Vaarta: An RSS reader

Vaarta is an RSS reader written in Java and JavaFX. It displays feed entries in a descending order and also shows a combination of all feeds, also i descending order, under "All feeds". It stores the feeds between application restarts and allows addition of new RSS feeds.

Each RSS feed is obtained over the network, parsed using Java DOM parser, sorted by publication time and displayed. A common thread pool is maintained for all tasks using ExecutorService. The size of the pool is fixed at the number of available CPU threads.

Team

Name	Git ID	Git email
Athul Iddya	aiddya	git@iddya.com
Prabhakar Kota	kotaprabhakar	kotaprabhakar@gmail.com
Sai Swetha Kondubhatla	swetha6	swetha.kondubhatla@gmail.com

Contributions

Athul Iddya

Code architechture, sort, merge and multi-threaded "iterative" processing code, poster diagrams

Prabhakar Kota

XML parser and multi-threaded "concurrent" processing code

Sai Swetha Kondubhatla

UI code, makefile, poster and poster diagrams

Running instructions

- Ensure that `JAVA_HOME` environment variable is set and JavaFX is installed
- Run makefile using `make`
- Run the program using `java -cp src/ appui.Main`
- `config.ini` controls the initial feeds and processing strategy
- `strategy` can be either "MULTI_ITERATIVE", "MULTI_CONCURRENT" or "SINGLE_THREADED"
- `feedCount` is the total number of feeds, this has to be decreased to use a smaller number of feeds

Single-threaded processing

- Perform network request, parsing and sorting in sequence for each RSS feed
- Display each feed using JavaFX's ObservableList
- Insert each feed's entries into a priority queue to combine them
- Display the combined entries under "All feeds" using an ObservableList

Multi-threaded, "iterative" processing

- Perform network request, parsing and sorting in parallel for as many feeds as there are threads available
- Once a feed has been parsed and sorted, display it and add it to an ArrayBlockingQueue for merging
- A parallel thread takes feeds from the ArrayBlockingQueue and merges them one by one to create "All feeds"
- Each merge waits on the application thread to finish displaying the entries to prevent concurrent modification

Multi-threaded, "concurrent" processing

- Perform network request and parsing in parallel for as many feeds as there are threads available
- Once an item has been parsed, add it to a PriorityBlockingQueue
- After a feed has been completely parsed, sort it and display it
- In parallel, take entries from the PriorityBlockingQueue, merge it with existing entries in "All feeds"
- Repeat as long as there are entries in PriorityBlockingQueue, while waiting for the application thread to finish displaying the entries in between repetitions

Results

110 RSS feeds

Processing	Time taken	Memory (peak)	CPU (peak)
Single threaded	54615	57.58MB	17.69%
"Iterative"	4040	64.37MB	61.73%
"Concurrent"	3662	56.32MB	64.35%

50 RSS feeds

Processing	Time taken	Memory (peak)	CPU (peak)
Single threaded	40433	63.13MB	17.31%
"Iterative"	1013	63.12MB	61.18%

Processing	Time taken	Memory (peak)	CPU (peak)
“Concurrent”	806	55.45MB	61.70%

10 RSS feeds

Processing	Time taken	Memory (peak)	CPU (peak)
Single threaded	548	40.09MB	19.07%
“Iterative”	691	58.07MB	33.08%
“Concurrent”	668	56.73MB	30.80%