**Implementation assignment #4**

**Subject = Operating Systems Design**

**Major = ITM**

**Student ID = 17102062**

**Name = Lee Yong Hoon(이용훈)**

Table

**1. Step 1~8**

**2. Step 9 - Execution**

**3. Step 9 - Code analysis**

**1. input part**

**<AddressTranslator.java>**

**2. FIFO part**

**<AddressTranslator.java>**

**<PhysicalMemory.java>**

**3. LRU part**

**<PhysicalMemory.java>**

**<PhysicalMemory.java - findVictimPN method>**

**<PhysicalMemory.java - LRUaddFrame method>**

**<AddressTranslator.java>**

**4. Compare result of FIFO and LRU**

**1. Step 1~8**

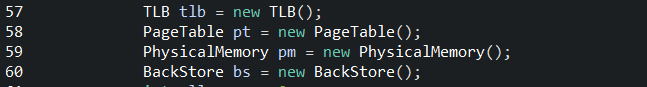
**Analyzing the source code**

**<AddressTranslator.java>**

1. using scanner bring inputFile.txt

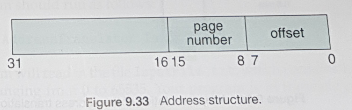


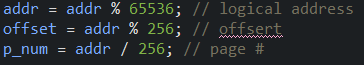
2. make TLB, PageTable, PhysicalMemory and BackStore by class declaration.



3. read inputFile.txt the number of inputFile.txt is virtual address.



You should divide logical address as 65536 because, in this case, size of physical memory and logical memory are same. Size of physical memory is 65,536 bytes. Therefore set logical address var addr as addr % 65536. In page 453, size of offset is 8 bytes, therefore, offset var is set as addr $ 256. Size of page number is 8 bytes, therefore, p\_num is set as addr/256.



4. now we try to change logical address to physical address. Firstly, you should check TLB that as page number is in TLB. If page number is in TLB, bring relevant frame number in TLB. 

If there are no valid frame number, it means tlb\_miss. Therefore you should go page table.

The reason of why you search TLB than page table is than TLB is more efficient and fast than page table.

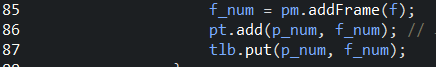
Check page table whether relevant frame number is there. 

If there is no valid frame number, it means page fault. you check BackStore and bring frame to Physical memory.

You should check BackStore. And bring frame. 

And then add frame to physical memory. 

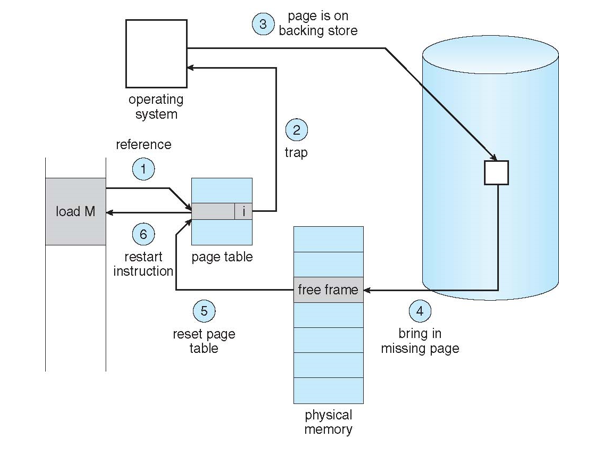
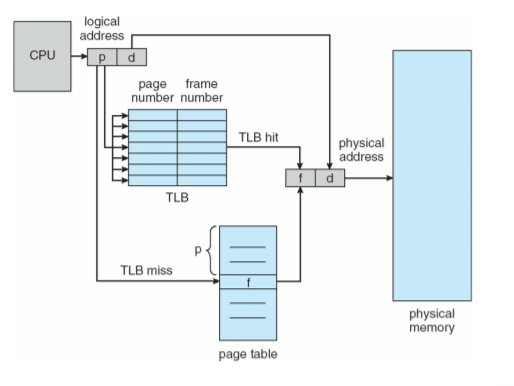
And then add relevant page number and frame number to page table and TLB



