ASE Stage 2 - Individual Report

Wanchana Ekakkharanon (H00148811)

Technical Review – Multithreading

The traditional application in Java is sequential programme that will execute the codes in a sequential step, one line at a time. Multithreading allowed your application to divided some of the big operation into multiple small operations and run them in parallel and help improve speed of your application. Java also provide the tools that allowed you to create an application that contain multiple threads in one application.

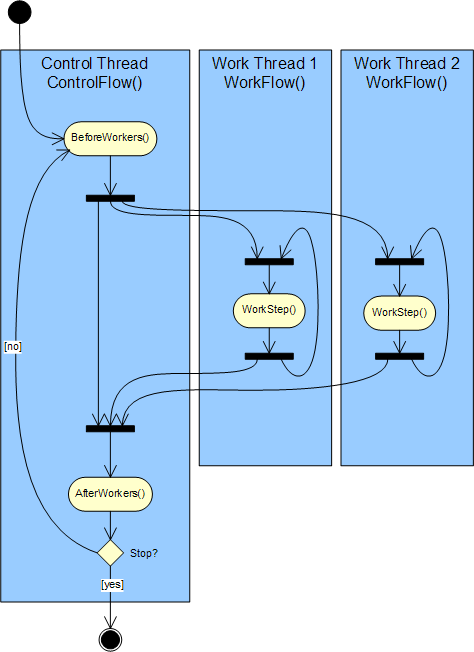
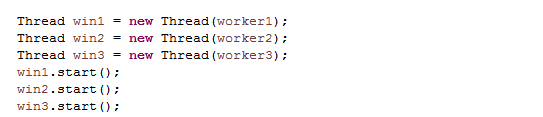


Figure 1 A diagram show how multithreading work

For example, Kiosk taxi service have multiple windows to serve a customers from one queue, all windows must be able to operate in parallel. If the code was developed as a sequential programme, only one window will operate at the time. In this case, Java provide “Thread” class that allow you to create new threads and run them in parallel as following code.



The code above create new 3 threads win1, win2 and win3. Worker 1 to worker 3 are instance of “KioskWindowsWorker” class that implement Java “Runable” interface, java thread only accept the classes that implement this interface and contain “run()” method. The “start()” will invoke a “run()” method inside “KioskWindowsWorker” class and execute the sequential code inside this method in parallel with other threads.

The “run()” method inside “KioskWindowsWorker” will retrieve a passenger from passenger queue and then retrieve a taxi from taxi queue and then remove them from both queues. In this application, all of the workers are using the same queues to process the tasks. The task of retrieving and removing data from a queue are done by calling “getNextJob()” method in “KioskWindowsList” class which is Singleton class.

With multiple treads accessing the same queue at the same time, each thread may receive the same passenger or the same taxi which should not allow to happen in real situation. In this case Java also provide a barrier that will allow you to control the execution of a method with “synchronized” key word. This will allow only one thread to access the method at a time.

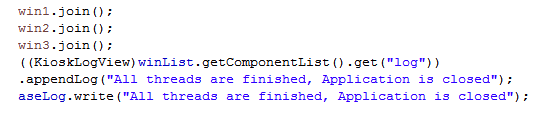


Java also provide very useful method to control threads inside an application such as “sleep()” and “join()”.

C:\Users\Wonchana\Desktop\sleep.png

Code above make the current thread stop working to simulate a processing time that required to process each customer (the time is random number from “getProcessingTime()” ) in the queue.

A “join()” method allow you application to wait until the thread is end. In Kiosk taxi application, all windows (threads) must stop working before the application is closed.



An example code above waiting for all 3 threads to end then record application status into a log file.

Comparison in methodologies

In the first stage, the planning is the most important part. We used most of the time to discuss about the plan such as data structure that we will used in the actual programme, output of the reports and class diagram. The development on the first stage is fairly easy with only a few differences from our plan. Since we have already have a class diagram, we can easily split the tasks between me and group member with almost no problem at all. The development of a program comprise of 2 major iterations and 1 minor iteration. The first iteration is to develop main requirement of the programme such as read file, store it in data structure and write the report (output). The seconded requirement, we discuss more about the custom exception and unit test which are not included in the plan from the beginning and then write the code as we agreed. The last minor iteration, we fixing several minor errors in a program and adjust style of reports (output).

In stage 2, we use agile based programming, the planning and developing in this stage have many iteration. The planning process is more like a stand-up meeting. We all exchange ideas in every iteration, each iteration is very short. We used pair programming to develop a program in this stage and we also communicate between the pairs as a development go.

At the beginning of stage 2, we try to divide the workload equally which is very hard to do because we did not have a clear plan or just a rough class diagram. In the end we design to sprit the work into 3 packages thread, view and model. Stage 2 require a lot of communication in order to create just a single class to meet the requirement. The pair programming help a lot in this part, two of us do the coding and another two control the direction of the code. After view and model are done the pairs switch the place and continue on coding the threads.

Most of the time in stage 2 is about coding. On another hand, stage 1 did not required much communication after we got the plan ready, in fact we barely speak to each other, each person just working on their parts. When it comes to coding, stage 1 is much easier to write because we did not have to think that much just write the code according to the plan. Stage 2 development process is a mess, we have to change a lot of thing in the code many times over the development especially on thread parts.

In contrast, stage 1 programme is in fact a lot smaller than stage 2 but the whole process from planning to complete programme required the same amount of times as stage 2. But it is not that bad, stage 1 did not required all team members to work at the same place all the time. If your project is much bigger also a co-project with many developer groups, the planning is very important.

As you can see Agile programming is suitable to develop a small application or small modules of the larger project the same as stage 2 programme. But if the project are much bigger, it may produce a lot more problems to the project. The planning in stage 1 is in fact unnecessary, since the actual program is very small. It should be faster if you just write the code. But, planning your project a head will certainly help in larger project.

Version control

The version control allowed to track coding history, redo-undo and share source code via internet. BitBucket repository was used as group repository for stage 2. Git client was installed to all members of group in order to communicate with our repository.



Figure 2 Screenshot of our repository

Resources used

The lecture note and code example from tutorial contained most of the technics that were used inside our application. Furthermore, we are a group of 4 people, each of us have some experiences in java programming. Also, since we were working together most of the time, we can just ask each other’s opinion about the code. If we have some problem that we did not know the answer, we can find it using google search but for most parts we can find the answer from java document on The Oracle website.

Contributions

The original pair in the stage 1 are myself and Thanaphong Sontayasara. The coding process is quite the same as in stage 2 which is the two of us sitting together on the meeting time and write the code. The amount of work for each of us were discussed before we actually write the code. As you can see in our source code we have multiple package and classes that were designed using Object oriented programming, so it really easy to distribute the work equally (there are some different but not much).

In stage 2, we combined with Jonas Galdikas and Aaron Czerski Group so we became a group of 4 people. We reuse some of the basic classes from stage 1 as a starting point. Jonas and I mainly work on threads also adding additional functions to the program such as adding observers, adding synchronize point to the models and modify a log over all iteration of development as we agreed in each meeting.

Thanaphong work on the GUI of the program inside the view package. The design of the GUI came from all of us 4 as we discuss about it in our meeting. The components and style were adjusted by him as we agreed in the meeting. Aaron Czerski mainly work on the models, data structure and details of methods inside model package.

It really hard to tell who contributed the most of the programme since we are working together most of the time. Model and view packages seem to have a lot of work from the surface but when we combined the code together both view and model have to be adjust to support the MVC pattern, so basically the amount of work for each person are quite the same.