Assignment 3

DUE Wednesday Sept 20, 11:59pm

**Assignment 3 Part I is Part I of Assignment 3 and is worth 30% of the grade. Lab03B is also worth 30% of the grade of Assignment 3.**

[**Assignment 3 part 1 as a PDF**](http://www.cs.binghamton.edu/~lander/cs140/Fall17Assig3Part1.pdf)**A typo was just fixed and uploaded--I think we announced it some time ago. I do not know why the upload of the correction failed)**

**This is the remaining 40% is the following:**

Make a class CalendarItem that imports java.time.LocalDateTime. The 6 private fields are: two LocalDateTime fields called start and end, three String fields called title, description and location, and one CalendarItem field called nextItem.

We need *getters* and *setters* for all fields.

Add the lines

if(start != null && start.isAfter(end))

throw new IllegalArgumentException("the start cannot be after end");

at the beginning of the setEnd and similar code to the beginning of setStart.

Run the following main method:

# public static void Assignment 3

DUE Wednesday Sept 20, 11:59pm

**Assignment 3 Part I is Part I of Assignment 3 and is worth 30% of the grade. Lab03B is also worth 30% of the grade of Assignment 3.**

[**Assignment 3 part 1 as a PDF**](http://www.cs.binghamton.edu/~lander/cs140/Fall17Assig3Part1.pdf)**A typo was just fixed and uploaded--I think we announced it some time ago. I do not know why the upload of the correction failed)**

**This is the remaining 40% is the following:**

Make a class CalendarItem that imports java.time.LocalDateTime. The 6 private fields are: two LocalDateTime fields called start and end, three String fields called title, description and location, and one CalendarItem field called nextItem.

We need *getters* and *setters* for all fields.

Add the lines

if(start != null && start.isAfter(end))

throw new IllegalArgumentException("the start cannot be after end");

at the beginning of the setEnd and similar code to the beginning of setStart.

Run the following main method:

public static void main(String[] args) {

CalendarItem test = new CalendarItem();

test.setDescription("this is just a sample meeting of CS 140");

test.setEnd(LocalDateTime.of(2017,9,11,11,50));

test.setStart(LocalDateTime.of(2017,9,11,10,50));

test.setTitle("CS 140");

test.setLocation("FA 212");

System.out.println(test);

}

The output is very disappointing, so liven it up with the following toString method

// at the start of the class import java.time.format.DateTimeFormatter

public String toString() {

DateTimeFormatter dayForm = DateTimeFormatter.ofPattern("MMMM d, yyyy");

DateTimeFormatter timeForm = DateTimeFormatter.ofPattern("h:m a");

return start.format(dayForm) +" from " + start.format(timeForm)

+ " to " + end.format(timeForm) + "\n"

+ title + " in " + location + "\n"

+ description;

}

There is a defect and that is that it does not print completely an appointment that goes across midnight, say from 11:30pm to 12:30am. It only shows the date when the appointment starts.

Objects of type LocalDateTime can be compared to see which is first: the meaning of dateTime1.isBefore(dateTime2) and dateTime1.isAfter(dateTime2) should be obvious but you can see [API page for LocalDateTime](http://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html)

Provide a method public boolean endsBefore(CalendarItem other) that returns the value of end.isBefore(other.start)

Provide a method public boolean startsAfter(CalendarItem other) that returns the value of start.isAfter(other.end)

Add the following 2 methods:

public boolean noTimeConflictWith(CalendarItem other) {

return endsBefore(other) || startsAfter(other);

}

public boolean hasTimeConflictWith(CalendarItem other) {

return !noTimeConflictWith(other);

}

Write test cases in the main method above with various CalendarItems, checking cases where they do overlap in time and cases where they do not overlap.

Write a class that starts:

package assignment03;

import java.time.LocalDateTime;

import java.util.Optional;

public class CalendarList {

    private CalendarItem first;

    private CalendarItem next;

The idea is that first is the beginning of a linked list of nodes, which are CalendarItems, and next is the next appointment coming up (previous CalendarItems before next in the list are those that are already past).

We will assume that all the code will ensure that the linked list of CalendarItems are in the correct time sequence, going from the earliest appointments to the latest (treat CalendarItem and appointment as meaning the same).