And print virtual address, physical address and value



This 4 step is same with below figure



A) check TLB - TLB hit then bring frame number to physical memory

B) if TLB miss happens, check page table and bring frame number to physical memory

C) if page fault happens, check Backing Store (trap is skip)

D) bring missing page to physical memory

E) after D) reset page table and TLB

**Overall process.**

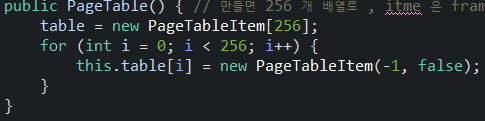
1. there are nothing in page number and TLB. Therefore firstly bring frame number and data value corresponding page number from BackStore.

2. after that if the page number is in TLB or page table use it.

3. If there is not exist go to BackStore and repeat 2,3 steps.

**<pageTable.java>**

1. using constructor, initial value is set 1. In **AddressTranslator.java** , check page table and if value is -1, page fault happens and check Backing Store and bring frame number to page table.



Method **get** is return frame number.(if not valid return -1)

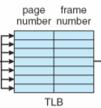
Method **add(int p\_num, int f\_num)** is add **f\_num** frame number to page table that page number is **p\_num**.

2. table = new PageTableItem[256];

you can see that there is PageTableItem class. The variables are frame number and Boolean that whether it is valid or not.

**<TLB.java>**



 1. There are 16 entries in the TLB, so 16 cases can be executed fast than using page table. and if the entries are ful, we delete the oldest data in TLB and insert new data.

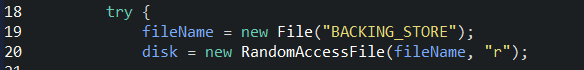
2. TLB.java use Hashtable to match page number and frame number

3. if you see the put method, there are queue and through queue, delete the oldest data and insert the new data.

4. the get method is that if there is key(page number), you can get relevant value(frame number)

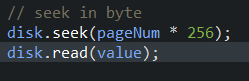
**<BackStore.java>**

1. There is BACKING\_STORE.txt. and read txt file randomly

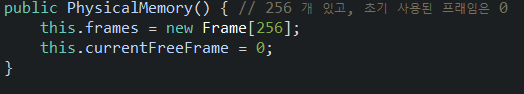
Through seek() function, you can use file pointer. And can get 256 values. And copy to result array and return result.

You can get frame’s date(256) by return results in getData method

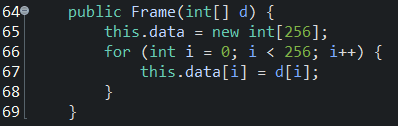
**<PhysicalMemory.java>**

1. there are 256 frames in Physical memory. Using constructor, set frames.



2. There is Frame class.





There is int[] data array to store data of the frame. Through parameter, frame’s data are set.

3. there is 2 methods, addFrame(Frame f) and getValue(int f\_num, int offset).

addFrame(Frame f) means, new frame is made with f.data and this is added to frames array.

getValue(int f\_num,int offset) means, return data value of frame which is same with frame number(f\_num).

2. Step 9 - Execution screenshots

* 1. Implement ‘Modifications’ part in page 456.
     + - Change the physical memory accommodates only **128** frames
       - Need **page replacement** to handle all memory references
         * Implement **two page replacement algorithms** (two among LRU, LRU approximation, and FIFO)
         * Complete the page replacement part with the above algorithms
         * Plug the page replacement part on the initial implementation given to you

The statistics add the number of page replacements

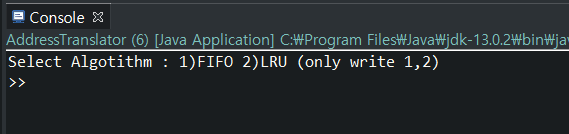
* + - * Analyze the result with the original implementation given to you
      * Also describe the difference of the results with two replacement algorithms are applied each.
      * The report will focus on the difference between the original and your modification

