Taylor Pedretti

005488635

CSE-461

1. We have discussed in the class the implementation of the readers-writers problem in Java. However, the read and write tasks of the **reader** thread and the **writer** thread are not given. Implement these tasks in Java as reading and writing of a file named *counter.txt*, which contains an integer counter.

A **reader** thread

* reads the counter from the file, and
* prints out its thread name and the value of the counter.

A **writer** thread

* increments the value of the counter in the file,
* prints out its thread name and the new value of the counter.

Each thread repeats its task indefinitely in a random amount of time between 0 and 3000 ms. Your **main** program should create 20 **reader** threads and 3 **writer** threads.

Source Code:

readerwriter.java

**import** java**.**io**.**File**;**

**import** java**.**io**.**FileInputStream**;**

**import** java**.**io**.**FileWriter**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**Random**;**

**import** java**.**util**.**Scanner**;**

**import** java**.**util**.**concurrent**.**locks**.**Condition**;**

**import** java**.**util**.**concurrent**.**locks**.**Lock**;**

**import** java**.**util**.**concurrent**.**locks**.**ReentrantLock**;**

public class readerwriter **{**

final Lock mutex **=** **new** ReentrantLock**();**

final Condition readerQueue **=** mutex**.**newCondition**();**

final Condition writerQueue **=** mutex**.**newCondition**();**

int nReaders **=** 0**;** // number of reader threads

int nWriters **=** 0**;** // number of writer threads (0 or 1)

String file **=** "counter.txt"**;**

public static Random rand **=** **new** Random**();**

public void init**()**

**{**

FileWriter f**;**

**try**

**{**

f **=** **new** FileWriter**(new** File**(**file**));** f**.**write**(new** Integer**(**0**).**toString**());**

f**.**close**();**

**}**

**catch** **(**IOException e**)**

**{**

e**.**printStackTrace**();**

**}**

**}**

void reader**()** **throws** InterruptedException **{**

mutex**.**lock**();** // MUTUAL EXCLUSION

**while** **(!(**nWriters **==** 0**))** **{**

readerQueue**.**await**();**

// WAIT IN THE READER QUEUE UNTIL WRITERS BECOME ZERO

**}**

nReaders**++;** // ONE MORE READER

**if** **(--**nReaders **==** 0**)** **{**

writerQueue**.**signal**();** // WAKE UP A WRITING THREAD

**}**

mutex**.**unlock**();**

**}**

void writer**()** **throws** InterruptedException **{**

mutex**.**lock**();**

**while** **(!((**nReaders **==** 0**)** **&&** **(**nWriters **==** 0**)))**

**{**

writerQueue**.**await**();** // WAIT IN WRITER QUEUE

**}** // until no more writer & readers nWriters++; //one writer

// ONLY ONE WRITER AT A TIME

writerQueue**.**signal**();** // wake up a waiting writer

// readerQueue.signalAll(); //wake up all

// waiting readers mutex.unlock();

**}**

void readToFile**(**String path**)** **{**

**try** **{**

Scanner reader **=** **new** Scanner**(new** FileInputStream**(**path**));**

int x **=** reader**.**nextInt**();**

System**.**out**.**printf**(**" Counter: %d\n"**,** x**);**

**}**

**catch** **(**IOException ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

void writeToFile**(**String path**)** **{**

int counterToWrite**;**

**try** **{**

Scanner reader **=** **new** Scanner**(new** FileInputStream**(**path**));**

counterToWrite **=** **(**int**)** reader**.**nextInt**();**

counterToWrite**++;**

FileWriter f **=** **new** FileWriter**(new** File**(**path**));**

f**.**write**(new** Integer**(**counterToWrite**).**toString**());**

f**.**close**();**

System**.**out**.**printf**(**"WRITER: " **+** Thread**.**currentThread**().**getName**()** **+** " Writing... "**);**

