Operating System Design

- **Designing a Virtual Memory Manager –**

**19102091 YoungHwan Phan**

**AddressTranslator.java file**

In this file’s main method, initially declared some variables. inputFile String is declared for calling under the file object’s parameter. And addr, p\_num, offset, f\_num, value, phy\_addr, tlb\_miss, page\_fault integer type variables are declared for expressing page replacement implementation. For the setting, this program create 4 objects such as tlb, pt, pm and bs to use implementation.

Because in this program, virtual memory space is represented as 16bit, in the while, addr is modulated as 2^16. Page schema’s pre-8bit represent page num, so address is divided as 2^8. And post-8bit represent page offset, so address is modulated ad 2^8. Initially, frame num is initiated as -1.(Not allocated) program tried to get frame number that is matched to the parameter(in here this parameter is page number.)in TLB object. If fail to get appropriate frame number in tlb, increase tlb\_miss 1. Next, we try to get framenumber associated to the pagenumber in the pageTable object. If fail to get again, increase page\_fault 1. Finally program try to get frame from backing store. And add frame to the physical memory object and get frame number associated to the page number. And add the page number and frame number into page table and tlb.

For end this filtering job, physical address is determined as framenum \* 2^8 + offset.

End of the while, we can check how many tlb miss and page fault are occurred.

**BackStore.java file**

There is a getData method who require the pageNum parameter. With this parameter, program open the backingstore file as filename readonly condition and try to seek(in here, moving pointer in the disk) page number\* 2^8(because this is address when we set in the addressTranslator.) Disk.read(value) means read disk data as maximum size of value.length and store date in the value array. Under the for, store the value array’s value to the result array. This array is a data what we want.

**PageTable.java file**

In this file, there are inner class as PageTableItem(I will call this as pti.). pti has framenumber and valid bit(as Boolean here.) Pti constructor require framenumber and valid bit state. And pti has getFrameNumber method. Go back to the Page Table, pti type array is declared as global variable. In the PageTable(I will call this as pt) constructor, pt is initialized 256 ptis. And allocate to table’s each items as frame number is -1, and valid bit is false(Because, constructor is just set initial mode). And there are get and add method. Get method require the wanted page number as parameter. If matched frame number is existed, return it. Else, return -1. Add method require the page number and frame number as parameter. Table’s page number’s item is set as frame number and valid bit as true.

