package chapter7;

public interface QueueInterface<T> {

// Adds a new entry to the back of the queue.

public void enqueue( T newEntry);

// Removes and returns the entry at the front of the queue.

public void dequeue();

// Returns the front/first entry of the queue.

public T getFront();

// Returns true if the queue is empty and false if not.

public boolean isEmpty();

// Empties the queue

public void clear();

}

package chapter7;

public interface DequeInterface <T> {

// Add a new entry to the front or back of the deque.

public void addToFront(T newEntry);

public void addToBack(T newEntry);

// Removes and returns the entry from the front or back of the deque.

public T removeFront();

public T removeBack();

// Returns the entry from the front or back of the deque.

public T getFront();

public T getBack();

// Returns true if deque is empty and false if not.

public boolean isEmpty();

// Empties the deque.

public void clear();

package chapter7;

public class Assignment implements Comparable {

// Attributes for the object.

private String course;

private String task;

private date date;

// Default constructor.

public Assignment() {

}

// 3-parameter constructor.

public Assignment(String course, String task, String date) {

this.course = course;

this.task = task;

this.date = date;

}

public String getCourse() {

return course;

}

public String getTask() {

return task;

}

public String getdate() {

return date;

}

@Override

public int compareTo(Object o) {

// Prioritize based on due date.

return compareToByDueDate((Assignment)o);

// Prioritize based on course code

return compareToByCourseCode((Assignment)o);

return Comparator.comparing(Assignment::getCourse)

.thenComparing(Assignment::getTask)

.thenComparing(Assignment::getdate);

}

private int compareToByDueDate(Assignment assignment) {  
 return date.compareTo(assignment.date);

}

private int compareToByCourseCode(Assignment assignment) {  
 return date.compareTo(assignment.course);

}

package chapter7;

public class AssignmentLog {

private PriorityQueue <Assignment> log;

public AssignmentLog() {

log = new PriorityQueue<>();  
}

public void addProject(Assignment newAssignment) {

log.add(new Assignment);

}

public void addProject(String courseCode, String task, Date dueDate) {

Assignment newAssignment = new Assignment(courseCode, task, dueDate);

addProject(newAssignment);

}

public Assignment getNextProject() {

return log.peek();

}

public Assignment removeNextProject() {

return log.remove();

}