System**.**out**.**printf**(**" Counter: %d\n"**,** counterToWrite**);**

**}** **catch** **(**IOException ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

**}**

Main.java

**import** java**.**util**.**Random**;**

public class main **{**

public final static int NUMBER\_READ\_THREAD **=** 20**;**

public final static int NUMBER\_WRITE\_THREAD **=** 3**;**

public static readerwriter readerWriterClass **=** **new** readerwriter**();**

public static Random rand **=** **new** Random**();**

static class readerThread **extends** Thread **{**

@Override

public void run**()** **{**

System**.**out**.**print**(**"Reader " **+** getName**()** **+** ":Started\n"**);**

**while** **(true)** **{**

**try** **{**

readerWriterClass**.**reader**();**

Thread**.**sleep**(**rand**.**nextInt**(**3000**));**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**

static class writerThread **extends** Thread **{**

@Override

public void run**()** **{**

System**.**out**.**print**(**"Writer " **+** getName**()** **+** ": Started\n"**);**

**while** **(true)** **{**

**try** **{**

readerWriterClass**.**writer**();**

Thread**.**sleep**(**rand**.**nextInt**(**3000**));**

**}** **catch** **(**InterruptedException ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

**}**

**}**

public static void main**(**String**[]** args**)** **{**

readerWriterClass**.**init**();** readerThread readerThreads**[]** **=** **new** readerThread**[**NUMBER\_READ\_THREAD**];**

writerThread writerThreads**[]** **=** **new** writerThread**[**NUMBER\_WRITE\_THREAD**];**

System**.**out**.**print**(**"Create/start the thread\n"**);**

**for** **(**int i **=** 0**;** i **<** NUMBER\_WRITE\_THREAD**;** **++**i**)** **{**

writerThreads**[**i**]** **=** **new** writerThread**();**

writerThreads**[**i**].**start**();**

**}** **for** **(**int i **=** 0**;** i **<** NUMBER\_READ\_THREAD**;** **++**i**)** **{**

readerThreads**[**i**]** **=** **new** readerThread**();**

readerThreads**[**i**].**start**();**

**}**

**}**

**}**

Output:

Create/start the thread

Writer Thread-0: Started

Writer Thread-1: Started

Writer Thread-2: Started

Reader Thread-3:Started

Reader Thread-4:Started

Reader Thread-5:Started

Reader Thread-6:Started

Reader Thread-7:Started

Reader Thread-8:Started

Reader Thread-9:Started

Reader Thread-10:Started

Reader Thread-11:Started

Reader Thread-12:Started

Reader Thread-13:Started

Reader Thread-14:Started

Reader Thread-15:Started

Reader Thread-16:Started

Reader Thread-17:Started

Reader Thread-18:Started

Reader Thread-19:Started

Reader Thread-20:Started

Reader Thread-21:Started

Reader Thread-22:Started

2. We discussed in class the readers-writers problem with **writers priority**, which can be solved in guarded commands:

Source Code:

Readerwriterpriority.java

**import** java**.**io**.**File**;**

**import** java**.**io**.**FileInputStream**;**

**import** java**.**io**.**FileWriter**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**Random**;**

**import** java**.**util**.**Scanner**;**

**import** java**.**util**.**concurrent**.**locks**.**Condition**;**

**import** java**.**util**.**concurrent**.**locks**.**Lock**;**

**import** java**.**util**.**concurrent**.**locks**.**ReentrantLock**;**

public class readerwriterpriority **{**

final Lock mutex **=** **new** ReentrantLock**();**

final Condition readerQueue **=** mutex**.**newCondition**();** // CONDITION VARIABLE

final Condition writerQueue **=** mutex**.**newCondition**();** // CONDITION //VARIABLE

int nReaders **=** 0**;** // NUMBER OF READER THREADS

int nWriters **=** 0**;** // NUMBER OF WRITER THREADS 0 OR 1

int nActiveWriters **=** 0**;**

// NUMBER OF THREADS CURRENTLY WRITING

String file **=** "counter.txt"**;**

public static Random rand **=** **new** Random**();**

public void init**()** **{**

FileWriter f**;**

**try** **{**

f **=** **new** FileWriter**(new** File**(**file**));** f**.**write**(new** Integer**(**0**).**toString**());** f**.**close**();**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