**2. Step 9 - Execution**

**A. FIFO**

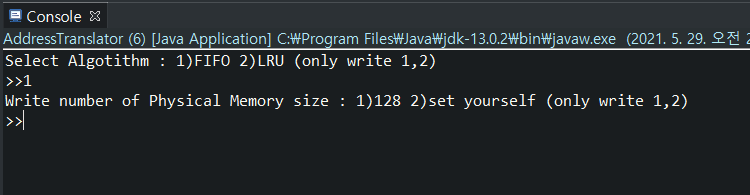
**You can choose algorithm between 1)FIFO and 2)LRU**

**You can only write 1 and 2 and if you don’t write 1 or 2, request other value repeat.**

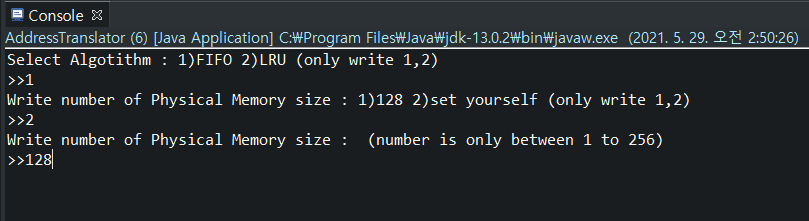


**If you choose 1)FIFO, you can choose number of physical memory size. 1) 128 2) set yourself**

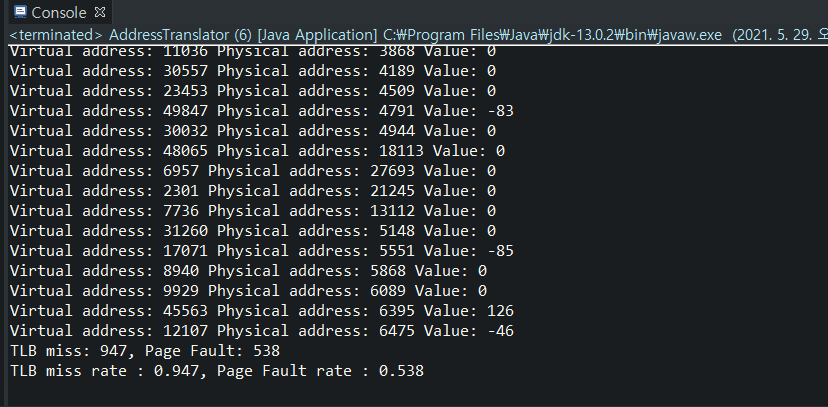
**You can only write 1 and 2 and if you don’t write 1 or 2, request other value repeat.**



**If you choose 2) set yourself, You can only write number between 1 to 256 you don’t write number between 1 to 256, request other value repeat.**