**PhysicalMemory.java file**

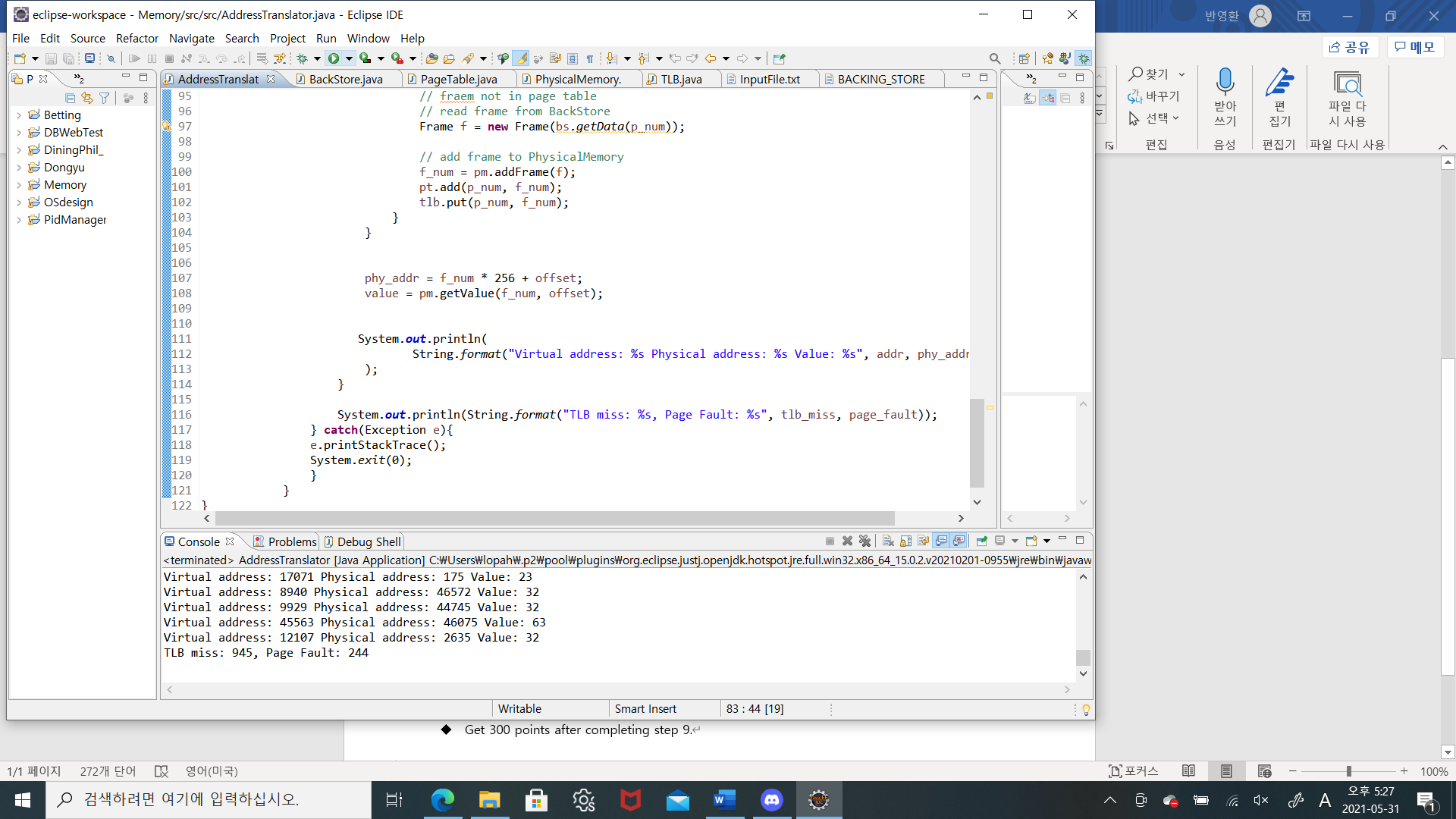
In this file, there are inner class as Frame. This inner class has int type array as date naming. There are two constructors exist. First one is initialization data array’s size as 256 and set data as d array that is passed by parameter. Second one is copy the frame if frame is passed by parameter. Go back to the physicalmemory(I will call this as pm) file, there are frame type array and int type current free frame. In pm’s constructor, frame size is set as 256 and current free frame is set to 0. There are two methods in pm. addFrame set this frame’s currentfreeframe order’s value as new passed frame as parameter. And increase current free frame 1 then return original one. getValue get frame as frame number that is passed as parameter. And return frame’s data as offset that is also passed as parameter.

**TLB.java file**

In this file, there are two global members Hashtable and LinkedList type. Tlb’s constructor initiate Hashtable and put key and values in the table and add key in the list. Someone call get method with page number, TLB check this table has key as pagenum. If exist, return this table’s values that is associated to the key. Else, return -1. In the put method, someone pass the pagenumber and framenumber as parameter. For flash, poll data in the list, if it is not null, remove it from table. And then add page number to the list and put page num and frame num into the table.

--------------------------------------**Step 1~8**------------------------------------------

