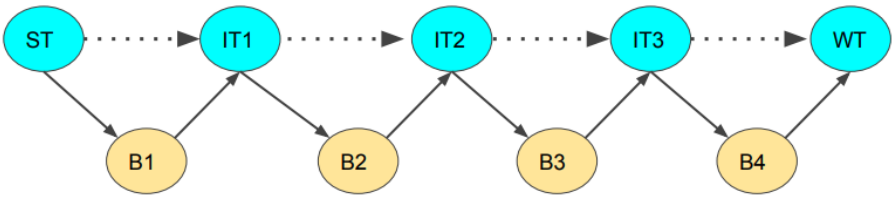
GUI
Process Simulator

Rahim Sharifov, Sara Shamilova

November 2018

1 Objective

Objective of this mini project to graphically simulate a processing chain of 5 threads in parallel according to following diagram using Graphical User Interface of Java.



To do so, we created 3 classes.

2 class Circle and Buffer

Classes of Circle and Buffer,both of which are extended from JPanel class, as it is obvious, represents Circles in our swing aplication . Class Circle represent blue circles that are TASKs, class Buffer represents orange circles that are BUFFERs.

As it appears, we will take every circle as a seperate component (or container).

2.1 Circle

Class Circle has attributes of integer x and y which represent coordinates of that circle that will show up on the window, variable color is color of circle, by default it is CYAN, but as we go through we will need to change it,so in order to be less complicated we created this attribute.String s is a attribute that is appears on the center of circle.As we have already defined variable color, Contstructor only needs to define 3 of attributes.

Method paintComponent() is predefined method, by default it displays in swing application whatever it has in its body, when application is run. So inside this method, drawOval() , fillOval() methods of class Graphics draws oval and fill it with color that is set, with arguments x,y where to draw in container and width,height of circle that will be drawn.Then prints String s on center of the circle with the color that that is set to BLACK . (Line 21-27)

Method recieve() is to simulate process of receiving message. It changes color of vareiable color and repaint() method recalls paintComponent() as we now have different color it will change the color of Circle. We use green to simulating process of receiving. (Line 29-31)

Method sleep() is to simulate sending message and sleeping randomly between 0 and 2 seconds. We use red to simulating process of sleeping. So sleep() method changes color of circle repaints it, sleeps for randomly 0-2 seconds, using sleep() method class Thread , this is why method throws InteruptedException, and it gets random number using nextInt() method of class Random.After sleeping it returns back previous estate which is color CYAN. (Line 35-41)

When message reaches to the last circle,after receiving it circle doesn't need to sleep. It recieves the message and returns back to real estate. Method displayLast() is for this purpose.It just turns circle back to color CYAN.(Line 46-48)

2.2 Buffer

Class Buffer looks like almost the same as class Circle. Reason that we created Buffer as seperate class is that in the beginning we thought that there will be much differences between TASK and BUFFER that if we use the same class for both of them it will complicated.It turned out that there is not so much differences, but still it makes program much more understandable.

So unlike class Circle , Buffer doesn't need sleep() method , because buffer is just for exchanging message, it receives and transmits it.

But like class Circle, it has the same atributes.Color is is ORANGE by default, Construct only defines 3 atributes as Circle.

Method paintComponent() draws oval and fills it,but smaller than Circle.(Line 17-23)

Method receive() is to simulate receiving message.So it turns to GREEN, sleeps for 1 seconds and then returns previous estate which is ORANGE. (Line 27-32)

3 class Main

Class Main is our main class.And inside its main method we manipulate class Circle and Buffer themselves, and their methods to simulate process chain.

We first create top-level container frame myFrame from class JFrame.This is our main container that appears on the screen when we run program.And we set background WHITE (Line 7-8)

For the Layout Manager,we decided to use GridLayout and divided our frame into 10 sections with the same size. (Line 9)

As usual we set size of our frame.And we set our from to be visible and close operation to be EXIT_ON_CLOSE which in integer form is 3.(Line 11-13)

In lines 16-22 we create array of Circles which are our TASKs and initilize them with corresponding name, coordinates where Circle appears. With the same idea in the lines 25-31 we create array of buffers.

And we add them to our frame, as we divided our frame into 10 section (2 row, 5 coloums),so as we add our inizilized components ,every component matches to the corresponting section in the order we add. (Line 36-45)

Our main task is done inside while loop. We wanted simulation to run continuously, so we write it inside infite loop.

TASK recieves the message and turns into GREEN and sleeps for 0.5 second just to show where is the message. IF IT IS NOT LAST TASK then it sends message to buffer and sleeps. To make circle sleep we created new thread to run simultaneously as event-dispatch thread. Because transmission should contuniue while previous circle sleeps. What program does is that make previous circle sleep with method of sleep that we defined above while message is sent to buffer.With method of receive buffer receive message turns into GREEN sleep for 1 second and again turns into ORANGE, then we are back to top of while loop as our counter is increased now it is second Circle. WE go though with the same idea. (Line 50-70)

IF IT IS LAST TASK (i==4) there is no buffer after TASK , TASK doesn't need to sleep. It just receives message changes its color after 1 second returns to real estate. (Line 72)

4 Conclusion

As conclusion, TASK with color CYAN means it is now free.GREEN means TASK is currently having a message. RED means after transmission TASK is sleeping. For buffer , ORANGE means buffer is currently free, GREEN means buffer currently having a message.

Idea behind this simulator is that TASK receives messages send message to the another task via buffer. Previous TASK sleeps while our transmission goes and so on.