**When execution finish, you can see TLB miss rate and Page Fault rate**



**B. LRU**

**LRU is same way with FIFO. What you need is only choosing 2)LRU**

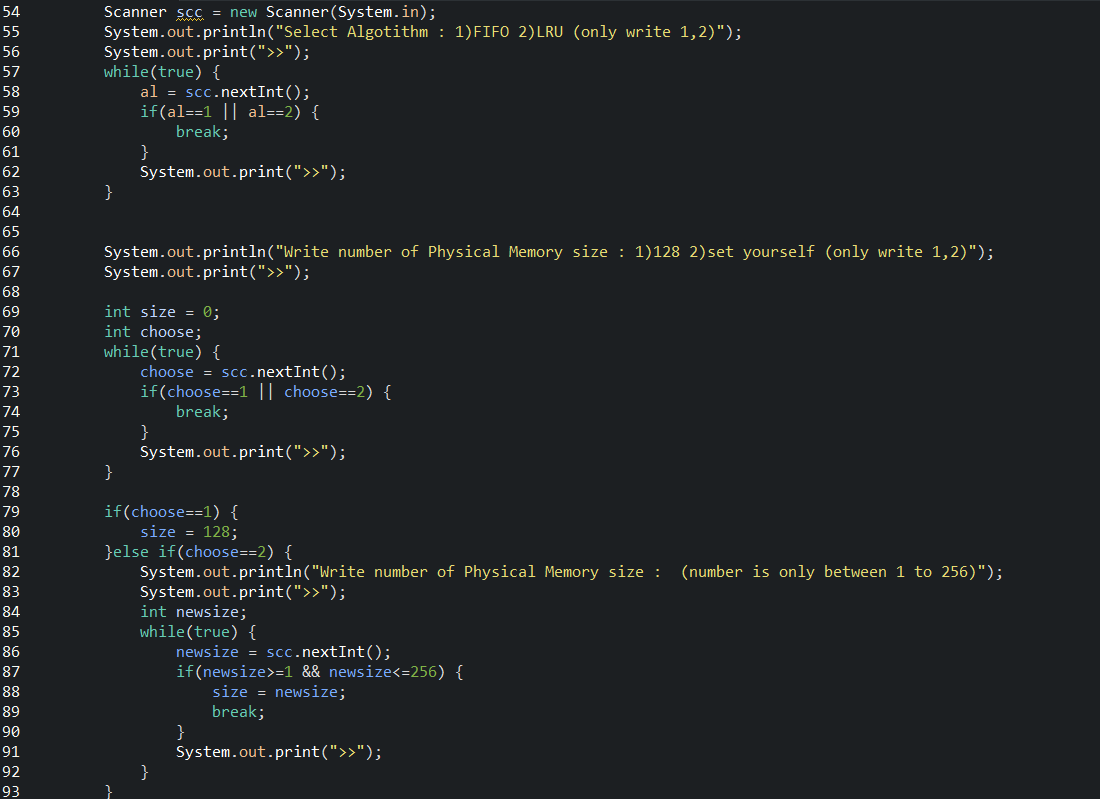
**3. Step 9 - Code analysis**

**1. input part**

**<AddressTranslator.java>**



**-al variable. 1 is FIFO, 2 is LRU**



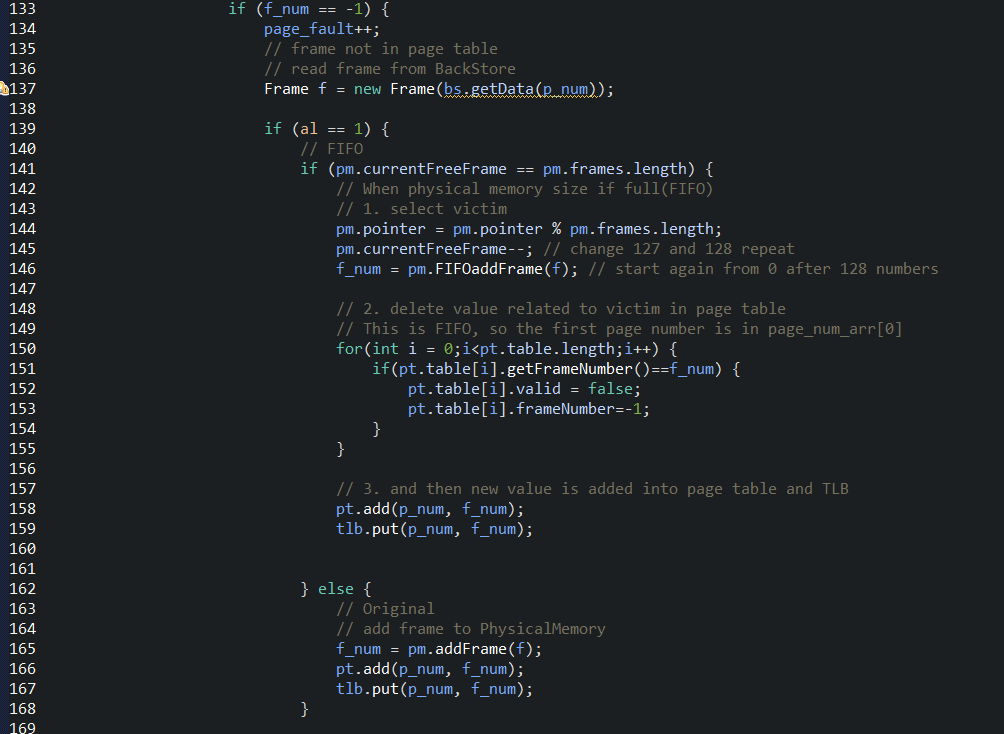
**You can get size of physical memory size to size variable**