The method call LocalDateTime.now() gives an object with the current date and time. We will call this the **"now object"**. Use this object in the following method

public void setNext()

which runs through all the CalendarItems in the linked list, starting at first and sets next to reference the*first*CalendarItem that has a start in the future or is right now (use isEqual || isAfter the "now object"). **NOTE this will involve adding a method isEqual to CalendarItem that checks (say) that the startTimes are the same. It also involves turning the "now object" into a "nowItem" by setting the start and end of nowIem to the now object. If you have solved this a different way, then stay with your method.**

**YOU CANNOT USE (start == other.start) TO COMPARE DATES. With any reference type, there are only very few special cases where == will work; it WILL NOT work here. You need to use (start.equals(other.start)) or (start.isEqual(other.start))**

Write a method public void resetFirst(), which first calls setNext() and then sets first to next.

This operation wipes out the history of past appointments.

Write a method public Optional<CalendarItem> findConflict(CalendarItem newItem), which loops through the whole list until it finds a CalendarItem in the list that has a time conflict with newItem (use hasTimeConflictWith in CalendarItem)

Return Optional.empty() if there are no conflicts.

You run through a list using temp = first, checking temp and then while(temp.getNextItem() != null) set temp = temp.getNextItem() and checking temp.

Write a method private void insert(CalendarItem newItem), which will insert a new appointment in the list of appointments assuming there are no time conflicts. It is *private* because it can only be called from the insertItemmethod below that checks for conflicts before calling this one. **If first is null, then this means the CalendarList is empty, so we set first equal to newItem and then return.**If newItem endsBefore first, then set the nextItem of newItem to first and change first to be newItem. Otherwise loop through the list with a tempvariable until newItem endsBefore temp.getNextItem. At that point set the nextItem of newItem to temp.getNextItem and change the nextItem of temp to newItem. 

Write a method public void insertItem(CalendarItem newItem), to insert a new appointment but checks for conflicts or other problems. **If** first **is not** null, it first calls setNext(). It then gets the "now object" and

if newItem ends before the nowObject throw new IllegalArgumentException("It is too late for that appointment");

if newItem starts before the nowObject throw new IllegalArgumentException("That appointment is under way--run!");

**If** first **is not** null: **get** Optional<CalendarItem> temp = findConflict(newItem);

if(temp.isPresent()) {

throw new IllegalArgumentException("You have a conflict. "

+ "Please use findConflict to find the conflicting appointment");

}

After all these checks are passes you can call insert(newItem).

***Note that you do not need an "else" after throwing an exception.***

Write methods public void printNext(), which prints just the next CalendarItem (it will automatically use the toString of CalendarItem) and public void printCalendar(), which loops through all the CalendarItems starting at first and prints each one.

Write a class CalendarTester with a main method containing lots of tests.

**TESTING: We found it will help to have a package private method, say void testingInsert(CalendarItem), which has the conflict test but not the test comparing with "now" to, so that it allows insertion of appointments in the past. We need this to test resetFirst: we want to have a bunch of appointments in the past that can be thrown away after calling resetFirst. (A package private method is one that is not public, private, or protected and is only visible form other classes in the same package.)**

[assignment3](https://piazza.com/class/j2pqenz45a72r?cid=71)

Updated 1 hour ago by Leslie Lander and Shawn Bailey

**followup discussions**

*for lingering questions and comments*

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**5 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Can I get clarification as to how I should write the setNext method? Based on my interpretation of it, we're supposed to create a temporary calendarItem object and use it along with first, next, and getNextItem to cycle through the entire list in a while loop until next is null, but assuming that I do have the right idea, I'm not 100% sure how to put it into practice.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [4 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If you cycle through till next is null, then you would be inserting at the end of the list, which would mean you are adding a CalandarItem that starts later than any other item in the list. If someone has asked you to add an appointment that ends earlier than the last element in the list, then you have to insert the new temp calendar item in between two existing CalendarItems.

Reply to this followup discussion

Resolved Unresolved



[**Matthew Van Gorden**](https://piazza.com/class/j2pqenz45a72r?cid=71) [4 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

How do we access the linked list of nodes? Where do we get the value for the CalendarItem after next?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [4 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The instances of the class CalendarList have a field first. That is the fist element of linked list of CalendarItem nodes. In this problem a *node* is a CalendarItem.



