

PARALLELISM AND CONCURRENCY

PRACTICAL ASSIGNMENT #2 WINTER TERM 2022

MONITORS

THE SUSHI BAR PROBLEM

There is a sushi bar with 5 seats. If someone arrives while there's an empty seat, that seat can be taken immediately. But if someone arrives when all 5 seats are already taken, they think all of them are dining together (they are a "group") and waits for the entire party to leave before sitting down. Incoming customers wait if there are no free seats or if they have been told to wait until the group inside completely "dissolves". Notice that for five commensals to be considered a group, an incoming customer must see them.

| SushiBarMonitor monitor |
|--|
| Client _i |
| loop monitor.enter(i) /* enjoy your sushi meal */ monitor.exit(i) |

The following is a possible trace showing the precise points when a group start and cease to exist.

```

A client enters and takes a seat (sitting: 1)
A client enters and takes a seat (sitting: 2)
A client enters and takes a seat (sitting: 3)
A client eats and leaves (sitting: 2)
A client enters and takes a seat (sitting: 3)
A client enters and takes a seat (sitting: 4)
A client enters and takes a seat (sitting: 5)
/* five people sitting but no group */
A client eats and leaves (sitting: 4)
A client eats and leaves (sitting: 3)
A client enters and takes a seat (sitting: 4)
A client enters and takes a seat (sitting: 5)
/* five people sitting but no group */
A client enters and... sees five people sitting and considers it's a group (sitting: 5; waiting: 1)
/* A group situation has just started */
A client enters and... is told to wait until the whole group leave (sitting: 5; waiting: 2)
A client enters and... is told to wait until the whole group leave (sitting: 5; waiting: 3)
A client eats and leaves (sitting: 4; waiting 3)
A client eats and leaves (sitting: 3; waiting 3)
A client eats and leaves (sitting: 2; waiting 3)
A client enters and... is told to wait until the whole group leave (sitting: 2; waiting: 4)
A client eats and leaves (sitting: 1; waiting 4)
A client eats and leaves (sitting: 0; waiting 4)
/* group situation has just finished */
A waiting client takes a sit (sitting: 1)
A waiting client takes a sit (sitting: 2)
A waiting client takes a sit (sitting: 3)
...

```

PART A (40%)

Design (pseudocode) a **single-condition** MESA-style SushiBarMonitor. The only available operations on **conditions** are waitC and signalC. Starvation is not an issue (waiting clients can overtake each other).

Translate your monitor into Java using explicit locks and Condition objects. Use the project provided. No marks will be awarded if the pseudocode is not correct. **Do not force the lock to follow a fair policy.**

In order to analyse the output generated by your program, the monitor must:

| WRITE | WHEN |
|--|---|
| ----> Entering C(id) | Just after entering the enter operation |
| +++ [free: nsb] I sit down C(id) | Just before sitting down |
| ---> now leaving [free: nsa] C(id) | Just before leaving the exit operation |
| *** Possible group detected. I wait C(id) | Just after discovering a group of five people |
| *** I'm told to wait for all free C(id) | Just before being made to wait because someone else has seen a five-people group. |

Where id refers to the id of the thread executing the operation; **nsb** refers to the number of free seats before sitting down and **nsa** refers to the number of free seats after leaving the seat.

The following images show two fragments of the output produced by a correct implementation of the monitor

```
----> Entering C(9)
+++ [free: 5] I sit down C(9)
    EATING MY SUSHI C(9)
----> Entering C(4)
+++ [free: 4] I sit down C(4)
---> now leaving [free: 4] C(9)
    EATING MY SUSHI C(4)
----> Entering C(5)
+++ [free: 4] I sit down C(5)
----> Entering C(2)
+++ [free: 3] I sit down C(2)
---> now leaving [free: 3] C(4)
    EATING MY SUSHI C(5)
    EATING MY SUSHI C(2)
----> Entering C(8)
+++ [free: 3] I sit down C(8)
---> now leaving [free: 3] C(5)
---> now leaving [free: 4] C(2)
    EATING MY SUSHI C(8)
---> now leaving [free: 5] C(8)
----> Entering C(1)
+++ [free: 5] I sit down C(1)
----> Entering C(0)
+++ [free: 4] I sit down C(0)
----> Entering C(3)
+++ [free: 3] I sit down C(3)
    EATING MY SUSHI C(1)
    EATING MY SUSHI C(0)
----> Entering C(4)
+++ [free: 2] I sit down C(4)
```

```
----> Entering C(11)
+++ [free: 1] I sit down C(11)
    EATING MY SUSHI C(5)
----> Entering C(6)
*** Possible group detected. I wait C(6)
----> Entering C(0)
*** I'm told to wait for all free C(0)
----> Entering C(18)
*** I'm told to wait for all free C(18)
    EATING MY SUSHI C(14)
---> now leaving [free: 1] C(7)
----> Entering C(10)
*** I'm told to wait for all free C(10)
    EATING MY SUSHI C(11)
----> Entering C(2)
*** I'm told to wait for all free C(2)
---> now leaving [free: 2] C(5)
    EATING MY SUSHI C(8)
----> Entering C(1)
*** I'm told to wait for all free C(1)
---> now leaving [free: 3] C(14)
---> now leaving [free: 4] C(11)
----> Entering C(16)
*** I'm told to wait for all free C(16)
----> Entering C(9)
*** I'm told to wait for all free C(9)
---> now leaving [free: 5] C(8)
+++ [free: 5] I sit down C(6)
+++ [free: 4] I sit down C(0)
+++ [free: 3] I sit down C(18)
+++ [free: 2] I sit down C(10)
+++ [free: 1] I sit down C(2)
----> Entering C(13)
*** Possible group detected. I wait C(13)
```