**And size variable is being parameter of PhysicalMemory class.**



**2. FIFO part**

**<AddressTranslator.java>**



**Until physical memory is full, the way is same with original code.**

**(137 line) get new Frame from backstore and (165 166 167 line) add frame to PhysicalMemory. And doe pt.add and tlb.put**

**After physical memory is full, the way is FIFO(First In First Out)**

**1. select victim**

**We use pointer to use FIFO. Pointer is indicator of frame number in physical memory. After 127(last index), the pointer is go to back 0 and increase 1 by 1(0->1->2 … 125->126->127->0->1->2 …)**

**And get frame number by using new method named FIFOaddFrame.**

**2. delete value related to victim in page table(150 line ~ 155 line)**

**Through frame number and page number , we can choose value in page table and set it to invalid.**

**3. new value is added into page table and tlb.**

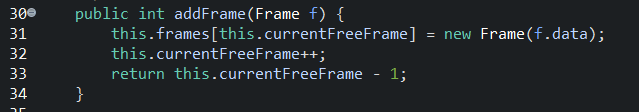
**<PhysicalMemory.java>**

**1. this is original method in PhysicalMemory.java to get frame number.**

**curruntFreeFrame is a variable to store how many frames are used.**

**if size of physical memory is 256, then frame number is increasing to 256 firstly(0->1->….->255)**

**After physical memory is full, currentFreeFrame is over size of Frame array size. Error happens.**

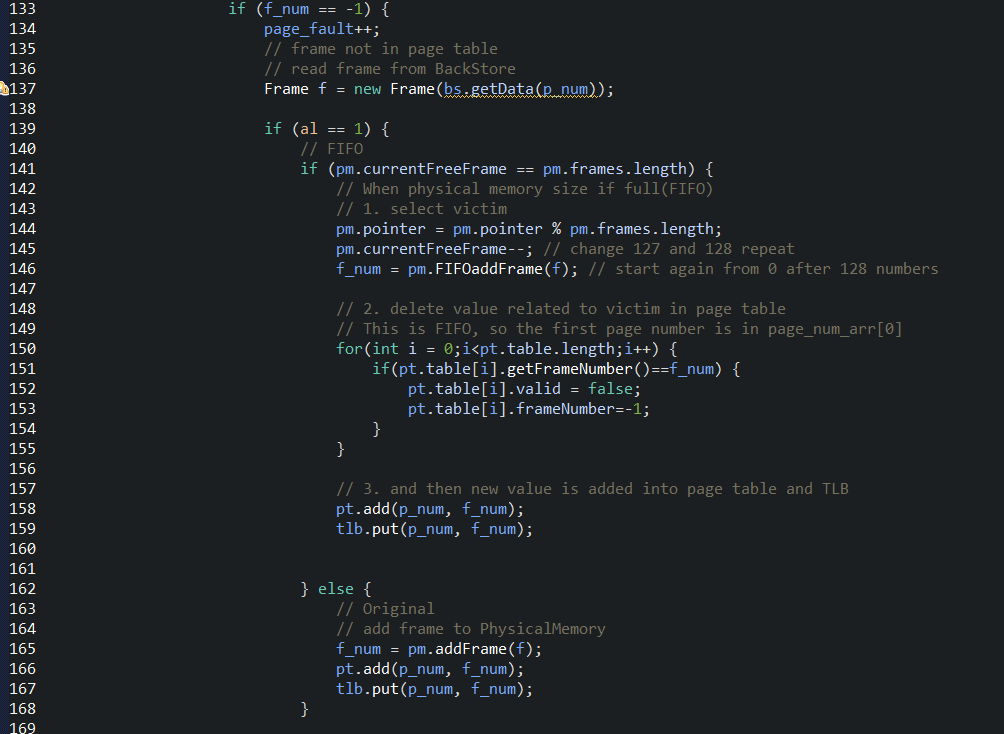


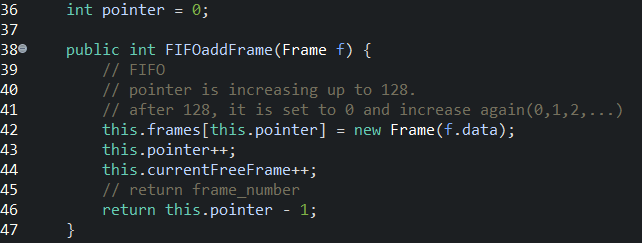
**2. this is new method in PhysicalMemory.java to get frame number named FIFOaddFrame**

