**Project Source code Location** - <https://github.com/Anshul-Java/Anshul-Java/blob/master/ICE.zip>

**Steps to Execute:**

* Download this zip .
* Import this into Eclipse as Maven Project.
* Based on Junit /Main Class details giving below run the classes .

**Problem 1:**

You are given a file formatted like this:

CUSIP

Price

Price

Price

…

CUSIP

Price

Price

CUSIP

Price

Price

Price

…

Price

CUSIP

Price

…

Think of it as a file of price ticks for a set of bonds identified by their CUSIPs.

You can assume a CUSIP is just an 8-character alphanumeric string.

Each CUSIP may have any number of prices (e.g., 95.752, 101.255) following it in

sequence, one per line.

The prices can be considered to be ordered by time in ascending order, earliest to latest.

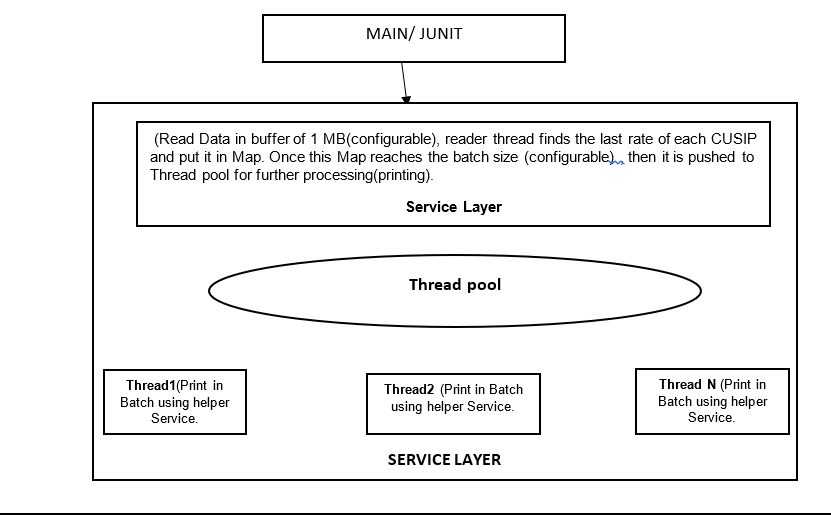
Write me a Java program that will print the closing (or latest) price for each CUSIP in the file.

DO NOT assume the entire file can fit in memory!

**Design /Implementation Details**

1. Main Method / Junit invokes the Business Service layer to process cusip Details .
2. Business Service Layer do the following :
   1. Read the file in chunk of 1 MB ( this parameter can be tuned based on UAT testing ) using buffered Reader.
   2. It will create a map of CUSIP to its last rate . So 1 cusip will have only 1 entry with its last rate.
   3. Once the Map reaches the batch size . then it will pushed the map for further processing into threadpool. Then Clear this map to optimize the memory.
   4. Thread Pool Tasks uses the Helper service to Print the data in batch.

In real project this map( batches of CUSIP’s ) will be persisted in DB , then we can use ORM ( JDBC) Batching to limit the calls to DB as well as process persisting logic concurrently .



**Classes Used:**

* IceFileReaderMain – Main class that can be called from Shell Script or from any scheduler.
* IIceBusinessService – Top layer Service Interface having main operation processCusipDetails() to process cusip details .
* IceBusinessServiceImpl – Implementation of IIceBusinessService..
* PrintCusipTask – Runnable task that uses fine grained Helper service to Print data asynchronously in batches .
* IceHelperService– is an helper operations( fine grained) interface that provide helping hand to all components.
* IceHelperServiceImpl- Implementation of IceHelperService.

**Test Cases** –

* I ran the Test case with **275 MB file** which **have 28 Millions records** , processed in ~ 30 -40 seconds with an average of **128 MB memory** . Unfortunately GIT hub is not allowing me to upload file more than 25 MB .But we can find the Junit Test class **IceBusinessServiceImpl\_LargeFileTest** in source code for this one only.
* Other Test cases are there in file **IceBusinessServiceImplTest**. It can be executed individually or by executing pom.xml for this project.