[**Matthew Van Gorden**](https://piazza.com/class/j2pqenz45a72r?cid=71) [2 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Why is it necessary that I iterate through all the items in the linked list if they are in order? If I iterate in chronological order, then the first one that satisfies isEqual || isAfter the now object would be the correct appointment.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [2 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Right, you just iterate through from the beginning (first) until you find the correct place for an insert.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**3 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I have several questions regarding this assignment:

1. In setNext(), do we assume that there will always be a CalendarItem after the now object? For instance, I do not know what to do if there is only one calendarItem (set to first), yet the now object is past the start of first. Do we just set next to null?

2. In findConflict(), what should we return if there is a conflict? The CalendarItem of which newItems conflicts with? What do we do if first is set to null; do we just return Optional.empty(), indicating that there are no conflicts?

3. In insert(), again, what do we do if first is set to null? Should we just set first to newItem?

4. For printNext(), should we just print the next object, or do you want us to call setNext() and then print the next object?

5. Should resetFirst() be used anywhere in CalendarList, or is it only to be called/tested in main() of CalendarTester? Should we make a getFirst() method to test that first is being set correctly? Also, how can we test resetFirst() if we cannot make a node that begins before the now object (because we check for this in insertItem())?

6. Do we need to explicitly test insert(), findConflict(), and setNext(), or can these be tested through testing insertItem()?

7. Should setNext() be called again at the end of insertItem() so if the added item is next, the next field can be appropriately updated?

Sorry for so many questions!



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [3 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

(1) No, you may be inserting the last item in the list and all the others may already be past

(2) If there is a conflicting CalenderItem someItem in findConflict then it returns Optional.of(someItem)

(3) correct, if first is null, the item being inserted becomes first.

(4) In think my intention was just to print what next is referencing. The user would probably want to call setNext themselves before calling printNext. You will not lose points if you already wrote code that also called setNext()

(5) I think you have to make a list and print various insertions at different times, before calling resetFirst and seeing that all the old items have disappeared from the list. After doing the reset you need to insert a variety of items to make sure you did not break something by the reset.

(6) Really should should have been testing all those methods along the way. If you test a combination of methods and something is wrong, it is very hard to find the problem. Test every method you write.

(7) My intention was that setNext would either be called by the user or by the resetFirst method and nowhere else.



[**Chris Glasheen**](https://piazza.com/class/j2pqenz45a72r?cid=71) [9 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

You say that we're only supposed to call setNext() either by ourselves in main or in resetFirst(), but the assignment specifically says to call it in the insertItem method. Should we still call it there?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [8 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Yes it is specified there too. I forgot when I made the comment. You must check first is not null before calling setNext or findConflict in insertItem

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**3 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I am confused as to where we are supposed to store our list of CalendarItems. Are we supposed to create a linkedList and store them in there?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [2 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Assuming you have written the methods correctly, you can create a new CalendarItem, using similar lines that we see in the main of CalendarItem.java. Once you have created the CalendarItem, you can add it to the CalendarList using the insert() method. Just keep making CalendarItems, and use the insert() method to add them. This should build the Linked List for you. The Linked List is "stored" in CalendarList's first field. This first field is a reference to the first scheduled CalendarItem of the list, and the next event is stored in first's nextItem field and so forth.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

For the insert method, we are to "set the nextItem of newItem to first" but since we don't have a setNextItem() method, how would we do that? Would we use setNext() in some capacity, or am I just missing obvious something here?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Make a class CalendarItem that imports java.time.LocalDateTime. The 6 private fields are: two LocalDateTime fields called start and end, three String fields called title, description and location, and one CalendarItem field called **nextItem**.

**We need *getters* and *setters* for all fields.**

You should have a setNextItem() method inside the CalendarItem that you can use to set the nextItem.

Hope that clarifies.

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Must have missed that. Thanks

Reply to this followup discussion

Resolved Unresolved



[**Zachary Marshman**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I'm confused on the use of the now item.  Why aren't we just comparing first and next?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

First represents the entire linked list of the Calendar Items (your schedule). Next is just meant to point to the next Calendar Item within that linked list that is either happening right now, or will happen the earliest in the future. We are comparing each item in the linked list (which again, begins at first) to the now object, so that we can set next to the next event that will occur on our schedule. The now object is a nice way to see which event in our Linked List fits these properties. Keep in mind, if a Calendar Item is for an event that already happened in the past, then we missed it and cannot go to it. We want to skip such scheduled events, and only worry about those that are happening in the present (right now) or in the nearest future.