**Original result:**

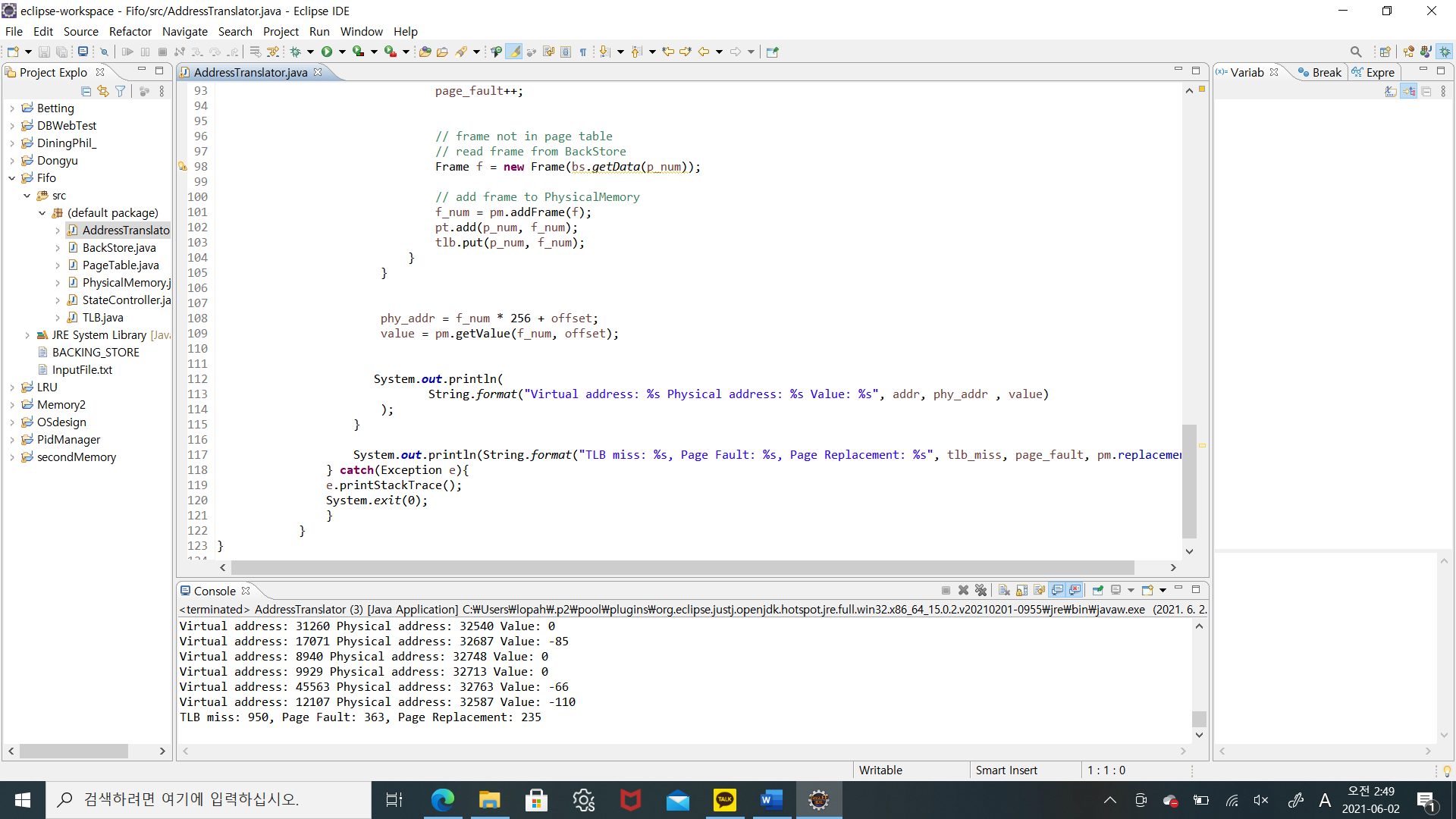




Because the frame size is 256, 244 times page fault is occurred.