**NOTE** – Entire Project is based on Spring DI.

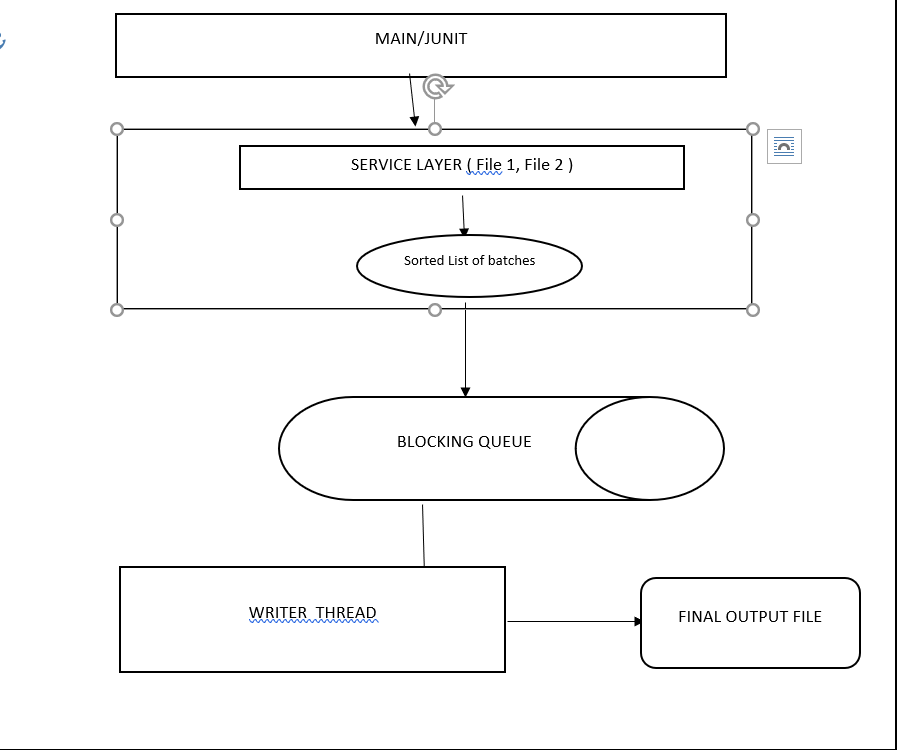
**Problem 2:**

Given two sorted files, write a Java program to merge them to preserve sort order.

DO NOT assume either of these files can fit in memory!

**Design /Implementation Details:**

* Main Class invokes the business layer to merge sort files with below parameters:
  + First File Name/Path
  + Second File Name/Path
  + Most Important Comparator Implementation (This implementation will tell how the 2 files should be compared, and because of this it makes the service implementation very generic. Now the below service can be called to merge 2 sorted files with 1 column / 2 columns or N columns ) .
* Business layer do the following:
  + Read both the files in chunk of 1 MB ( this parameter can be tuned based on UAT testing ) using buffered Reader.
  + Then compare it line by line using provided comparator and create a sorted list .
  + Once this sorted list reaches preconfigurable batch size then push this list to Blocking queue . Then clear the list for optimized memory.
  + Here we cannot use multiple threads as we need to preserver sort order , so only 1 thread is listening to this queue and writing this list into the output file.
  + Once all comparison is finished , push “EOF” Message into the queue to exist the listening thread .



**Classes Used**

* IceFileMergeMain– Main class that can be called from Shell Script or from any scheduler.
* IIceBusinessService – Top layer Service Interface having main operation mergeSortFile () to create single file from 2 sorted files .
* IceBusinessServiceImpl – Implementation of IIceBusinessService.
* WriteTask– Runnable task that uses fine grained Helper service to write data into file asynchronously .
* IceHelperService– is an helper operations( fine grained) interface that provide helping hand to all components.
* IceHelperServiceImpl- Implementation of IceHelperService

**Test Cases :**

* **MergeSortedFile1Test -**  This Junit class is used to create a final sorted files from 2 sorted files of 1 column each .
* **MergeSortedFile2Test -**  This Junit class is used to create a final sorted files from 2 sorted files of 2 columns each .
* **Memory Related test -**  In Junit of 1st problem we have already tested 275 MB file with 28 Millions rows since in this case also we are using buffered reader with buffer size of 1 MB for reading /writing, so here we have not tested large file.

**NOTE** – Entire Project is based on Spring DI.