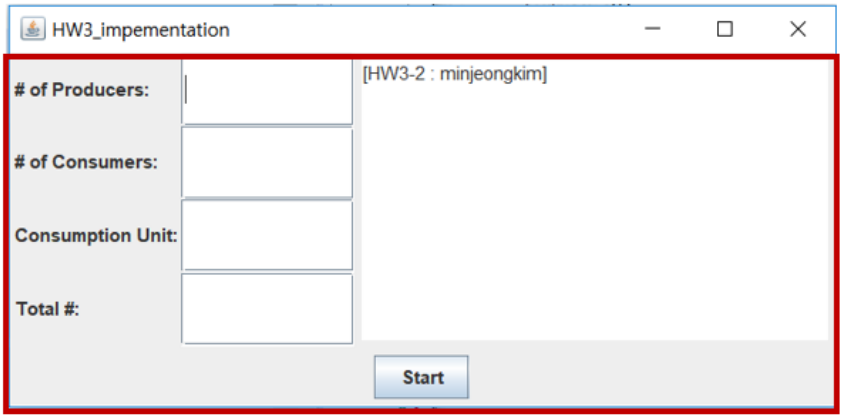


[Code analysis]

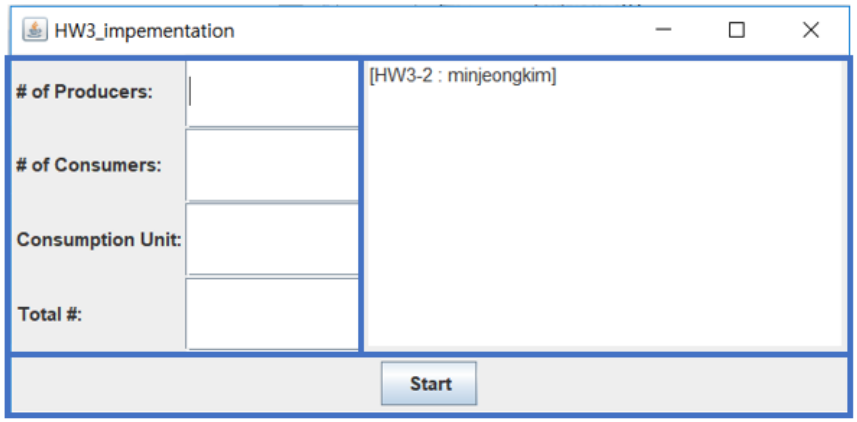
to use GUI window, make the JFrame object. I fill the frame with three panels. and to make output format, set the frame's layout and fill the JPanel objects by format. Moreover, p1 Panel have 4 labels and 4 text field. to view the layout let's look the picture



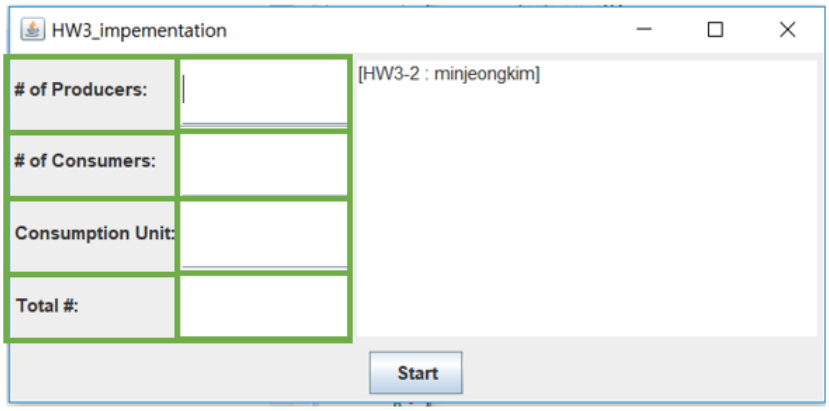




first set the frame. it is for setting GUI window open



in the frame, insert 3 components. this, layout can implement by using BorderLayout class
And, set location according to direction, components located.



in the p1 component, insert 8 componets.
this, layout can impletment by using GridLayout class
set Grid having 4 rows and 2 columns, when defining.
if components add this p1 component, the location is settled according to the grid.

And, when the button click, the Bakery simulation run according to setting from input value. to doing this, I define 4 class variable and assign the text fields' input value to each class variable. by setting class variable, I can access those variables other class and use them.

according to meaning of the input value, set the variable suitably.
First, generate Producer objects' by the value of numOfPro (# of Producers input field : the value of field1)
Second, generate Consumer objects' by the value of numOfCon (# of Consumer input field : the value of field2)
And start objects generated.