void reader**()** **throws** InterruptedException **{**

mutex**.**lock**();** // MUTUAL EXCLUSION

**while** **(!(**nWriters **==** 0**))** **{**

readerQueue**.**await**();**

// WAIT IN THE READER QUEUE UNTIL WRITERS BECOME ZERO

**}**

nReaders**++;** // ONE MORE READER

mutex**.**unlock**();**

readToFile**(**file**);**

mutex**.**lock**();**

**if** **(--**nReaders **==** 0**)** **{**

writerQueue**.**signal**();** // WAKE UP A WRITING THREAD

**}**

mutex**.**unlock**();**

**}**

void writer**()** **throws** InterruptedException **{**

mutex**.**lock**();**

nWriters**++;** // WRITER ARRIVED

**while** **(!((**nReaders **==** 0**)** **&&** **(**nActiveWriters **==** 0**)))**

// IF THERE ARE ANY ACTIVE READERS OR WRITERS

**{**

writerQueue**.**await**();** // WAIT IN WRITER QUEUE

**}**

// UNTIL NO MORE READERS OR WRITERS

nActiveWriters**++;** // ONE ACTIVE WRITER

mutex**.**unlock**();**

writeToFile**(**file**);**

mutex**.**lock**();**

// MUTUAL EXCLUSION NEEDED

nActiveWriters**--;**

// ONLY ONE WRITER AT A TIME

**if** **(--**nWriters **==** 0**)**

// NO MORE WAITING WRITERS

**{**

readerQueue**.**signalAll**();**

**}** **else** // HAVING WAITING WRITER

**{**

writerQueue**.**signal**();** // WAKE UP ONE WAITING WRITER

**}**

mutex**.**unlock**();**

**}**

public void readToFile**(**String path**)** **{**

**try** **{**

Scanner reader **=** **new** Scanner**(new** FileInputStream**(**path**));**

int x **=** reader**.**nextInt**();**

System**.**out**.**printf**(**"Reader: " **+** Thread**.**currentThread**().**getName**()** **+** " Reading..."**);**

System**.**out**.**printf**(**" Counter: %d\n"**,** x**);**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

public void writeToFile**(**String path**)** **{**

int counterToWrite**;**

**try** **{**

Scanner reader **=** **new** Scanner**(new** FileInputStream**(**path**));**

counterToWrite **=** **(**int**)** reader**.**nextInt**();**

counterToWrite**++;**

FileWriter f **=** **new** FileWriter**(new** File**(**path**));**

f**.**write**(new** Integer**(**counterToWrite**).**toString**());**

f**.**close**();**

System**.**out**.**printf**(**"WRITER: " **+** Thread**.**currentThread**().**getName**()** **+** " Writing... "**);**

System**.**out**.**printf**(**" Counter: %d\n"**,** counterToWrite**);**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

void readerQueue**()** **{**

**throw** **new** UnsupportedOperationException**(**"Not supported yet."**);** //To change body of generated methods, choose Tools | Templates.

**}**

**}**

Main.java

**import** java**.**util**.**Random**;**

public class main **{**

public final static int NUMBER\_READ\_THREAD **=** 1**;**

public final static int NUMBER\_WRITE\_THREAD **=** 1**;**

public static readerwriterpriority readerWriterClass **=** **new** readerwriterpriority**();**

public static Random rand **=** **new** Random**();**

static class readerThread **extends** Thread **{**

public void run**()** **{**

System**.**out**.**print**(**"Reader " **+** getName**()** **+** ": Started\n"**);**

**while** **(true)** **{**

**try** **{**

readerWriterClass**.**reader**();**

int time **=** rand**.**nextInt**(**3000**);**

Thread**.**sleep**(**time**);**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**

static class writerThread **extends** Thread **{**

public void run**()** **{**

System**.**out**.**print**(**"Writer " **+** getName**()** **+** ": Started\n"**);**