**curruntFreeFrame is a variable to store how many frames are used.**

**But it is not used. The pointer is used to indicate frame number.**

**The way is same with the original way. But difference is that after physical memory is full, pointer is divided by size of physical memory(in AddressTranslator.java) and remainder is being pointer, so error not happens**





**3. LRU part**

**<PhysicalMemory.java>**

**< PhysicalMemory.java - findVictimPN method>**

**I will explain LRU part starting from PhysicalMemory.java.**

**1. To implement LRU algorithm I use LinkedList data structure in java. It is used as Queue.**

**Add : When page number firstly, it add to LinkedList in order.**

**Update : If page number is reused, then it go back to last of LinkedList. The age of page number is reset. (remove it and again add by using addLast)**

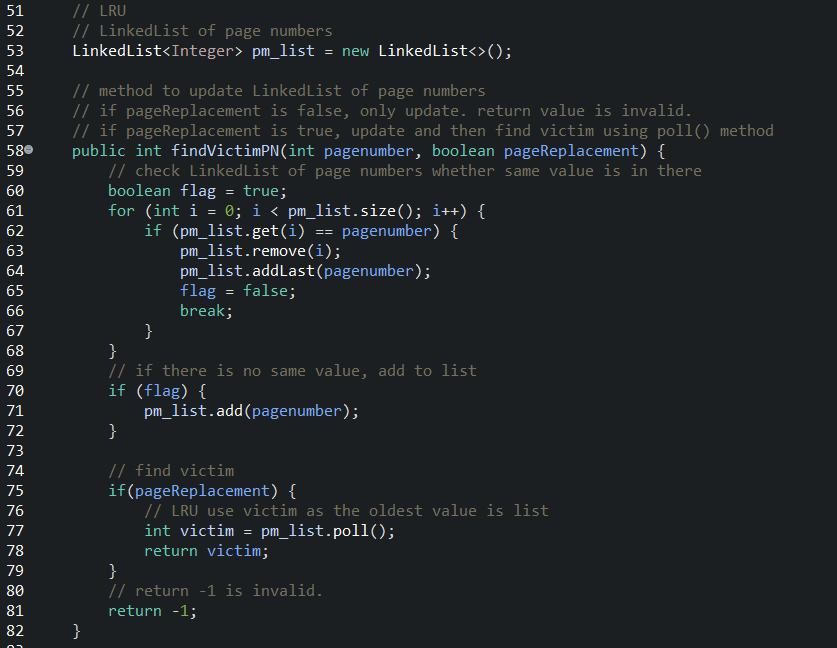
**Poll : And you should select victim, then you can get the oldest page number by using poll() method.**

**2. Boolean pageReplacement**

**When pageReplacement is false, only add and update LikedList. Return value has nothing meaning**

**When pageReplacement is false, add and update LikedList. And then, find victim from LinkedList Return value is victim page number**

**3. using for loop if same page number is in LinkedList, then update. If not, add to LinkedList. To classify two stage, I use flag variable.**



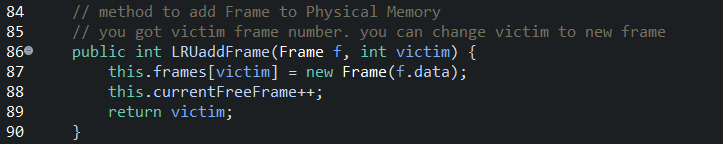
**< PhysicalMemory.java - LRUaddFrame method>**