Consumer.java

[Code]

```java
package HW3_2;

public class Consumer extends Thread { //The Consumer class
    private Bakery bakery; // to use synchronize, using one bakery object
    private int number; // consumer id
    private int bread; // in processing, the number of total consuming breads, which is consumed by this consumer object

    public Consumer(Bakery bakery, int number) { //constructor. to initialize
        this.bakery = bakery; //to adjust thread, using one bakery object. (int this homework, the bakery parameter is same)
        this.number = number; // set consumer id
        this.bread = 0; // in init, the number of sum consuming breads must set zero. (because counting)
    }

    public int getBread() { //to access the bread element at the bakery class
        return bread; //get bread element
    }

    public void run() { //In the main function, if consumer objects start, execute this method and add thread.
        while(true) {//use infinite loop, but break the loop and process exit, when the satisfy the condition (reach the number of goal consuming breads)
            bread += bakery.get(number, this);//at a time, the number of sum consuming breads increase 5, which is the return value of bakery.get() method.
            try {
                sleep((int)(Math.random()*100)); // if I don't set sleep randomly. the order of processing is same (like : p1 -> p2 -> c1 -> c2)
            }
            catch (InterruptedException ex) { // set exception
                System.out.println(ex); // to know content of exception, use print method.
            }
        }

    }
}
```

[Code analysis]

It is same as HW3_1/Consumer.java (page 3)

Producer.java

[Code]

```java
package HW3_2;

public class Producer extends Thread{//The Producer class

    private Bakery bakery; //to use synchronize, using one bakery object
    private int number; //producer id
    private int bread; //in processing, producer objects generate/store breads

    public Producer(Bakery bakery, int number) { //constructor. to initialize
        this.bakery = bakery; //to adjust thread, using one bakery object. (in this homework, the bakery parameter is same)
        this.number = number; //set producer id
        this.bread = 0; // in init, the number of generating breads set zero.
    }

    public void run() {//In the main function. if producer objects start, execute this method and add thread.
        while(true) { //use infinite loop, but break the loop and process exit, when the satisfy the condition (reach the number of goal consuming breads)
            bread = (int)(Math.random()*10+1); //at a time, the number of generating breads are random, which bound is 1 to 10.
            bakery.put(number, bread); // by using bakery class, I can adjust only one thread at a time (synchronize)
            try {
                sleep((int)(Math.random()*100)); // if I don't set sleep randomly. the order of processing is same (like : p1 -> p2 -> c1 -> c2)
            }
            catch (InterruptedException ex) { // set exception
                System.out.println(ex); // to know content of exception, use print method.
            }
        }
    }
}
```

[Code analysis]

It is same as HW3_1/Producer.java (page 4)

Bakery.java

[Code]

```java
package HW3_2;

public class Bakery {//actually, the bakery class controls threads
    public static int contents; // on bakery have one total breads num. so, I using class value, which have one data in one class
    private boolean available = false;
    //when available true, the time that consumer's can consume.
        //when available false, the time that consumer's can't consume.

    public synchronized int get(int who, Consumer con) {  //consumer
        while(available == false) {//when available false, it is the time that consumer's can't consume breads
            try {
                wait(); //because it is the time that producer generate breads, the consumer object's thread must be wait.
                        //to doing this i use the get method's, since wait method must execute in synchronize
            }catch(InterruptedException ex) {//exception
                System.out.println(ex);//print exception content
            }
        }
        //available is true = the time that consumer's can consume
        contents -= HW3_2.unitOfCon; //consumer consuming unitOfCon breads at a time. to update the total number of breads, assign the contents variable.
        System.out.println("Consumer "+ who + " got: " + HW3_2.unitOfCon + " (Total # of Breads = " + contents + ")");
        //print the console to satisfy the output format
        if(contents>=HW3_2.unitOfCon) { //if the contents is greater or equal than unitOfCon, consumer could consuming the bread.
            available = true; //when available true, it is the time that consumer can consume breads
        }else { //if the contents is not greater than unitOfCon, consumer couldn't consuming the bread.
            available = false; //when available false, it is the time that consumer can consume breads.
        }
        if(con.getBread() + HW3_2.unitOfCon >= HW3_2.total) { //since the getBread's return value is the number of consuming breads,
                                                              //before this get method executing. so set this condition.
            System.out.println("\n:Done");//print the console to satisfy the output format.
            System.out.println("=> Consumer " + who + " won the game! (first ate "+HW3_2.total+ " breads)"); //print the console to satisfy the output format.
            System.exit(0); //the process end condition is satisfied, the program exit.
        }else {//the case that consumer dosen't satisfy the number of the consuming breads
            notifyAll(); //because this tread is ended, this request lock and one of the other thread start
        }
        return HW3_2.unitOfCon;//to update the con object's breads number. return unitOfCon (consuming breads at a time)

    }

    public synchronized void put(int who, int value) { //producer
        contents += value; //producer generate value(random) at a time. to update the total number of breads, assign the contents variable
        if(contents>=HW3_2.unitOfCon) { //if the contents is greater or equal than unitOfCon, consumer could consuming the bread.
            available = true; //when available true, it is the time that consumer can consume breads
        }else { //if the contents is not greater than unitOfCon, consumer couldn't consuming the bread.
            available = false; //when available false, it is the time that consumer can consume breads.
        }

        System.out.println("Producer "+ who + " put: "+ value + " (Total # of Breads = " + contents + ")");//print the console to satisfy the output format
        notifyAll(); //because this thread is ended, this request lock and one of the other thread start
    }
}
```

