**Week 3 assignment**

Couse: Operating Systems Design(146043-31001), 2021

Name: Chang wan Woo

Programming Assignment NUMBER: Implementation assignment #2

Due DATE: 2021/04/27

**Program environment**

Window 10, Interllij, java version "1.8.0\_291",

Java(TM) SE Runtime Environment (build 1.8.0\_291-b10)

**Problem Statement** :

PID manager manage assignment and releasement for pid. To manage thread, Main problem for this project is managing multiple thread(including creation, termination, sleep) and realize synchronization. I used minimal priority queue and to implement synchronization, I used “synchronized” keyword.

**Software design:**

To make race condition and deadlock condition which need synchronization tools to work properly, I made threadWorker class and threadMake class. Former class manage thread including creation, termination and sleep, latter class make thread. Latter class is needed to make more than two threads access to getPID() and getPIDWait() so that there could be race condition.

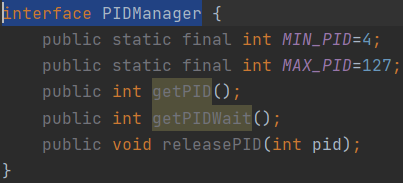
Also, I utilized priority queue which stored available pid which is more useful than array when creating and terminating pid. Specifically speaking, my program test both getPID() method and getPIDWait() method in one execution. Firstly test for getPID is started. In main class, it checks periodically whether it should be terminated by endtime. Then, terminate all threads which is created by getPID(), and execute test for getPIDWait. If it is over, terminate the program.

**How to start?**

Program take three input, number of thread, lifetimeProgram of program and lifetime for thread. If three inputs are invalid, take inputs again and otherwise, execute program. (Input should not be character, or negative number.)

**Implementation**

**Interface Description**



The interface PIDManager defines MIN\_PID, MAX\_PID, getPID(), getPIDWait(), and releasePID(). It means class which implements PIDManager should override those methods.

**class Description:**

1**. class ThreadWorker:**

It implements Runnable so that it can create thread by executing start() method. In run() method, which can be executed by calling start method(), it checks whether time to terminate threads below is the function.



In class, it initiates PID, lifetime, createTime, endTime, and startTime which is defined ‘private’ and those are necessary in managing threads.

**2.class Manager**

Manager class implements PIDManger which predefined getPID, getPIDWait, releasePID. Also, PID queue is managed in Manager class.

**Method: public static void fillQueue**

input: none

output void

There is priority queue in class. If other call fillQueue(), add value from MIN\_PID,4 to MAX\_PID,128. Also it is defined public, because it can be called from other class.

**Method: public synchronized int getPID()**

Input: none

Output: int( return -1, if there is no available pid, otherwise it returns peek in queue.

The getPID returns -1 if there is no available pid, and returns PID if there is available pid. Also, I added synchronized keyword, because when more than 2 threads can access to getPID(), synchronized makes them synchronized.

**Method: public synchronized int getPIDWait()**

Input: none

Output: int(It returns peek in queue).

Unlike getPID(), if there is no available pid, it wait in method by executing wait() until other thread call notify(). Also synchronized keyword in method to synchronize multiple accessed threads. It returns top of the value in the queue.

**Method: public synchronized void releasePID(int pid)**

Input: int (pid which will be released.)

Output: void

The releasePID adds pid to priority queue, PIDpq. Likewise getPID and getPIDWait I used synchronized to synchorize if more than two threads access to relasePID(). When releasePID is called, add pid in the queue and execute notify() to wakeup thread in wait().

**3. class threadMake**

The class threadMake is necessary for making concurrent access to getPID, and getPIDWait(). In class, threads are waited random time from zero to lifetime of thread. If flag==1, execute getPID(), otherwise execute getPIDWait(). In constructor, create the thread by calling start() method. Class threadMake is responsible for making new Thread. I design my architecture to execute getPID() and getPIDWait() concurrently. To make this situation, I made one more class, threadMake which implements Runnable. In this class, thread sleep 0~lifetimeProgram seconds randomly and once thread wakeup from thread.sleep(), execute getPID or getPIDWait to create new threads.

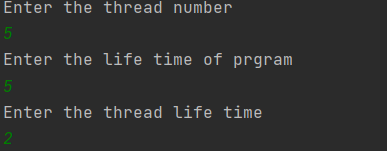
**4. class ThreadExample**

ThreadExample is main class for my class. It takes inputs from users and sets startTime which is needed for thread creation and termination, fill pid in queue and create threadmake thread. Also, if test for getPID() is finished, release all pid and set startTime and endTime again.