Hope this helps.



[**Zachary Marshman**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

So are we creating a now object before testing isEqual or isAfter?  and is "isEqual" comparing the end of now to the start of first?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Right, you will have to make the now object first for comparison, either as a LocalDateTime as before the professor's update, or as a nowItem.

But, you are comparing the **start times** of the now object with a CalendarItem in the linked list.

If you compare the starts times of both and they are equal, that means that Linked List Calendar Item is starting **right now.**

If you compare the start times and the Linked List item starts **after** the now start time, that means the Linked List Calendar Item is starting sometime **in the future.**

In either case, you have found the next CalendarItem on your schedule that you need to worry about attending.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If you make a temporary CalendarItem with a start and end time equal to "now" you can use your isAfter method. You next need a way to see if the start of one of the CalendarItems is the same as "now"--you can do that with either writing an isEqual method (that compares start with the start time of your temporary CalendarItem using the equals of LocalDateTime), or you add a getStart() getter method and do that comparison with "now" itself.

I do not want to to override the "equals" method for CalendarItems because (i) we have not reached that topic in class and (ii) the only real meaning would be that two CalendarItems are equal when they have all 5 fields equal, so it would no be of use to what we are asking in this question.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Hi, the submission of assignment3 has not been posted yet, could you make a submission.



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The submission link has been added to mycourses. Thanks for reminding me.

Reply to this followup discussion

Resolved Unresolved



[**Connor Tremblay**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Are we supposed to do the part that is in the PDF file? There is no mention of it other than the link so I am confused



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Yes, you need to do that part as well.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Do I need to write a getFirst() method in my NodeList class in order to call sieve(first) with my NodeList object?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

No, first is a private field **within** the class NodeList. Any methods written as a part of Nodelist can access the private field of the Nodelist class, because they reside within the class too.

the method sieve lives in NodeList, so it can directly access the private field first, no need for a getter method here.

Hope this helps.

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If i call sieve(first) in my main method it says I cannot reference first from a static context. Wouldn't a getFirst() method solve this?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

You should be calling sieve(first) on a NodeList object you have made in the main. It will look something like this:

NodeList l = new NodeList();

...

l.sieve(first);

That should solve your issue. You should be calling all these methods on an instance of NodeList you have made with the NodeList constructor. This will solve the static issue because the NodeList has to be made at runtime, it is definitely not static, so the issue is moot from that point.



[**Connor Tremblay**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The error stating that you cannot reference first from static context is because you have first initialized to a non-static variable. To reference first, you must initialize it as

public static Node first;



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Take static out. In Part 1 and Part 2, "first" is an instance field of the NodeList and CalendarList classes.



[**Chris Glasheen**](https://piazza.com/class/j2pqenz45a72r?cid=71) [19 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If first is a private instance field of NodeList, how are we supposed to reference it in main without a getter method? If I create a NodeList in main called test and try to do test.sieve(first), it won't work. However, if I have a getter method for first called getFirst() and then try test.sieve(test.getFirst()), it works perfectly. The only other thing I can think of that works is defining first as "private *static* Node first" but you don't want us to use static.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [8 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I used list.sieve(list.first)

OK if your main is in the NodeList class

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

For the testingInsert() method, is it basically the same thing as InsertItem() but without the if() tests that throw exceptions?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

without the if's that test whether the item is before "now" you keep the test for conflicts. We only need this new method to fully test setNext, and resetFirst.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

What is the syntax for testing the now object in insertItem()? newItem.endsBefore(now) gives me an error saying that it wants a calendarItem in place of now, so I'm not quite sure how to test it.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

You can make a nowCalendarItem as a new CalendarItem with a start and end time set equal to "now".

Reply to this followup discussion

Resolved Unresolved



[**Kevin Wallace**](https://piazza.com/class/j2pqenz45a72r?cid=71) [21 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The sample output for NodeList.sieve() in the Part 1 pdf gives all the prime numbers up to 83, but doesn't include 89. I'm assuming sieve() was called on the list made above, with max value 90. Should the output of sieve(first) on a NodeList with max 90 include 89?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [21 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The code may need a minor tweak to print the last element. Maybe you have an old version of the PDF. Do you have System.out.println(temp.getValue()); at the end of listNodeValues?

