```
import java.util.LinkedList;

public class Swapper
{

    final int IN = 0;
    final int OUT = 1;

    /**
     * VARIABLES*****************************************************************
     */
    LinkedList<Integer> blockedOutQueue;    // To be swapped out
    LinkedList<Integer> blockedInQueue;     // To be swapped in
    LinkedList<Integer> defaultOutQueue;    // To be swapped out
    LinkedList<Integer> defaultInQueue;     // To be swapped in
    int inDrum;

    /**
     * CONSTRUCTOR***************************************************************
     */
    Swapper ()
    {
        blockedOutQueue = new LinkedList<Integer>();
        blockedInQueue = new LinkedList<Integer>();
        defaultOutQueue = new LinkedList<Integer>();
        defaultInQueue = new LinkedList<Integer>();
        inDrum = -1;
    }

    /**
     * PRIVATE METHODS***********************************************************
     */

    /**
     * Adds new swap job to appropriate queue
     * @param jobID the job to be added
     */
    void add (int jobID)
    {
        if (jobID != inDrum) {
            remove(jobID);
            // In Queues
            if (JobTable.getDirection(jobID) == 0) {
                // Blocked
                if (JobTable.isBlocked(jobID)) {
                    blockedInQueue.add(jobID);
                }
                // Default
                else {
                    defaultInQueue.add(jobID);
                }
            }
            else {
                // Blocked
                if (JobTable.isBlocked(jobID)) {
                    blockedInQueue.remove((Integer)jobID);
                    blockedOutQueue.add(jobID);
```

```
            }
            // Default
            else {
                defaultOutQueue.add(jobID);
            }
        }
    }
}

/**
 * Removes any or all instances of job in queues
 * @param jobID job to be removed
 */
void remove (int jobID)
{
    blockedInQueue.remove((Integer)jobID);
    blockedOutQueue.remove((Integer)jobID);
    defaultInQueue.remove((Integer)jobID);
    defaultOutQueue.remove((Integer)jobID);
}

/**
 * PUBLIC METHODS************************************************************
 */
/**
 * Handles the actual searching and swap calling
 */
public int swap ()
{
    // System.out.println("-Swap Queues:");
    // System.out.println("--BlockedOutQueue has " + blockedOutQueue.size());
    // System.out.println("--BlockedInQueue has " + blockedInQueue.size());
    // System.out.println("--DefaultInQueue has " + defaultInQueue.size());
    // System.out.println("--DefaultOut Queue has " + defaultOutQueue.size());
    if (inDrum == -1) {
        if (!defaultInQueue.isEmpty()) {
            inDrum = defaultInQueue.remove();
        }
        else if (!blockedOutQueue.isEmpty()) {

            inDrum = blockedOutQueue.remove();
        }
        else if (!blockedInQueue.isEmpty()) {
            inDrum = blockedInQueue.remove();
        }
        else if (!defaultOutQueue.isEmpty()) {
            inDrum = defaultOutQueue.remove();
        }
        if (inDrum != -1) {
            if (JobTable.doingIO(inDrum)) {
                add(inDrum);
                inDrum = -1;
            }
            else {
                JobTable.setSwapping(inDrum);
                Job swapJob = JobTable.returnJob(inDrum);
                sos.siodrum (swapJob.idNum, swapJob.size,
```

```
                    swapJob.address, swapJob.direction);
                String descriptor = "";
                if (swapJob.direction == 0) {
                    descriptor = " to ";
                }
                else if (swapJob.direction == 1) {
                    descriptor = " from ";
                }
                // System.out.println("--Begin swapping Job " +
                //  swapJob.idNum +" with size " + swapJob.size +
                //  descriptor + swapJob.address);
            }
        }
    }
    return inDrum;
}

/**
 * Prints status of Drum and Drum Queue
 */
public void print ()
{
    // System.out.println("-Swap Report:");
    // System.out.println("--In Drum : " + inDrum);
    // System.out.print("--In Queue: ");
    // System.out.print("DefaultIn:");
    // for (int i = 0; i < defaultInQueue.size(); i++)
    // {
    //  System.out.print(defaultInQueue.get(i) + ", ");
    // }
    // System.out.print("BlockedOut:");
    // for (int i = 0; i < blockedOutQueue.size(); i++)
    // {
    //  System.out.print(blockedOutQueue.get(i) + ", ");
    // }
    // System.out.print("BlockedIn:");
    // for (int i = 0; i < blockedInQueue.size(); i++)
    // {
    //  System.out.print(blockedInQueue.get(i) + ", ");
    // }
    // System.out.print("DefaultOut:");
    // for (int i = 0; i < defaultOutQueue.size(); i++)
    // {
    //  System.out.print(defaultOutQueue.get(i) + ", ");
    // }
    // System.out.println("");
}

/**
 * Handles drmint
 * @return idNum of job drum-to-memory, -1 if memory-to-drum
 */
public int swapDone ()
{
    // System.out.println("-Swapper getting swap details");
    int jobID = inDrum;
    inDrum = -1;
```

```
        JobTable.stopSwapping(jobID);
        JobTable.resetPriorityTime(jobID);
        Job job = JobTable.returnJob(jobID);

        // Status
        if (job.direction == 1) {
            // System.out.println("--Memory-to-Drum done for job " + jobID);
            JobTable.setSwapped(jobID);
            JobTable.outMemory(jobID);
        }
        else {
            // System.out.println("--Drum-to-Memory done for job " + jobID);
            JobTable.inMemory(jobID);
        }
        return jobID;
    }

    /**
     * [swapIn description]
     * @param jobID [description]
     */
    public void swapIn (int jobID)
    {
        if (jobID != -1) {
            // System.out.println("-Swapper beginning swap in of Job " + jobID);
            // System.out.println("--Added Job " + jobID + " to swap queue");
            JobTable.setDirection(jobID, 0);
            add(jobID);
        }
    }

    /**
     * [swapOut description]
     * @param jobID [description]
     */
    public void swapOut (int jobID)
    {
        if(jobID != -1) {
            // System.out.println("-Swapper beginning swap out of Job " +
            //  jobID);
            // System.out.println("--Added Job " + jobID + " to swap queue");
            JobTable.setDirection(jobID, 1);
            add(jobID);
        }
    }
}
```