**Test Description and Results**

(Number is thread number, lifetimeProgram is lifetime for program and lifetime is lifetime for thread.)

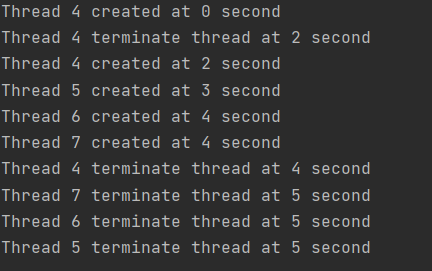
**Input:**

Number: 5

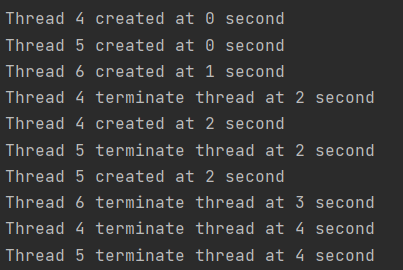
lifetimeProgram:5

lifetime:2

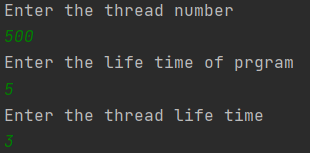
**Result for getPID()**



**Result for getPIDWait()**



Both two method are work properly according to my intention. Program makes 5 threads and thread live for 2 seconds. Also, if smaller number of pid is released, data structure for pid use smaller pid first. Last thing to mention is that if lifetime for system is over, thread 7,6,5 is also terminated as you can see in the top image.

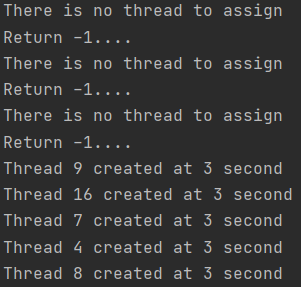
**Input**

**Number: 500**

**lifetimeProgram: 5**

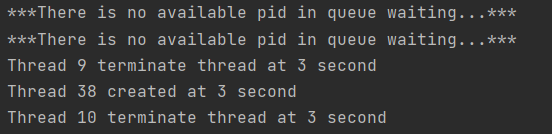
**lifetime: 3**

**Result for getPID()**



We can see that if there is no available pid in the queue, program printout error message and returns -1.

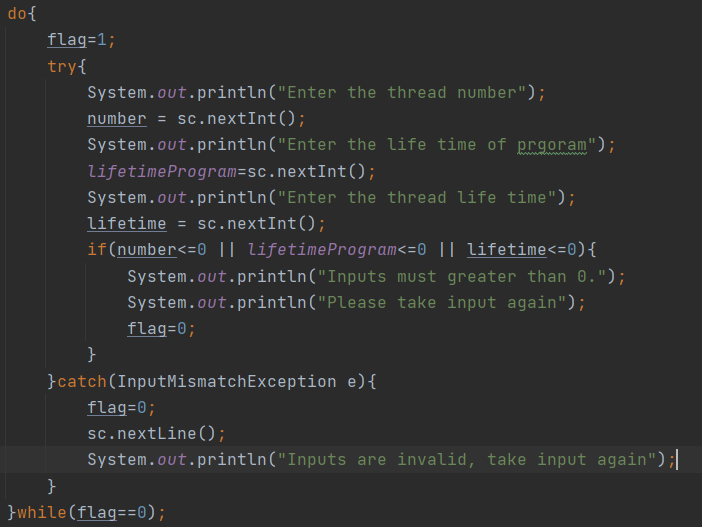
**Result for getPIDWait()**



In getPIDWait() test, thread waits in the method by calling wait(). It is waked up other thread release pid and call notify(). It is difference from getPID().

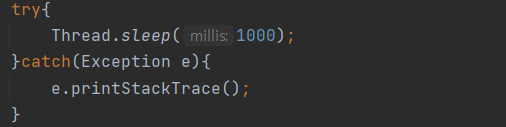
**Error Handling:**

1.



One possible scenario is input exception. If users, take inputs as string or negative numbers, program print out error message and take input again. As you can see in the above image, if invalid inputs are entered, change flag as 0 and take input again.

2.



In program, it is needed to execute thread.sleep() to control thread manually. When call sleep() method it should be used with try() catch() because it is possible thread exception.

**Error result.**