**This is method to add Frame to physical Memory.**

**Using findVictimPN method, you can get victim frame number.**

**And victim frame number if parameter of LRUaddFrame method.**

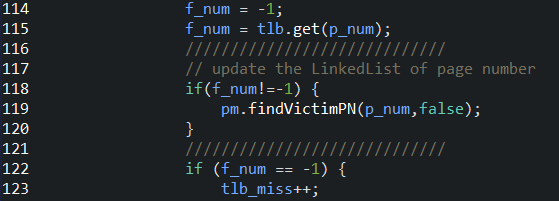
**The victim frame is change to new frame and return frame number which is reused.**



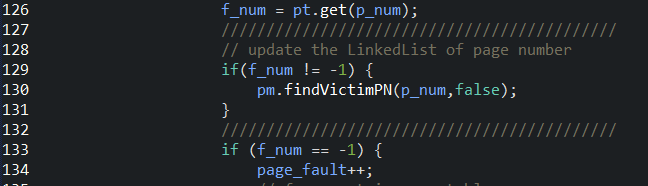
**<AddressTranslator.java>**

**When page fault is not happens, we should update page number LinkedList.**

**When TLB hit happens, update or add value in the LinkedList of page number**



**When page table hit happens, update or add value in the LinkedList of page number**



**Until physical memory is full, the way is same with original code.**

**(137 line) get new Frame from backstore and (201 202 203 line) add frame to PhysicalMemory. And doe pt.add and tlb.put. The difference with FIFO is that you should update or add page number LinkedList by using pm.findVictimPN(p\_num,false) - victim is not needed, so parameter is false.**



**After physical memory is full, the way is LRU(Least Recently Used)**

**1. select victim**

**Firstly, you should update and add page number in page number LinkedList.**

**And then find victim page number from page number LinkedList.**



**findVictimPN is used and it need victim page number, so parameter is true.**

**1-A. delete value related to victim in page table**

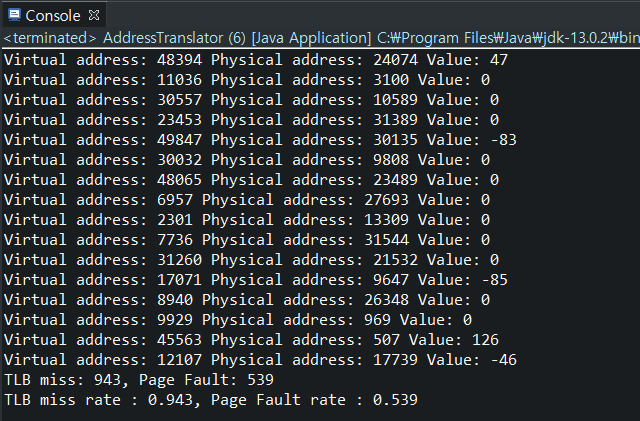
**Through page number, we can choose value in page table and set it to invalid.(line 179 - line 180)**

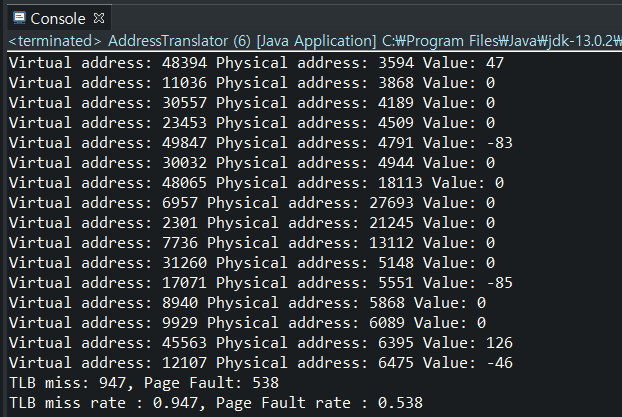
**2. get frame number by using pm.LRUaddFrame method.**

**3. new value is added into page table and TLB**



**4. Compare result of FIFO and LRU**

**The result of FIFO(Physical memory size is 128) The result of LRU(Physical memory size is 128)**



**The normally LRU performance is better than FIFO. But in this case, the result is different with our expectation.**

**The expectation of LRU is that the oldest page number will be used in closer future, so we give some more priority to that page number. But if program doesn’t use the oldest page number then, increasing of performance can not be happens. Also The page table and TLB entry is different between FIFO case and LRU case. The result can be different with our expectation with various environment.**