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 hour ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Im so confused on the CalendarList class. I dont understand how to make this "now" object nor do i understand how to even begin. Im trying to implement the setNext() function but im not sure how to go about it. Like someone previously said, I made a temp CalendarItem and set it  = to first. After i Have a while loop saying while(temp!= null) but im not sure how to proceed from there. Any type of help would be appreciated.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [53 minutes ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

CalendarItem nowObject = new CalendarItem();

nowObject.setStart(LocalDateTime.now());

nowObject.setEnd(LocalDateTime.now());

// assuming you have made a getEndTime()

LocalDateTime now = LocalDateTime.now();

while(temp != null && (temp.getEndTime().isEqual(now) || temp.getEndTime().isBefore(now)))

temp = nem.getNextItem();

Reply to this followup discussion

**Start a new followup discussion**

main(String[] args) {

CalendarItem test = new CalendarItem();

test.setDescription("this is just a sample meeting of CS 140");

test.setEnd(LocalDateTime.of(2017,9,11,11,50));

test.setStart(LocalDateTime.of(2017,9,11,10,50));

test.setTitle("CS 140");

test.setLocation("FA 212");

System.out.println(test);

}

The output is very disappointing, so liven it up with the following toString method

// at the start of the class import java.time.format.DateTimeFormatter

public String toString() {

DateTimeFormatter dayForm = DateTimeFormatter.ofPattern("MMMM d, yyyy");

DateTimeFormatter timeForm = DateTimeFormatter.ofPattern("h:m a");

return start.format(dayForm) +" from " + start.format(timeForm)

+ " to " + end.format(timeForm) + "\n"

+ title + " in " + location + "\n"

+ description;

}

There is a defect and that is that it does not print completely an appointment that goes across midnight, say from 11:30pm to 12:30am. It only shows the date when the appointment starts.

Objects of type LocalDateTime can be compared to see which is first: the meaning of dateTime1.isBefore(dateTime2) and dateTime1.isAfter(dateTime2) should be obvious but you can see [API page for LocalDateTime](http://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html)

Provide a method public boolean endsBefore(CalendarItem other) that returns the value of end.isBefore(other.start)

Provide a method public boolean startsAfter(CalendarItem other) that returns the value of start.isAfter(other.end)

Add the following 2 methods:

public boolean noTimeConflictWith(CalendarItem other) {

return endsBefore(other) || startsAfter(other);

}

public boolean hasTimeConflictWith(CalendarItem other) {

return !noTimeConflictWith(other);

}

Write test cases in the main method above with various CalendarItems, checking cases where they do overlap in time and cases where they do not overlap.

Write a class that starts:

package assignment03;

import java.time.LocalDateTime;

import java.util.Optional;

public class CalendarList {

    private CalendarItem first;

    private CalendarItem next;

The idea is that first is the beginning of a linked list of nodes, which are CalendarItems, and next is the next appointment coming up (previous CalendarItems before next in the list are those that are already past).

We will assume that all the code will ensure that the linked list of CalendarItems are in the correct time sequence, going from the earliest appointments to the latest (treat CalendarItem and appointment as meaning the same).

The method call LocalDateTime.now() gives an object with the current date and time. We will call this the **"now object"**. Use this object in the following method

public void setNext()

which runs through all the CalendarItems in the linked list, starting at first and sets next to reference the*first*CalendarItem that has a start in the future or is right now (use isEqual || isAfter the "now object"). **NOTE this will involve adding a method isEqual to CalendarItem that checks (say) that the startTimes are the same. It also involves turning the "now object" into a "nowItem" by setting the start and end of nowIem to the now object. If you have solved this a different way, then stay with your method.**

**YOU CANNOT USE (start == other.start) TO COMPARE DATES. With any reference type, there are only very few special cases where == will work; it WILL NOT work here. You need to use (start.equals(other.start)) or (start.isEqual(other.start))**

Write a method public void resetFirst(), which first calls setNext() and then sets first to next.

This operation wipes out the history of past appointments.

Write a method public Optional<CalendarItem> findConflict(CalendarItem newItem), which loops through the whole list until it finds a CalendarItem in the list that has a time conflict with newItem (use hasTimeConflictWith in CalendarItem)

Return Optional.empty() if there are no conflicts.