[Code analysis]

It is similar as HW3_1/Bakery.java (page 5)

But, it differs in that it uses the class variables of HW3_2. there are unitOfCon, and total variable.

First. unitOfCon variable's meaning is that the number of consuming breads at a time. (in HW3_1, unitOfCon is 5)

Second, total variable's meaning is that the number of finish consuming breads. (in HW3_1, total is 20)

I just replace the 5 to HW3_2.unitOfCon, and 20 to HW3_2.total.

What I learned from this homework3_2

<JAVA GUI>

I did not have a GUI lesson on that day. So, in the homework, I had many confusing, however to doing this homework, I summarized the concept of the GUI

- java.awt.*
  awt will configure the screen according to the characteristics of the OS. Therefore, the feeling varies depending on the operating system.
  example : Button, TextField,
- java.swing.*
  swing uses the look and feel used in the java realm to provide the same feeling on all operating systems.
  example : JButton, JFrame, JTextField

swing and awt can be mixed.

- Container = window : Frame, Applet etc.
- Component = the elements that are placed on the Container and responsible for screen composition.
  : Button, TextField, TextArea, List
- Layout Manager = placement method. : FlowLayout, BorderLayout, GridLayout
  + FlowLayout : By default, components are added from left to right when the container size changes, the component size is fixed and the position changes default center
  + BorderLayout : add component with location!, if don't set the location, setting center.
  + GridLayout : the form of table (is similar to HTML/CSS grid) set the (row, col)

```
frame.pack();
```

- the task that fit the window frame with layout of subcomponents.
- by doing this, we don't modulate the window size, to show the components.

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

- if the GUI window close, the system also exit
- if don't set this method, the system(server) already running, even though the GUI window is closed.

```
frame.setTitle("title set");
```

- GUI window's title setting, chrome's every tap have title.

```
frame.setLocationRelativeTo(null);
```

- open the GUI window in the center of screen
- if don't set this method, the GUI window is located at left-top side.

```
frame.setVisible(true);
```

- if don't insert this statement, the GUI window is not visible (like not open.) (show container)

<event listener format (inner)>

```
button.addActionListener(new addActionListener(){
    public void actionPerformed(ActionEvent e){

    }
});
```

ActionListener : when the event occur, it triggers execution of method. It check the actions occurring.
actionPerformed() : when the action listener check the object's action, this method is execute.

<the parameter ActionEvent e>

the parameter ActionEvent e is the contain the content of object's action.
for example, I click the button, input click button's content in variable e.
So, I got the information of action, like e.getX() can return x coordinate.

I got the e value, by

```
System.out.println(e);
```

[console]
java.awt.event.ActionEvent[ACTION_PERFORMED,cmd=Start,when=1542889333001,modifiers=Button1] on javax.swing.JButton[,235,5,61x28,alignmentX=0.0,alignmentY=0.5,border=javax.swing.plaf.BorderUIResource$CompoundBorderUIResource@88cc04d,flags=296,maximumSize=,minimumSize=,preferredSize=,defaultIcon=,disabledIcon=,disabledSelectedIcon=,margin=javax.swing.plaf.InsetsUIResource[top=2,left=14,bottom=2,right=14],paintBorder=true,paintFocus=true,pressedIcon=,rolloverEnabled=true,rolloverIcon=,rolloverSelectedIcon=,selectedIcon=,text=Start,defaultCapable=true]

for this result, I got that e contains information about the attributes of the action.

(Appendix) File Packaging

```
+HW3
  ∟HW3_1
      - HW3_1.java (main)
      - Consumer.java
      - Producer.java
      - Bakery.java
  ∟HW3_2
      - HW3_2.java (main)
      - Consumer.java
      - Producer.java
      - Bakery.java
```