**while** **(true)** **{**

**try** **{**

readerWriterClass**.**writer**();**

int time **=** rand**.**nextInt**(**3000**);**

Thread**.**sleep**(**time**);**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**

public static void main**(**String**[]** args**)** **{**

readerWriterClass**.**init**();**

readerThread readerThreads**[]** **=** **new** readerThread**[**NUMBER\_READ\_THREAD**];**

writerThread writerThreads**[]** **=** **new** writerThread**[**NUMBER\_WRITE\_THREAD**];**

System**.**out**.**print**(**"Create/start the thread\n"**);**

**for** **(**int i **=** 0**;** i **<** NUMBER\_READ\_THREAD**;** **++**i**)** **{**

readerThreads**[**i**]** **=** **new** readerThread**();**

readerThreads**[**i**].**start**();**

**}**

**for** **(**int i **=** 0**;** i **<** NUMBER\_WRITE\_THREAD**;** **++**i**)** **{**

writerThreads**[**i**]** **=** **new** writerThread**();**

writerThreads**[**i**].**start**();**

**}**

**}**

**}**

Output:

Writer Thread-0: Started

Writer Thread-1: Started

Reader Thread-3: Started

Writer Thread-2: Started

Reader Thread-4: Started

Reader Thread-8: Started

Reader Thread-7: Started

Reader Thread-6: Started

Reader Thread-9: Started

Reader Thread-11: Started

Reader Thread-5: Started

Reader Thread-12: Started

Reader Thread-10: Started

Reader Thread-15: Started

Reader Thread-14: Started

Reader Thread-13: Started

Reader Thread 16: Started

Reader Thread-18: Started

Reader Thread-19: Started

Reader Thread-17: Started

Reader Thread-20: Started

Reader Thread-21: Started

Reader Thread-22: Started

WRITER: Thread-0 Writing... Counter: 1

WRITER: Thread-1 Writing... Counter: 2

WRITER: Thread-2 Writing... Counter: 3

Reader: Thread-3 Reading...

Reader: Thread-5 Reading...

Reader: Thread-11 Reading...

Reader: Thread-4 Reading... Counter: 3

Reader: Thread-8 Reading... Counter: 3

Counter: 3

Reader: Thread-9 Reading... Counter: 3

Counter: 3

Counter: 3

Reader: Thread-12 Reading... Counter: 3

Reader: Thread-6 Reading... Counter: 3

Reader: Thread-19 Reading... Counter: 3

Reader: Thread-10 Reading...

Reader: Thread-15 Reading...