You run through a list using temp = first, checking temp and then while(temp.getNextItem() != null) set temp = temp.getNextItem() and checking temp.

Write a method private void insert(CalendarItem newItem), which will insert a new appointment in the list of appointments assuming there are no time conflicts. It is *private* because it can only be called from the insertItemmethod below that checks for conflicts before calling this one. **If first is null, then this means the CalendarList is empty, so we set first equal to newItem and then return.**If newItem endsBefore first, then set the nextItem of newItem to first and change first to be newItem. Otherwise loop through the list with a tempvariable until newItem endsBefore temp.getNextItem. At that point set the nextItem of newItem to temp.getNextItem and change the nextItem of temp to newItem. 

Write a method public void insertItem(CalendarItem newItem), to insert a new appointment but checks for conflicts or other problems. **If** first **is not** null, it first calls setNext(). It then gets the "now object" and

if newItem ends before the nowObject throw new IllegalArgumentException("It is too late for that appointment");

if newItem starts before the nowObject throw new IllegalArgumentException("That appointment is under way--run!");

**If** first **is not** null: **get** Optional<CalendarItem> temp = findConflict(newItem);

if(temp.isPresent()) {

throw new IllegalArgumentException("You have a conflict. "

+ "Please use findConflict to find the conflicting appointment");

}

After all these checks are passes you can call insert(newItem).

***Note that you do not need an "else" after throwing an exception.***

Write methods public void printNext(), which prints just the next CalendarItem (it will automatically use the toString of CalendarItem) and public void printCalendar(), which loops through all the CalendarItems starting at first and prints each one.

Write a class CalendarTester with a main method containing lots of tests.

**TESTING: We found it will help to have a package private method, say void testingInsert(CalendarItem), which has the conflict test but not the test comparing with "now" to, so that it allows insertion of appointments in the past. We need this to test resetFirst: we want to have a bunch of appointments in the past that can be thrown away after calling resetFirst. (A package private method is one that is not public, private, or protected and is only visible form other classes in the same package.)**

[assignment3](https://piazza.com/class/j2pqenz45a72r?cid=71)

Updated 1 hour ago by Leslie Lander and Shawn Bailey

**followup discussions**

*for lingering questions and comments*

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**5 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Can I get clarification as to how I should write the setNext method? Based on my interpretation of it, we're supposed to create a temporary calendarItem object and use it along with first, next, and getNextItem to cycle through the entire list in a while loop until next is null, but assuming that I do have the right idea, I'm not 100% sure how to put it into practice.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [4 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If you cycle through till next is null, then you would be inserting at the end of the list, which would mean you are adding a CalandarItem that starts later than any other item in the list. If someone has asked you to add an appointment that ends earlier than the last element in the list, then you have to insert the new temp calendar item in between two existing CalendarItems.

Reply to this followup discussion

Resolved Unresolved



[**Matthew Van Gorden**](https://piazza.com/class/j2pqenz45a72r?cid=71) [4 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

How do we access the linked list of nodes? Where do we get the value for the CalendarItem after next?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [4 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The instances of the class CalendarList have a field first. That is the fist element of linked list of CalendarItem nodes. In this problem a *node* is a CalendarItem.



[**Matthew Van Gorden**](https://piazza.com/class/j2pqenz45a72r?cid=71) [2 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Why is it necessary that I iterate through all the items in the linked list if they are in order? If I iterate in chronological order, then the first one that satisfies isEqual || isAfter the now object would be the correct appointment.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [2 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Right, you just iterate through from the beginning (first) until you find the correct place for an insert.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**3 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I have several questions regarding this assignment:

1. In setNext(), do we assume that there will always be a CalendarItem after the now object? For instance, I do not know what to do if there is only one calendarItem (set to first), yet the now object is past the start of first. Do we just set next to null?

2. In findConflict(), what should we return if there is a conflict? The CalendarItem of which newItems conflicts with? What do we do if first is set to null; do we just return Optional.empty(), indicating that there are no conflicts?

3. In insert(), again, what do we do if first is set to null? Should we just set first to newItem?

4. For printNext(), should we just print the next object, or do you want us to call setNext() and then print the next object?

