Concurrent and Distributed Systems

Assignment Distributed Systems

Assignment Outline

This assignment is the second of the two CSCU9V5 assessments. This assignment covers material presented during the lectures on distributed systems and builds upon the work in the three distributed practicals.

The deadline for submitting this assignment is **Tuesday, 27th November 2018, just before the CSCU9V5 lab slot at 11am.** We will use both 2-hour laboratory sessions on that day to allow you to demonstrate your solution to the given problem.
For the submission you should prepare a single document which includes a report (roughly five pages plus a cover sheet with your student number) discussing the problem[1], any assumptions you made, and a description of your solution, **as well as** the code listings of your program. The report should include appropriate diagrams of the design and screen shots of your application. Describe the various classes, their relationships, and outline the class methods. The report should explain how complete your solution is, and if applicable, any special cases when your program is not functioning correctly.
It is important that your program code is properly commented, and neatly structured for readability. You will lose marks if the code is not sufficiently commented!

In short, your assignment should consist of:
- a cover sheet giving the module, title, and student number
- a document of about 5 pages discussing the problem and your solution
- a printout of your program code with comments.

You will receive marks for:

- the efficacy of the code (basic solution)	40%
- advanced features	30%
- the report	20%
- code comments	10%

Assignment Problem

Basic Solution

This assignment will extend the theme of the 3rd distributed laboratory where you developed a socket based solution implementing a distributed mutual exclusion system using the token passing approach. In this assignment you will develop an RMI based solution for the same approach. You will be issued with a skeleton RMI based solution as a

[1] Please do not simply repeat the problem as described in this sheet!

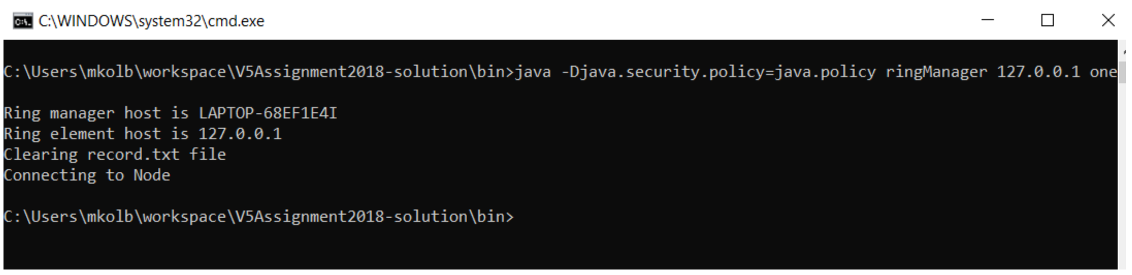
starting point. The circulating token in the issued version ensures fair access to a shared resource (e.g. a file as in the lab). It is strongly recommended that you complete the socket-based implementation from the lab first to fully understand the concepts.

In the lab you had a class implementing a ring node, and a start manager class to inject a token to start off the system. With the RMI solution you will also have a class implementing a ring node (ringMemberImpl) together with a defined interface (ringMember). The ringMember objects are RMI remote objects, so could be executed on different machines. The ringMemeber objects are connected in a chain forming a ring as in the lab (local host, local objectId, next host, next objectId). In the socket-based solution you established a connection to the next node symbolising passing the token. In RMI you will use a similar approach. Each ringMember object will implement a method takeToken() which is invoked to pass the token. Initially, the invocation symbolises the token (there is no actual token object).

Furthermore, as in the socket-based solution, the actual critical section behaviour is executed in a separate thread (criticalSection) to avoid deadlock (RMI calls are blocking). That is in the takeToken() method of a ringMember node, a criticalSection thread is initialised and started. Then the method ends. The end of the method will release the previous node. The criticalSection thread will then manipulate a shared resource (file) before it obtains the remote reference to the next ringMember object and invokes the takeToken() method.

Finally, as in the socket-based solution you will need a ringManager application to inject the token into the system to make it start to operate.

Develop a RMI token passing solution to the critical section problem based on the skeleton code provided. Sample screenshots demonstrating a setup with two ring nodes and a manager node are shown below.