Reader: Thread-7 Reading... Counter: 3

Counter: 3

Reader: Thread-22 Reading... Counter: 3

Reader: Thread-13 Reading... Counter: 3

Reader: Thread-16 Reading... Counter: 3

Reader: Thread-20 Reading... Counter: 3

Reader: Thread-14 Reading... Counter: 3

Reader: Thread-17 Reading... Counter: 3

Reader: Thread-18 Reading... Counter: 3

Reader: Thread-21 Reading... Counter: 3

Counter: 3

Reader: Thread-18 Reading... Counter: 3

WRITER: Thread-2 Writing... Counter: 4

Reader: Thread-8 Reading... Counter: 4

Reader: Thread-21 Reading... Counter: 4

Reader: Thread-15 Reading... Counter: 4

Reader: Thread-20 Reading... Counter: 4

Reader: Thread-6 Reading... Counter: 4

Reader: Thread-22 Reading... Counter: 4

Reader: Thread-11 Reading... Counter: 4

Reader: Thread-6 Reading... Counter: 4

Reader: Thread-8 Reading... Counter: 4

WRITER: Thread-0 Writing... Counter: 5

Reader: Thread-9 Reading... Counter: 5

Reader: Thread-3 Reading... Counter: 5

Reader: Thread-7 Reading... Counter: 5

Reader: Thread-21 Reading... Counter: 5

Reader: Thread-21 Reading... Counter: 5

WRITER: Thread-0 Writing... Counter: 6

Reader: Thread 5 Reading... Counter: 6

WRITER: Thread-1 Writing... Counter: 7

Reader: Thread-16 Reading... Counter: 7

Reader: Thread-12 Reading... Counter: 7

Reader: Thread-22 Reading... Counter: 7

WRITER: Thread-2 Writing... Counter: 8

Reader: Thread-5 Reading... Counter: 8

WRITER: Thread-1 Writing... Counter: 9

Reader: Thread-18 Reading... Counter: 9

Reader: Thread-18 Reading... Counter: 9

Reader: Thread-19 Reading... Counter: 9

Reader: Thread-4 Reading... Counter: 9

Reader: Thread-10 Reading... Counter: 9

Reader: Thread-11 Reading... Counter: 9

Reader: Thread-5 Reading... Counter: 9

Reader: Thread-17 Reading... Counter: 9

Reader: Thread-15 Reading... Counter: 9

Reader: Thread-3 Reading... Counter: 9

Reader: Thread-12 Reading... Counter: 9

Reader: Thread-22 Reading... Counter: 9

Reader: Thread-20 Reading... Counter: 9

Reader: Thread-7 Reading... Counter: 9

Reader: Thread-14 Reading... Counter: 9

Reader: Thread-8 Reading... Counter: 9

Reader: Thread-6 Reading... Counter: 9

Reader: Thread-7 Reading... Counter: 9

Reader: Thread-13 Reading... Counter: 9

Reader: Thread-17 Reading... Counter: 9

Reader: Thread-14 Reading... Counter: 9

WRITER: Thread-0 Writing... Counter: 10

Reader: Thread-8 Reading... Counter: 10

Reader: Thread-10 Reading... Counter: 10

Reader: Thread-9 Reading... Counter: 10

Reader: Thread-4 Reading... Counter: 10

Reader: Thread-20 Reading... Counter: 10

Reader: Thread-16 Reading... Counter: 10

Reader: Thread-19 Reading... Counter: 10

Reader: Thread-18 Reading... Counter: 10

Reader: Thread-13 Reading... Counter: 10

Reader: Thread-16 Reading... Counter: 10

Reader: Thread-21 Reading... Counter: 10

Reader: Thread-15 Reading... Counter: 10

Reader: Thread-8 Reading... Counter: 10

Reader: Thread-5 Reading... Counter: 10