5. Should resetFirst() be used anywhere in CalendarList, or is it only to be called/tested in main() of CalendarTester? Should we make a getFirst() method to test that first is being set correctly? Also, how can we test resetFirst() if we cannot make a node that begins before the now object (because we check for this in insertItem())?

6. Do we need to explicitly test insert(), findConflict(), and setNext(), or can these be tested through testing insertItem()?

7. Should setNext() be called again at the end of insertItem() so if the added item is next, the next field can be appropriately updated?

Sorry for so many questions!



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [3 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

(1) No, you may be inserting the last item in the list and all the others may already be past

(2) If there is a conflicting CalenderItem someItem in findConflict then it returns Optional.of(someItem)

(3) correct, if first is null, the item being inserted becomes first.

(4) In think my intention was just to print what next is referencing. The user would probably want to call setNext themselves before calling printNext. You will not lose points if you already wrote code that also called setNext()

(5) I think you have to make a list and print various insertions at different times, before calling resetFirst and seeing that all the old items have disappeared from the list. After doing the reset you need to insert a variety of items to make sure you did not break something by the reset.

(6) Really should should have been testing all those methods along the way. If you test a combination of methods and something is wrong, it is very hard to find the problem. Test every method you write.

(7) My intention was that setNext would either be called by the user or by the resetFirst method and nowhere else.



[**Chris Glasheen**](https://piazza.com/class/j2pqenz45a72r?cid=71) [9 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

You say that we're only supposed to call setNext() either by ourselves in main or in resetFirst(), but the assignment specifically says to call it in the insertItem method. Should we still call it there?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [8 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Yes it is specified there too. I forgot when I made the comment. You must check first is not null before calling setNext or findConflict in insertItem

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**3 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I am confused as to where we are supposed to store our list of CalendarItems. Are we supposed to create a linkedList and store them in there?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [2 days ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Assuming you have written the methods correctly, you can create a new CalendarItem, using similar lines that we see in the main of CalendarItem.java. Once you have created the CalendarItem, you can add it to the CalendarList using the insert() method. Just keep making CalendarItems, and use the insert() method to add them. This should build the Linked List for you. The Linked List is "stored" in CalendarList's first field. This first field is a reference to the first scheduled CalendarItem of the list, and the next event is stored in first's nextItem field and so forth.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

For the insert method, we are to "set the nextItem of newItem to first" but since we don't have a setNextItem() method, how would we do that? Would we use setNext() in some capacity, or am I just missing obvious something here?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Make a class CalendarItem that imports java.time.LocalDateTime. The 6 private fields are: two LocalDateTime fields called start and end, three String fields called title, description and location, and one CalendarItem field called **nextItem**.

**We need *getters* and *setters* for all fields.**

You should have a setNextItem() method inside the CalendarItem that you can use to set the nextItem.

Hope that clarifies.

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Must have missed that. Thanks

Reply to this followup discussion

Resolved Unresolved



[**Zachary Marshman**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I'm confused on the use of the now item.  Why aren't we just comparing first and next?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

First represents the entire linked list of the Calendar Items (your schedule). Next is just meant to point to the next Calendar Item within that linked list that is either happening right now, or will happen the earliest in the future. We are comparing each item in the linked list (which again, begins at first) to the now object, so that we can set next to the next event that will occur on our schedule. The now object is a nice way to see which event in our Linked List fits these properties. Keep in mind, if a Calendar Item is for an event that already happened in the past, then we missed it and cannot go to it. We want to skip such scheduled events, and only worry about those that are happening in the present (right now) or in the nearest future.

Hope this helps.



[**Zachary Marshman**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

So are we creating a now object before testing isEqual or isAfter?  and is "isEqual" comparing the end of now to the start of first?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Right, you will have to make the now object first for comparison, either as a LocalDateTime as before the professor's update, or as a nowItem.

But, you are comparing the **start times** of the now object with a CalendarItem in the linked list.

If you compare the starts times of both and they are equal, that means that Linked List Calendar Item is starting **right now.**

If you compare the start times and the Linked List item starts **after** the now start time, that means the Linked List Calendar Item is starting sometime **in the future.**

In either case, you have found the next CalendarItem on your schedule that you need to worry about attending.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If you make a temporary CalendarItem with a start and end time equal to "now" you can use your isAfter method. You next need a way to see if the start of one of the CalendarItems is the same as "now"--you can do that with either writing an isEqual method (that compares start with the start time of your temporary CalendarItem using the equals of LocalDateTime), or you add a getStart() getter method and do that comparison with "now" itself.

