Our Batch Operating System Project Documentation:

First implementing the five processes:

- **"Process" class:** an abstract class that holds the common characteristics that the processes have as

    1) Process state

    2) Process ID

    3) SemWait/SemPost methods

    4)The abstract method run()

    The classes "Process1", "Process2", "Process3", "Process4", "Process5" inherits from it.

- **"Process1" class:** the class that simulates the process that takes a file name and prints the data in this file.
- **"Process2" class:** the class that simulates the process that takes a file name and data and writes this data in the given file.
- **"Process3" class:** the class that simulates the process that counts from 0 to 300.
- **"Process4" class:** the class that simulates the process that counts from 500 to 1000.
- **"Process5" class:** the class that simulates the process that takes a lower & upper bound as inputs and prints the count from the lower to the upper in a file.
- **"ProcessState" Enum:** the enum that simulates the different processes state : NEW,READY,RUNNING,BLOCKED,TERMINATED.

Second Implementing the kernel function and system calls:

- **"kernel" class:** The class that simulates the OS kernel as it contains methods that acts as the kernel system calls like: takeInt(), takeString(), println(String x), readFile(String filepath),writeData(String filepath,String data)

it also contains as attributes the required Semaphores for protecting the resources like:

-printSemaphore(to protect the screen)
-readSemaphore(to protect reading from a file)
-writeSemaphore(to protect writing in a file)
-takeInputSemaphore(to protect getting input from the user)

Third implementing the semaphores:

- **"Semaphore" class:** the class that provides objects of type semaphores that contains attributes like: isAvailable, ownerID , Queue<>WaitingProcesses

Our Scheduling algorithm and our logic for implementing it:

We choose FCFS scheduling algorithm and we have **"BatchSystem" class** which acts as the **engine (the OS)** it contains a ready Queue for processes and inside the constructor of **"Process"** we insert the newly created process inside it **so the first process to be created is the first one to be out from the ready queue(I.e. the first one to be executed) (FCFS)**

The simulate() method inside "BatchSystem" is our dispatcher that choose the first process in the ready queue(First come) and execute it and waits(enters a loop) until the process finishes and then if the queue is still not empty it dispatches the first one and do the same.

For blocking a thread at a specific point and resuming it to continue execution at the same point we do the following: in the method run for all processes we do semWait before trying to get any resource and if it fails to get it, its state changes to BLOCKED so we make the process wait(enter a while loop) until the resource is available so it moves to ready queue and until the dispatcher selects it and make its state RUNNING again (so the

while loop after each semWait that makes the process wait is while(state != RUNNING))
The testing for that can be done by changing the **semPost** method (as there is no process blocked in FCFS alg.) so that it changes the state of the first process in the waiting queue of the semaphore to RUNNING instead of BLOCKED.
Also after the process finishes using the resource it does semPost.