Reader: Thread-10 Reading... Counter: 10

WRITER: Thread-2 Writing... Counter: 11

WRITER: Thread-1 Writing... Counter: 12

Reader: Thread-3 Reading... Counter: 12

Reader: Thread-8 Reading... Counter: 12

Reader: Thread 9 Reading... Counter: 12

Reader: Thread-13 Reading... Counter: 12

Reader: Thread-12 Reading... Counter: 12

WRITER: Thread-1 Writing... Counter: 13

Reader: Thread-17 Reading... Counter: 13

Reader: Thread-20 Reading... Counter: 13

Reader: Thread-11 Reading... Counter: 13

Reader: Thread-22 Reading... Counter: 13

Reader: Thread-6 Reading... Counter: 13

Reader: Thread-21 Reading... Counter: 13

Reader: Thread-5 Reading... Counter: 13

Reader: Thread-14 Reading... Counter: 13

Reader: Thread-4 Reading... Counter: 13

Reader: Thread-7 Reading... Counter: 13

Reader: Thread-18 Reading... Counter: 13

WRITER: Thread-0 Writing... Counter: 14

WRITER: Thread-1 Writing... Counter: 15

WRITER: Thread-2 Writing... Counter: 16

Reader: Thread-16 Reading... Counter: 16

Reader: Thread-17 Reading... Counter: 16

Reader: Thread-13 Reading... Counter: 16

Reader: Thread-17 Reading... Counter: 16

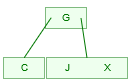
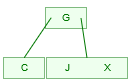
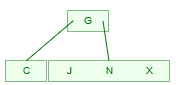
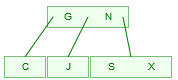
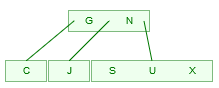
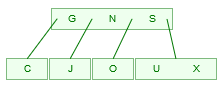
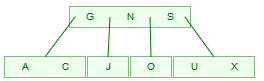
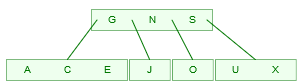
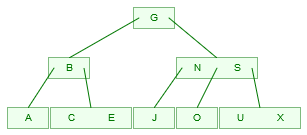
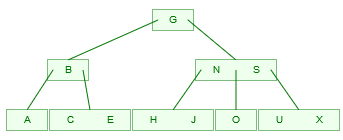
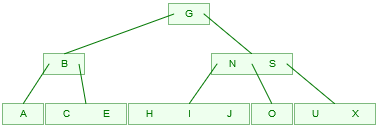
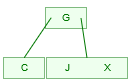
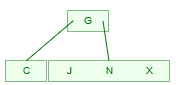
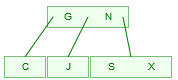
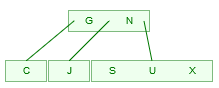
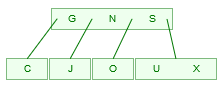
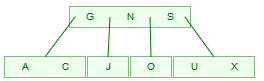
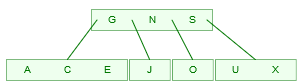
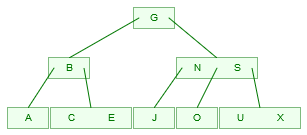
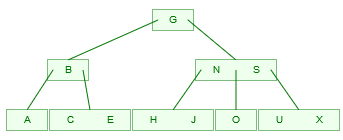
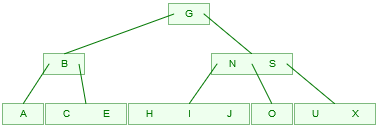
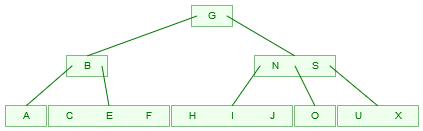
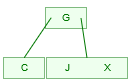
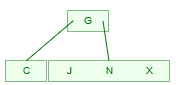
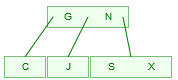
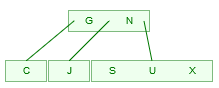
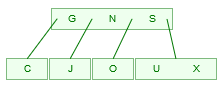
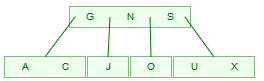
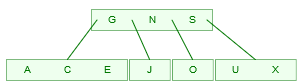
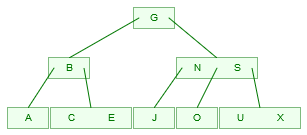
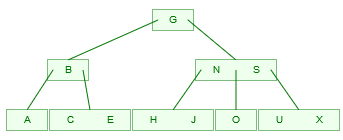
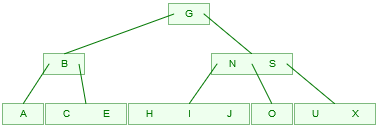
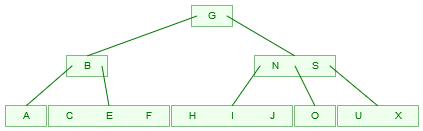
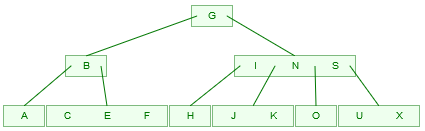
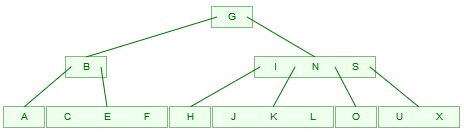
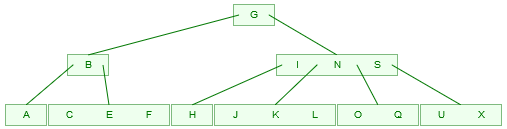
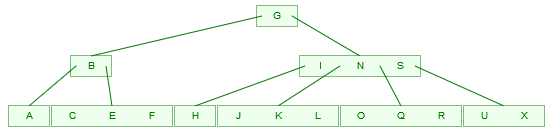
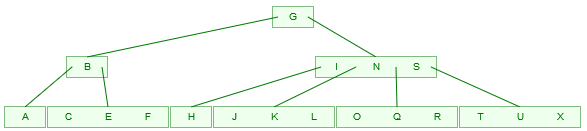
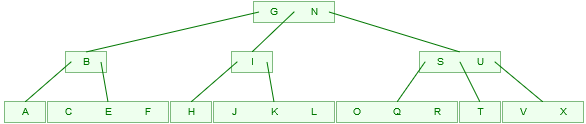
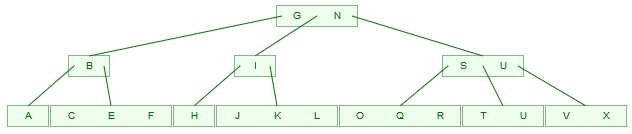
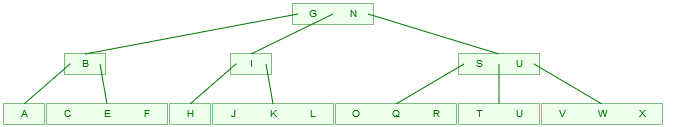
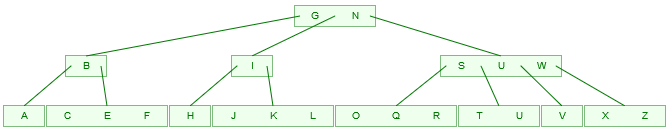
Reader: Thread-11 Reading... Counter: 16