HOW TO HAVE YOUR TRACE ANALYSED

Before running the Launcher, right click on the Eclipse console and in preferences set a high value for the property Console buffer size (characters) (e.g. 80000000)

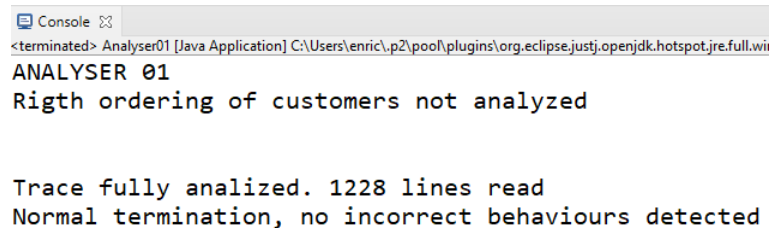
Run the launcher and wait until it terminates.

Clean the contents of the txt file Trace.txt

Copy all the text in the Eclipse console and paste it into the Trace.txt file

Run the analyser program

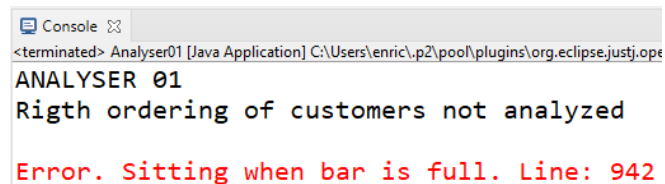
If no strange behaviour is detected the analyser will produce an output like this:



```
<terminated> Analyser01 [Java Application] C:\Users\enric\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wi
ANALYSER 01
Rigth ordering of customers not analyzed

Trace fully analized. 1228 lines read
Normal termination, no incorrect behaviours detected
```

If something goes wrong, the analyser terminates with an error message. Like this



```
<terminated> Analyser01 [Java Application] C:\Users\enric\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wi
ANALYSER 01
Rigth ordering of customers not analyzed

Error. Sitting when bar is full. Line: 942
```

Notice that the number of the line where the wrong situation was detected is shown

Always remember that not detecting errors is by no means tantamount to being free of them.

DELIVER

The pseudocode of the monitor designed. Use **bold** face and a different colour for the **waitC** and **signalC** operations

The translation of the monitor into Java (in the project provided)


PART B (30%)

Modify the design of the monitor so that it guarantees that customers sit down following a strict FIFO-policy. Provide your own ordering mechanism. You're advised to closely examine the solution of the car-park problem.

Translate the designed monitor into java. Again, do not force any lock to follow a fair policy; just translate the designed ordering monitor.

HAVE THE TRACES ANALYSED

Again a program that analyses the traces is provided. If something goes wrong regarding the expected FIFO-policy, it will produce a message like this:

A screenshot of a console window titled 'Console' with a search icon. The text inside shows a terminated Java application path, followed by 'ANALYSER 02' and 'FIFO-policy analysed'. Below this, a red message states: 'Order violation. Sitting id: 2 Expencing id: 3 Line: 2166'.

```
<terminated> Analyser02 [Java Application] C:\Users\enric\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16
ANALYSER 02
FIFO-policy analysed
Order violation. Sitting id: 2 Expencing id: 3 Line: 2166
```

You're advised to run the launcher several times (if possible in different environments) and have the traces analysed every time.

DELIVER

The pseudocode of the monitor designed. Use **bold** face and a different colour for the **waitC** and **signalC** operations. **Highlight** in yellow the differences with the previous monitor
The translation of the monitor into Java (in the project provided)

PART C (30%)

In this part, a new type of customer is introduced: the VIP customer.

VIP customers have priority over non-VIP ones. If there are VIP customers waiting to sit down, they will take a seat before any non-VIP does. Yes, VIP customers may starve non VIPs, but this is not an issue here.

Also, VIP customers do not care about groups: if there is a free seat, a VIP customer will take it even if the non VIPs are waiting for the whole group to leave the restaurant.

If there are customers (non VIPs) waiting for a group to “dissolve” and a VIP takes a seat, then the non VIPs no longer have to wait for the whole group to exit the bar (until, in the future, a new group is detected by an incoming non VIP customer)

Contrary to non VIPs, VIP customers may overtake each other (starvation of VIP customers is not an issue here)

The following fragment of a trace shows the behaviour induced by VIP customers

```
1598 *** Possible group detected. I wait C(6)
1599     EATING MY SUSHI C(8)
1600     EATING MY SUSHI C(18)
1601     EATING MY SUSHI C(11)
1602     EATING MY SUSHI C(4)
1603     EATING MY SUSHI C(7)
1604 ---> now leaving [free: 1] C(8)
1605 ---> now leaving [free: 2] C(18)
1606 ----> Entering VIPC(1)
1607 +++ [free: 2] I sit down VIPC(1)
1608 ----> Entering C(17)
1609 +++ [free: 1] I sit down C(0)
1610 ---> now leaving [free: 1] C(4)
1611 +++ [free: 1] I sit down C(16)
1612 ---> now leaving [free: 1] C(7)
1613 +++ [free: 1] I sit down C(10)
1614 ---> now leaving [free: 1] C(11)
1615 +++ [free: 1] I sit down C(9)
1616 ----> Entering VIPC(2)
1617     EATING MY SUSHI C(0)
1618 ----> Entering C(19)
1619 *** Possible group detected. I wait C(19)
1620     EATING MY SUSHI VIPC(1)
1621     EATING MY SUSHI C(9)
1622 ----> Entering C(2)
1623 *** I'm told to wait for all free C(2)
1624 ---> now leaving [free: 1] C(0)
1625 +++ [free: 1] I sit down VIPC(2)
1626     EATING MY SUSHI C(16)
1627 ----> Entering C(3)
1628 *** Possible group detected. I wait C(3)
```

In line 1598 C(6) detects a group and decides to wait. C(8) and C(18) leave the bar (lines 1604 and 1605) but C(6) keeps on waiting. In lines 1606 and 1607 a VIP customer enters and takes a seat because the “until the group dissolves” rule does not affect them. After the VIP has sat down a non VIP customer C(0) also sits down (line 1609). In line 1628 a normal customer detects a group situation again. Non VIPs will have to wait until the whole group “dissolves” or a VIP takes a seat.

When it comes to VIP customers, the monitor must:

| WRITE | WHEN |
|--|---|
| ----> Entering VIPC(id) | Just after entering the enter operation |
| +++ [free: nsb] I sit down VIPC(id) | Just before sitting down |
| ---> now leaving [free: nsa] VIPC(id) | Just before leaving the exit operation |

VIP customers do not participate in group detection nor a told to wait because of a group situation.

The messages for non VIP customers must remain the same as in the previous sections.

HAVE THE TRACES ANALYSED

Again a program that analyses the traces is provided. If something goes wrong regarding the expected VIP behaviour or the non VIP FIFO-policy, it will produce messages like these:

```
Console <
<terminated> Analyser03 [Java Application] C:\Users\sesa\.p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (
ANALYSER 03
VIP customer behaviour and FIFO-policy for non VIPS analysed

Vip preference violation. Normal customer sitting when vip waiting. Line: 7
```

```
Console <
<terminated> Analyser03 [Java Application] C:\Users\sesa\.p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (
ANALYSER 03
VIP customer behaviour and FIFO-policy for non VIPS analysed

Order violation. Sitting: 12 Expencting: 9 Line: 2129
```

```
Console <
<terminated> Analyser03 [Java Application] C:\Users\sesa\.p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (
ANALYSER 03
VIP customer behaviour and FIFO-policy for non VIPS analysed

Error. Sitting when party in bar (no vip). Line: 2254
```

DELIVER

The pseudocode of the monitor designed. Use **bold** face and a different colour for the **waitC** and **signalC** operations. **Highlight** in green the differences with the previous monitor
The translation of the monitor into Java (in the project provided)

SUBMISSION

Due date: Check eCampus

Instructions:

Solve the exercises in the given project. Rename the project (refactor...) and the folder that contains it. Use the following format for the name of the project and the folder (must to be the same for both):

PA1_Surname1Surname2_Surname1Surname2_Surname3Surname3

In the same folder, drop the document containing the pseudocode of the monitors (PDF. No scans or photos of handwriting)

Beware: the name of the folder and the name of the project must clearly identify all the authors of the practical assignment and that is why all surnames must be included.

Compress the folder and upload it.

Only one of the members does the uploading.