# ICSI412 Operating Systems – Project 2 – Advance the Scheduler

## Overview

We will be adding a way to call operating systems functions from userland. If you aren’t familiar with Singleton or Façade patterns, you should review them (Wikipedia is a good source for this). Patterns are just names for commonly used code ideas. We will be building an interface to the operating system and a Singleton Façade that calls our BasicScheduler. We will be adding a function to call from userland: sleep.

## Supplied Classes

**OSInterface** has changed to add new functionality.

**Task 1** – Build the OS class

Implement the OSInterface. This class should be a singleton and should implement OSInterface. It should create an instance of the BasicScheduler and call that instance’s methods (like CreateProcess and DeleteProcess). Make sure that you add sleep().

**Task 2** – Prepare the BasicScheduler for sleeping

Instead of a single work queue, you will now need two – a “sleeping” queue and an active queue.

When you are running, you will need to take a process off of the active queue and run() it. Then decide which queue to put it on. **Note** that “queue” here doesn’t necessarily mean to use the queue class in Java.

**Task 3** – Add the sleep() functionality

Each process will have to track how much (longer) it needs to sleep. Put that in an appropriate location and update it on sleep() being called.

**Task 4** – Update the list of sleeping process

After a process runs, the RunResult tells us how much time has passed. We need to update each of the sleeping processes to wait that much less time. If a process is done waiting, it needs to switch queues. Note that in Java, you may not (depending on data structure, looping mechanism, etc.) be able to directly remove an item from the queue. The normal way to deal with this is to make a temporary data structure of items that need removal, then remove them all once you are not looping over the data structure.

**Task 5** – Change startup

Change startup to call the OS class instead of creating and using a new BasicScheduler.

**Test your code!**

***You must submit buildable .java files for credit.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rubric | Poor | OK | Good | Great |
| OS Class | None (0) | Exists (5) | Is singleton or implements all methods (10) | Is singleton and implements all methods (15) |
| BasicScheduler – Prep | None (0) | Has two queues (5) | Removes and replaces processes (10) | Remove and replace respects ordering (15) |
| Sleep Functionality - storage | None (0) | Implemented in a sub-optimal location (3) |  | Implemented in a logical place (5) |
| Sleep functionality - implementation | None (0) | Function updates the storage (5) |  | Function updates the storage and can be called from the singleton (10) |
| Switching – add/remove | None (0) | Processes are added and removed, but resultant run order is wrong (8) |  | Processes are added and removed and run order is correct (15) |
| Switching – wait list reduced | None (0) | Waiting process time is reduced on task switch (5) | Waiting processes are moved to the process list incorrectly (10) | Waiting processes are moved to the process list correctly (20) |
| Switching – wait list added to | None (0) |  |  | Processes that should be added to the wait list are added correctly (15) |
| Startup | None (0) |  |  | Startup calls OS(5) |