Reader: Thread-19 Reading... Counter: 16

3. Consider a chain of processes P1, P2, ..., Pn implementing a multitiered client-server architecture. Process Pi is client of process Pi+1, and Pi will return a reply to Pi-1 only after receiving a reply from Pi+1. What are the main problems with this organization when taking a look at the request-reply performance at process P1?

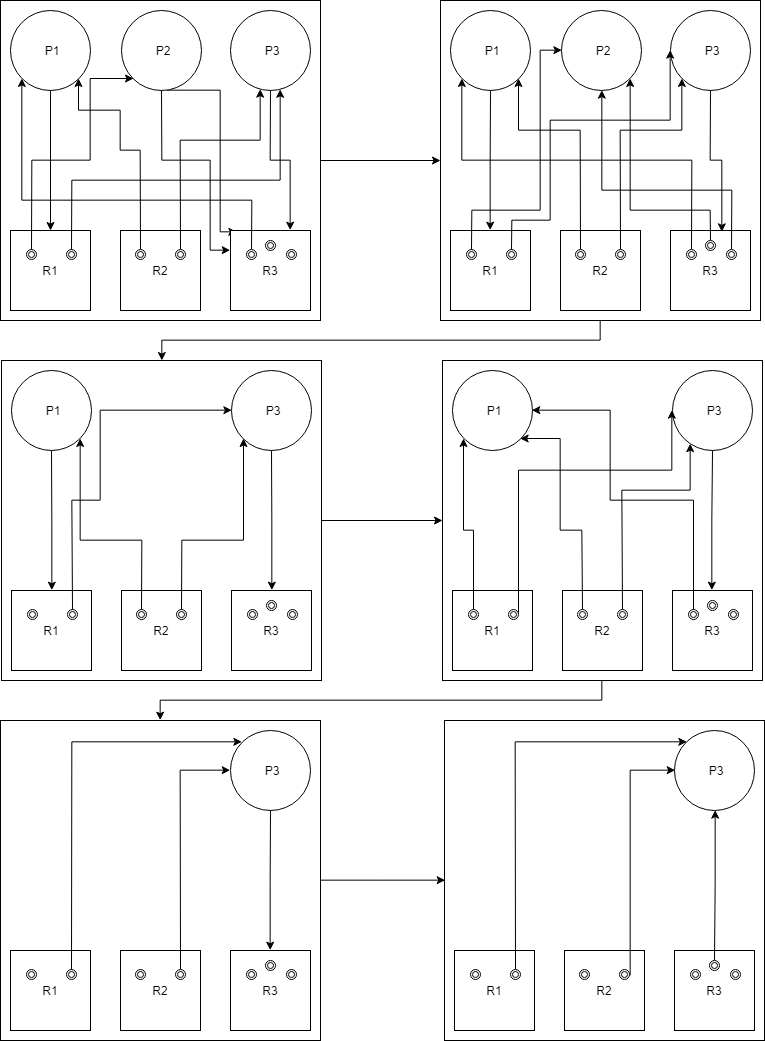
**Answer**: The larger *n* becomes the performance is expected to degrade. This is because the process relies on two different machines communicating with each other, so if one machine is performing badly or can’t be reached for some reason this immediately degrades the performance for all machines within the chain and at the highest level. Wherefore the performance between P1 and P2 could also be determined by n-2 request-reply interactions between other layers.