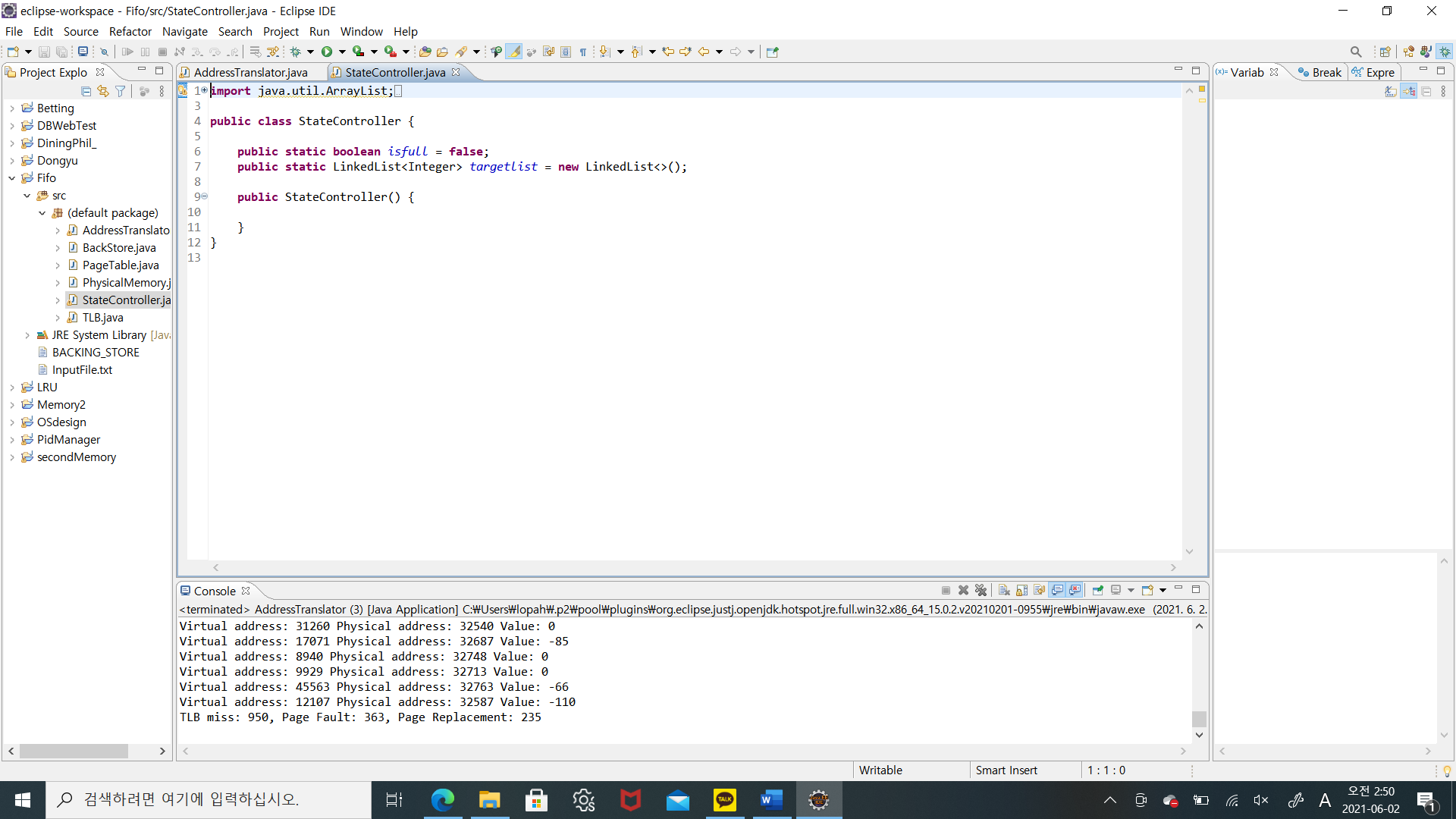
Continuously update the frame.

**FIFO Case:**

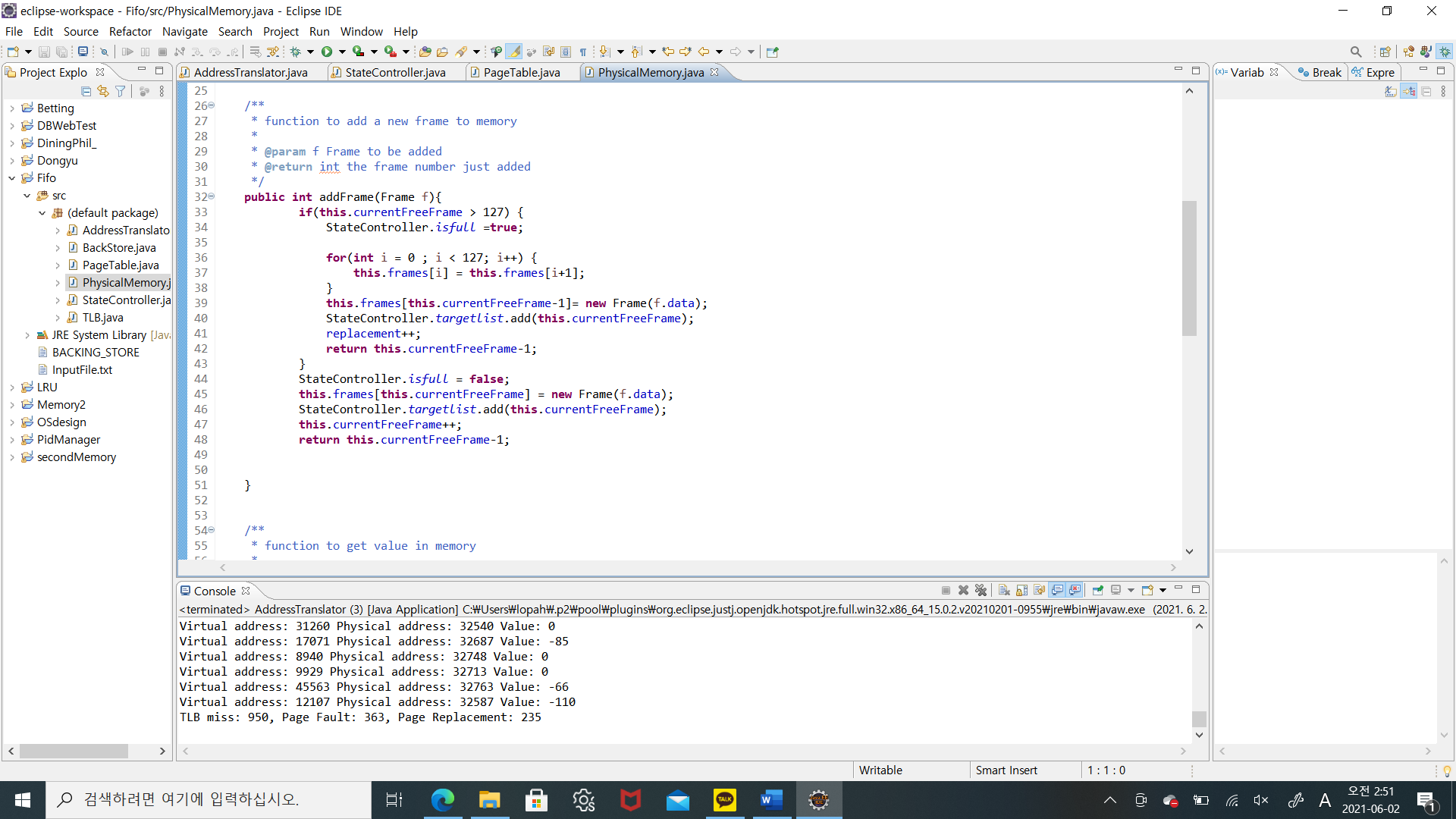




For check the result, page fault is 363 and page replacement is 235. Let’s see my algorithm.



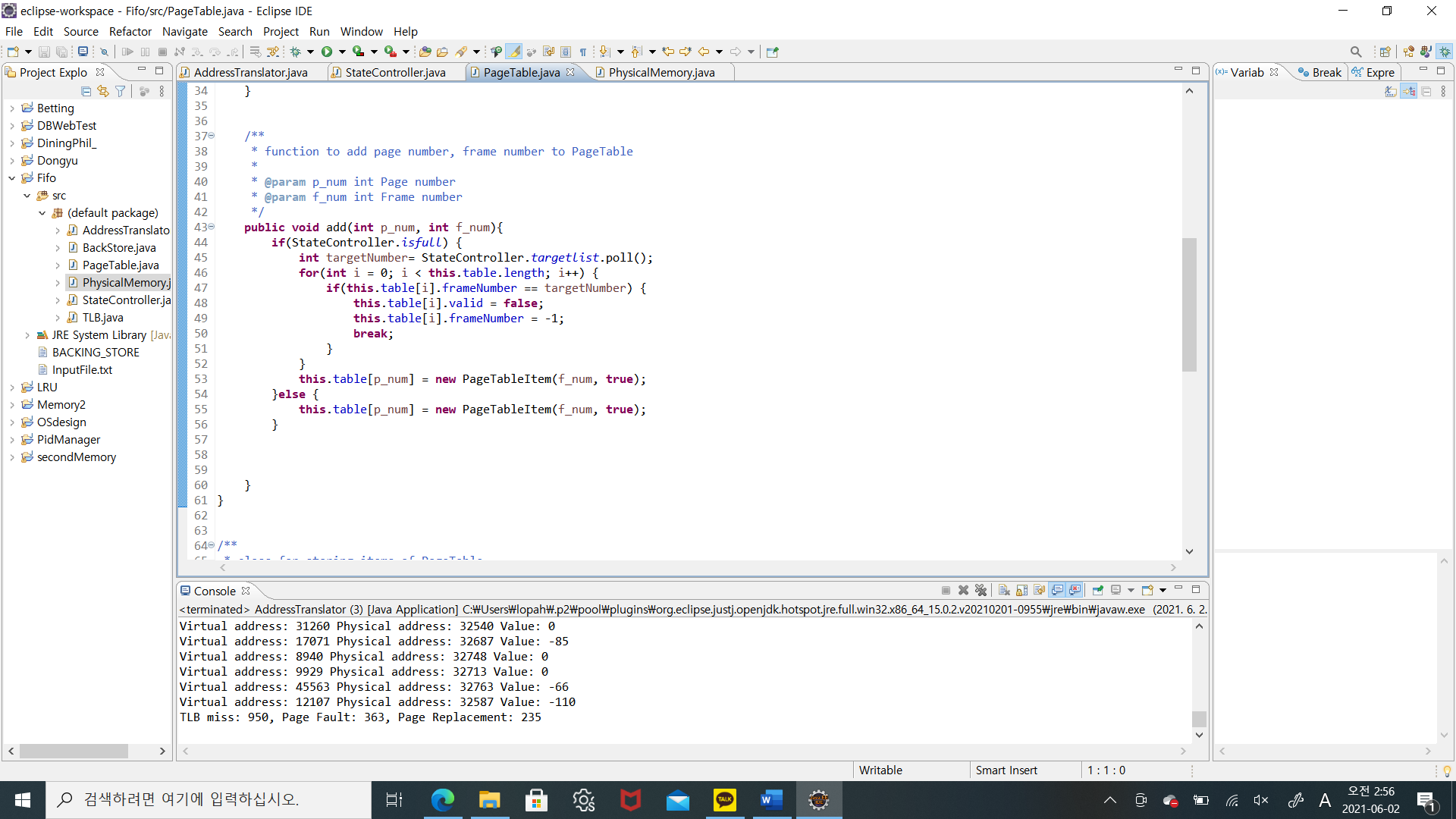
Before using static approach to the list, I make StateController class.



In physical memory’s addFrame method, if this frame is full, set state(isfull) true. And remove frames[0](first in one). Then input the frame the last part (frames[127]). For check victim, add the frame number to the target list in StateController. In this case, page replacement is increased and return its index.

If not full, just add frame in the current free frame index and in the target list.

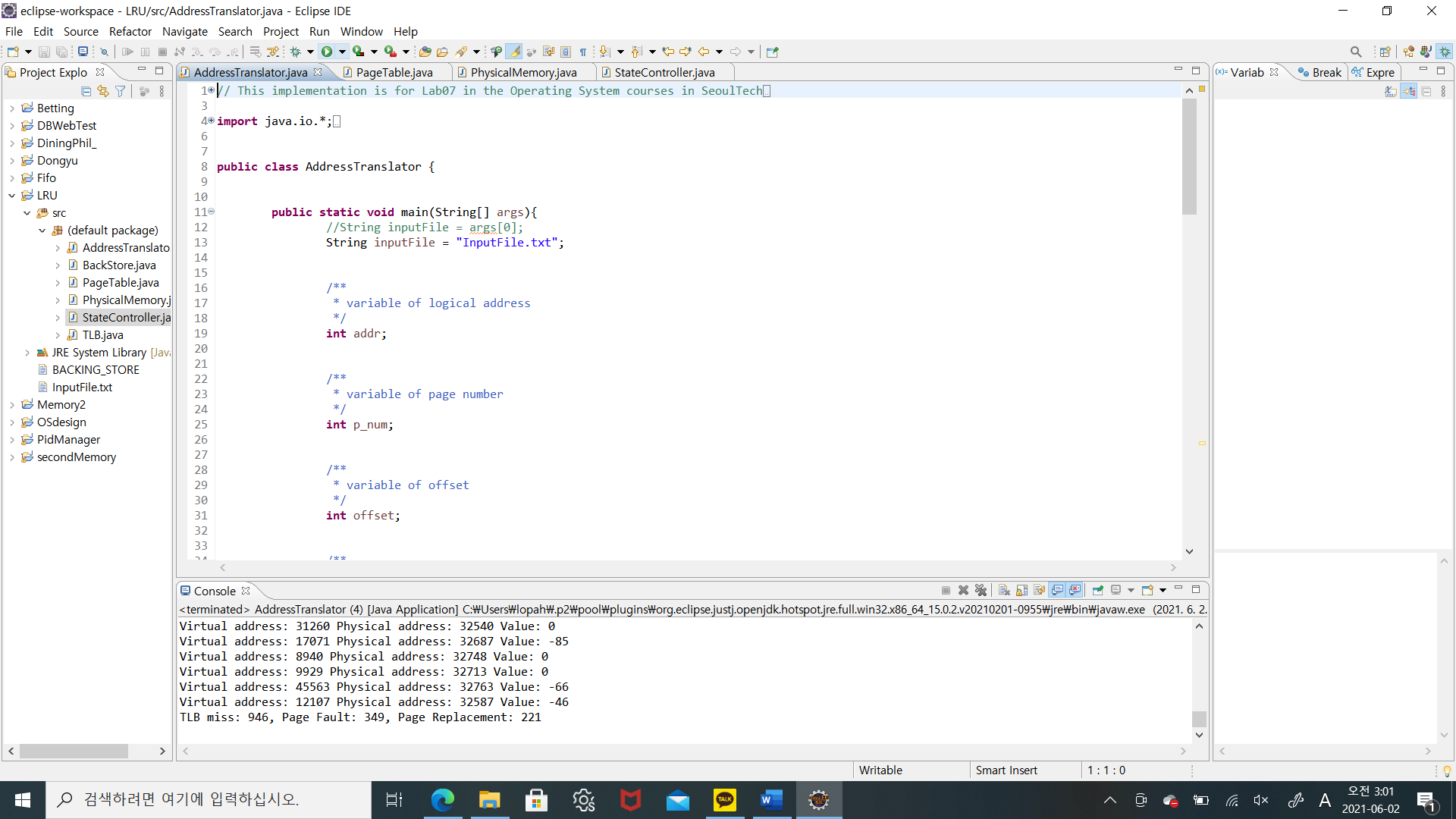
So, we can control the victim as stack.



And we have to update page table also. Poll the victim frame number in the target list and find it in the table. If it is in the table, set valid false and frame number as -1. And then break.

And finally add the new page table item at page\_num location.

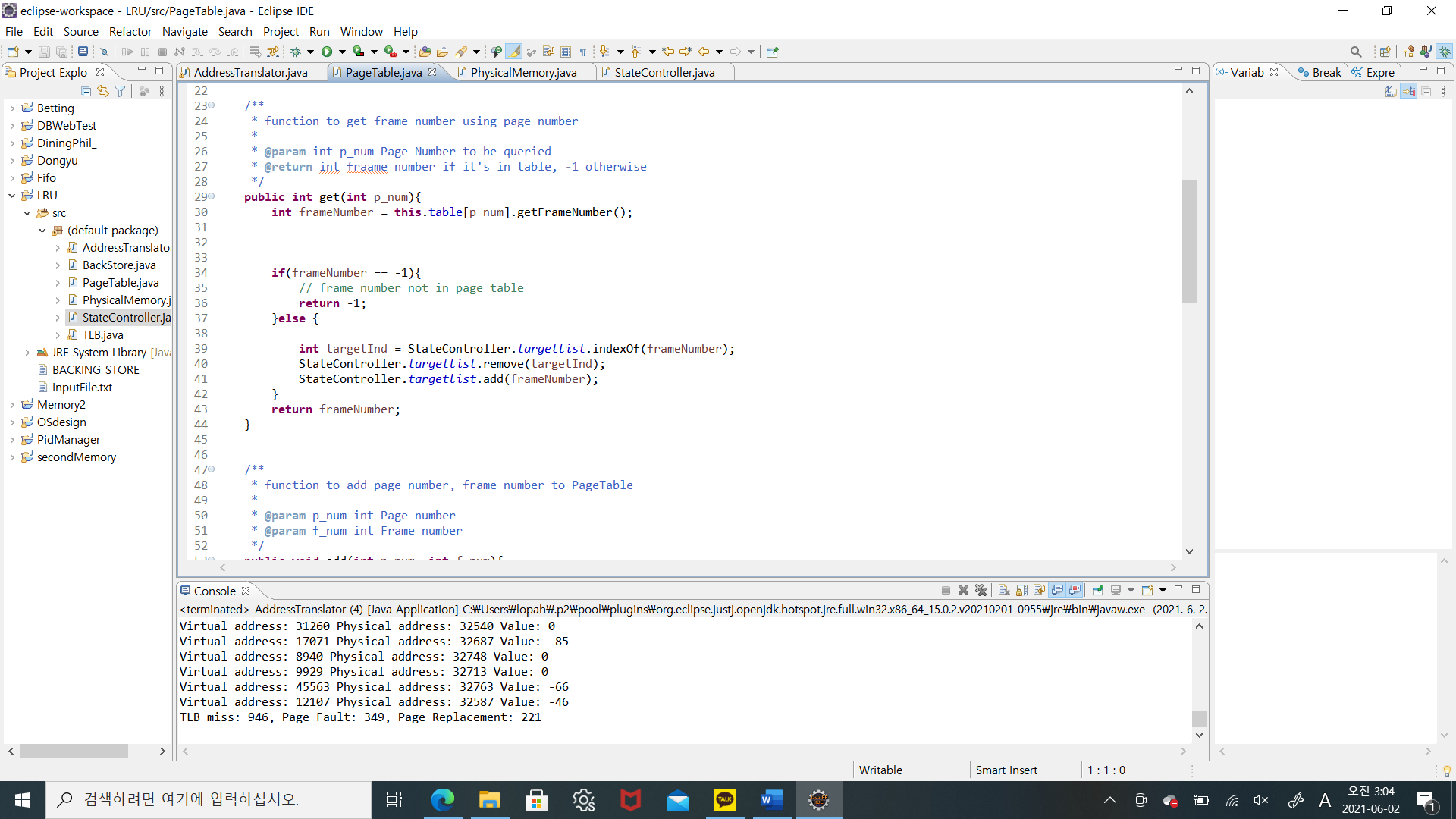
**LRU Case:**





For check the result, page fault is 349 and page replacement is 221. Let’s see my algorithm.

Basic logic is same. Select victim is same because we find stack’s bottom(list’s head). So add algorithm is same. But we have to focus on get method. If we reference the page, this page is floating to the top of stack(tail of list). As below picture.



First, we find the index of frame number and remove it. And add again to the list. As follow these steps, we can update the stack(list).

**Compare with FIFO and LRU:**

LRU is more efficient than FIFO because LRU’s page fault number is lower than FIFO.

**Compare with original one:**

Original’s frame number is 256, and page fault is occurred 244 times.

For ideally, we down size to 128, page fault will be increased to the 488 times.

LRU case’s page fault = 349, FIFO case’s page fault = 363. So my modify version is more efficiency.