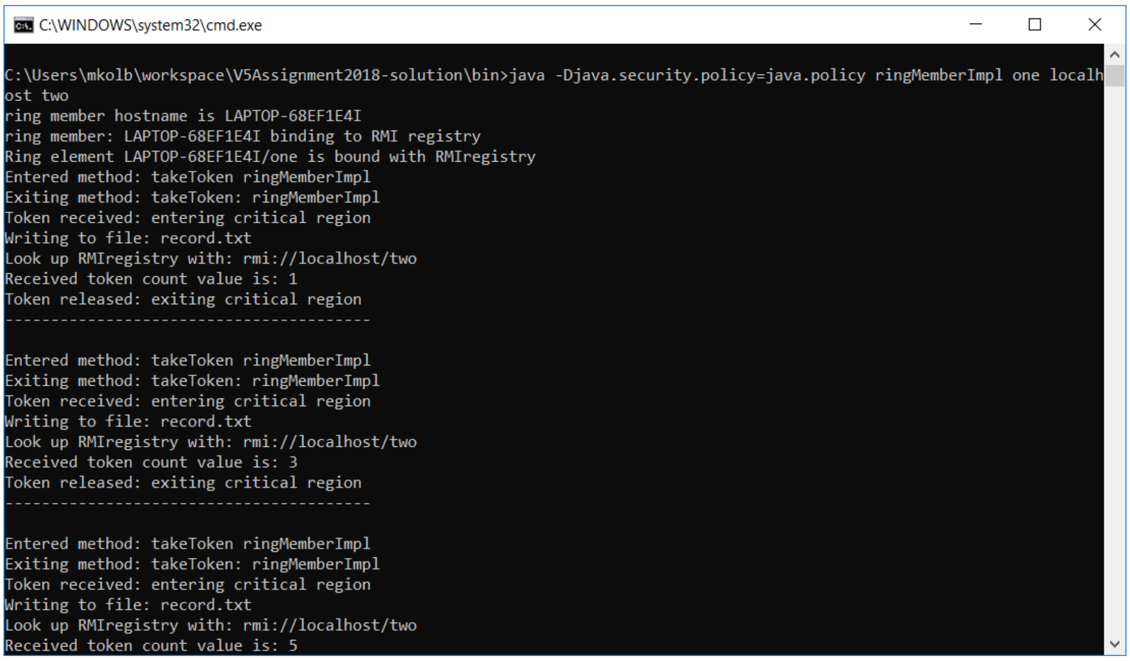


```
C:\Users\mkolb\workspace\V5Assignment2018-solution\bin>java -Djava.security.policy=java.policy ringManager 127.0.0.1 one

Ring manager host is LAPTOP-68EF1E4I
Ring element host is 127.0.0.1
Clearing record.txt file
Connecting to Node

C:\Users\mkolb\workspace\V5Assignment2018-solution\bin>
```

Manager injecting Token.



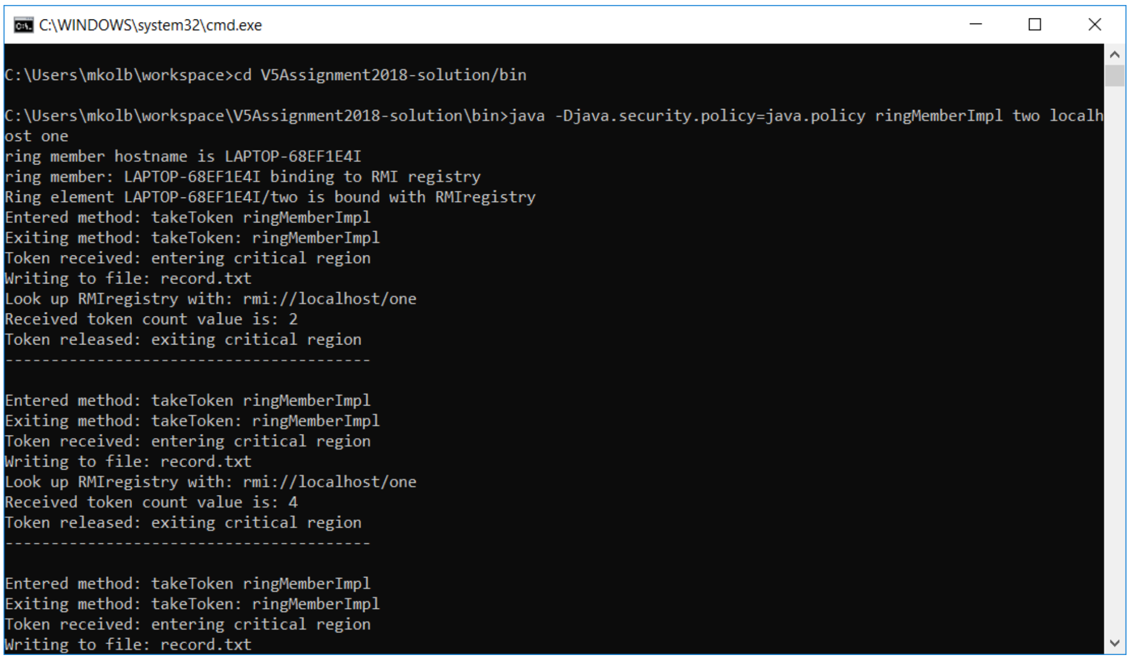
C:\WINDOWS\system32\cmd.exe

```
C:\Users\mkolb\workspace\V5Assignment2018-solution\bin>java -Djava.security.policy=java.policy ringMemberImpl one localh
ost two
ring member hostname is LAPTOP-68EF1E4I
ring member: LAPTOP-68EF1E4I binding to RMI registry
Ring element LAPTOP-68EF1E4I/one is bound with RMIregistry
Entered method: takeToken ringMemberImpl
Exiting method: takeToken: ringMemberImpl
Token received: entering critical region
Writing to file: record.txt
Look up RMIregistry with: rmi://localhost/two
Received token count value is: 1
Token released: exiting critical region
----------------------------------------

Entered method: takeToken ringMemberImpl
Exiting method: takeToken: ringMemberImpl
Token received: entering critical region
Writing to file: record.txt
Look up RMIregistry with: rmi://localhost/two
Received token count value is: 3
Token released: exiting critical region
----------------------------------------

Entered method: takeToken ringMemberImpl
Exiting method: takeToken: ringMemberImpl
Token received: entering critical region
Writing to file: record.txt
Look up RMIregistry with: rmi://localhost/two
Received token count value is: 5
```

Ring Node One



C:\WINDOWS\system32\cmd.exe

```
C:\Users\mkolb\workspace>cd V5Assignment2018-solution/bin

C:\Users\mkolb\workspace\V5Assignment2018-solution\bin>java -Djava.security.policy=java.policy ringMemberImpl two localh
ost one
ring member hostname is LAPTOP-68EF1E4I
ring member: LAPTOP-68EF1E4I binding to RMI registry
Ring element LAPTOP-68EF1E4I/two is bound with RMIregistry
Entered method: takeToken ringMemberImpl
Exiting method: takeToken: ringMemberImpl
Token received: entering critical region
Writing to file: record.txt
Look up RMIregistry with: rmi://localhost/one
Received token count value is: 2
Token released: exiting critical region
----------------------------------------

Entered method: takeToken ringMemberImpl
Exiting method: takeToken: ringMemberImpl
Token received: entering critical region
Writing to file: record.txt
Look up RMIregistry with: rmi://localhost/one
Received token count value is: 4
Token released: exiting critical region
----------------------------------------

Entered method: takeToken ringMemberImpl
Exiting method: takeToken: ringMemberImpl
Token received: entering critical region
Writing to file: record.txt
```

Ring Node Two

Advanced Features

You can enhance the basic solution by implementing more advanced features:

1) inject an actual token object that keeps a counter of how many objects it has been passed to. Output this counter in the shared file during the critical section.
2) allow the manager to accept a file name supplied by a user, and ensure this file becomes the shared resource.
3) ensure that once a token is injected into the ring it only circulates a certain number of times (defined through a parameter).
4) allow the manager (through a parameter) to direct the **n**th ring node (or a given node id) to have extra time in their critical section.
5) allow the manager (through a parameter) to direct the **m**th ring node (or a given node id) to skip *using* the token every second visit of the token. (That is, simply pass the token on every second visit without delay.)
6) allows the manager to direct all the node processes to clean up in an orderly manner.

Plagiarism

Work which is submitted for assessment must be your own work. All students should note that the University has a formal policy on plagiarism which can be found at http://www.stir.ac.uk/academicpolicy/handbook/assessmentincludingacademicmisconduct/#q-8.

Plagiarism means presenting the work of others as though it were your own. The University takes a very serious view of plagiarism, and the penalties can be severe.

Specific guidance in relation to Computing Science assignments may be found in the Computing Science Student Handbook.

We check submissions carefully for evidence of plagiarism, and pursue those cases we find. Several students received penalties on their work for plagiarism in past years. Penalties range from a reduced mark, through to no mark for the module, to being required to withdraw from studies.

Submission on CANVAS

Please ensure you submit your assignment on CANVAS before 11:00 on 27th November. This should be a single file: your report with the source code listings.

Late submission

If you cannot meet the assignment hand-in deadline and have good cause, please see Dr Mario Kolberg to explain your situation and ask for an extension. Coursework will be accepted up to seven calendar days after the hand-in deadline (or expiry of any agreed extension), but the grade will be lowered by 3 marks per day or part thereof. After seven days the work will be deemed a non-submission.