4. Show the B-trees of order four resulted from loading the following sets of keys (each letter is a key) in order:

1. C G J X
   1. Inserting C:
      * 
   2. Inserting G:
      * 
   3. Inserting J:
      * 
   4. Inserting X:
      * 
2. C G J X N S U O A E B H I
   1. Inserting C:
      * 
   2. Inserting G:
      * 
   3. Inserting J:
      * 
   4. Inserting X:
      * 
   5. Inserting N:
      * 
   6. Inserting S:
      * 
   7. Inserting U:
      * 
   8. Inserting O:
      * 
   9. Inserting A:
      * 
   10. Inserting E:
       * 
   11. Inserting B:
       * 
   12. Inserting H:
       * 
   13. Inserting I:
       * 
3. C G J X N S U O A E B H I F
   1. Inserting C:
      * 
   2. Inserting G:
      * 
   3. Inserting J:
      * 
   4. Inserting X:
      * 
   5. Inserting N:
      * 
   6. Inserting S:
      * 
   7. Inserting U:
      * 
   8. Inserting O:
      * 
   9. Inserting A:
      * 
   10. Inserting E:
       * 
   11. Inserting B:
       * 
   12. Inserting H:
       * 
   13. Inserting I:
       * 
   14. Inserting F:
       * 
4. C G J X N S U O A E B H I F K L Q R T V U W Z
   1. Inserting C:
      * 
   2. Inserting G:
      * 
   3. Inserting J:
      * 
   4. Inserting X:
      * 
   5. Inserting N:
      * 
   6. Inserting S:
      * 
   7. Inserting U:
      * 
   8. Inserting O:
      * 
   9. Inserting A:
      * 
   10. Inserting E:
       * 
   11. Inserting B:
       * 
   12. Inserting H:
       * 
   13. Inserting I:
       * 
   14. Inserting F:
       * 
   15. Inserting K:
       * 
   16. Inserting L:
       * 
   17. Inserting Q:
       * 
   18. Inserting R:
       * 
   19. Inserting T:
       * 
   20. Inserting V:
       * 
   21. Inserting U:
   22. Inserting W:
   23. Inserting Z:

5. Given a B Tree of order 256,

1. What is the maximum number of children from a node?
   * The maximum number of children will be 256.
2. Excluding the root and the leaves, what is the minimum number of children from a node?
   * Minimum number of children from a node is x/2 which is 128.
3. What is the minimum number of children from the root?
   * Root has 2 children other than if it is a leaf.
4. What is the maximum depth of the tree if it contains 100 000 keys?
   * To find the maximum depth we can use the formula: H\_Depth = logd((n+1)/2)
     1. d = minimum number of children: 128
     2. n = number of keys
   * H\_Depth = log128((100000+1)/2)
   * Maximum depth is: 2.2299487 but really is 2 because it cannot be in a fraction.

6. Construct a general resource graph for the following scenario and determine if the graph is completely reducible: R1, R2, and R3 are reusable resources with a total of two, two, and three units. Process P1 is allocated one unit each of R2 and R3 and is requesting one unit of R1. Process P2 is allocated one unit of R1 and is requesting two units of R3. Process P3 is allocated one unit each of R1 and R2 and is requesting one unit of R3.

**Answer**: