Job Class:

{

Protected:

String personname;

String \_barcode;

Public:

Job(){};

Virtual bool execute(Hypermarket\* a)=0;

Virtual string jobname()=0;

String error()

{

String S= jobname();

String space =” “;

Return \_barcode+S+space+personname

};

BP Class:

{

Private:

StackBase<QueueBase<job\*>\*>\* mystack; QueueBase<job\*>\*\* USER;

Public:

BP();

Void readandexecutejob(Hypermarket\*);

};

We have established two classes as can be seen above. The BP(batch processing) class will have methods to manage the stack and queue list. It will read from the txt files first, and store them to form a list. The list consists of a stack of queue pointers and each pointer points to a queue that contains job pointers.

STACK OF QUEUE POINTERS

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| QUEOF JOBS POINTERS |  |  |  |  |  |  |

After the list is formed, then the system will execute each of the job in the queue, according to the sequence of the stack.

We have a job class(which is an abstract class) and 5 sub classes(add, delete, dispose, restock, sale) which all of them have methods overriding the methods in the abstract parent class. In our case, *execute* and *jobname* are the methods that are virtual and to be overridden. We design in this way because it is more elegant and neat, and it is easier for us to code too! ☺

If there is an error during executing the jobs, we have a method in the job class that returns the string of message so that it can be returned to BP class and BP class writes them to the log.txt file.