I do not want to to override the "equals" method for CalendarItems because (i) we have not reached that topic in class and (ii) the only real meaning would be that two CalendarItems are equal when they have all 5 fields equal, so it would no be of use to what we are asking in this question.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Hi, the submission of assignment3 has not been posted yet, could you make a submission.



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The submission link has been added to mycourses. Thanks for reminding me.

Reply to this followup discussion

Resolved Unresolved



[**Connor Tremblay**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Are we supposed to do the part that is in the PDF file? There is no mention of it other than the link so I am confused



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Yes, you need to do that part as well.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Do I need to write a getFirst() method in my NodeList class in order to call sieve(first) with my NodeList object?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

No, first is a private field **within** the class NodeList. Any methods written as a part of Nodelist can access the private field of the Nodelist class, because they reside within the class too.

the method sieve lives in NodeList, so it can directly access the private field first, no need for a getter method here.

Hope this helps.

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If i call sieve(first) in my main method it says I cannot reference first from a static context. Wouldn't a getFirst() method solve this?



[**Shawn Bailey**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

You should be calling sieve(first) on a NodeList object you have made in the main. It will look something like this:

NodeList l = new NodeList();

...

l.sieve(first);

That should solve your issue. You should be calling all these methods on an instance of NodeList you have made with the NodeList constructor. This will solve the static issue because the NodeList has to be made at runtime, it is definitely not static, so the issue is moot from that point.



[**Connor Tremblay**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The error stating that you cannot reference first from static context is because you have first initialized to a non-static variable. To reference first, you must initialize it as

public static Node first;



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Take static out. In Part 1 and Part 2, "first" is an instance field of the NodeList and CalendarList classes.



[**Chris Glasheen**](https://piazza.com/class/j2pqenz45a72r?cid=71) [19 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

If first is a private instance field of NodeList, how are we supposed to reference it in main without a getter method? If I create a NodeList in main called test and try to do test.sieve(first), it won't work. However, if I have a getter method for first called getFirst() and then try test.sieve(test.getFirst()), it works perfectly. The only other thing I can think of that works is defining first as "private *static* Node first" but you don't want us to use static.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [8 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

I used list.sieve(list.first)

OK if your main is in the NodeList class

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

For the testingInsert() method, is it basically the same thing as InsertItem() but without the if() tests that throw exceptions?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

without the if's that test whether the item is before "now" you keep the test for conflicts. We only need this new method to fully test setNext, and resetFirst.

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

What is the syntax for testing the now object in insertItem()? newItem.endsBefore(now) gives me an error saying that it wants a calendarItem in place of now, so I'm not quite sure how to test it.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [1 day ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

You can make a nowCalendarItem as a new CalendarItem with a start and end time set equal to "now".

Reply to this followup discussion

Resolved Unresolved



[**Kevin Wallace**](https://piazza.com/class/j2pqenz45a72r?cid=71) [21 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The sample output for NodeList.sieve() in the Part 1 pdf gives all the prime numbers up to 83, but doesn't include 89. I'm assuming sieve() was called on the list made above, with max value 90. Should the output of sieve(first) on a NodeList with max 90 include 89?



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [21 hours ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

The code may need a minor tweak to print the last element. Maybe you have an old version of the PDF. Do you have System.out.println(temp.getValue()); at the end of listNodeValues?

Reply to this followup discussion

Resolved Unresolved

https://dvngeac8rg9mb.cloudfront.net/images/dashboard/common/default_anonymous.png

[**Anonymous**1 hour ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

Im so confused on the CalendarList class. I dont understand how to make this "now" object nor do i understand how to even begin. Im trying to implement the setNext() function but im not sure how to go about it. Like someone previously said, I made a temp CalendarItem and set it  = to first. After i Have a while loop saying while(temp!= null) but im not sure how to proceed from there. Any type of help would be appreciated.



[**Leslie Lander**](https://piazza.com/class/j2pqenz45a72r?cid=71) [53 minutes ago](https://piazza.com/class/j2pqenz45a72r?cid=71)

CalendarItem nowObject = new CalendarItem();

nowObject.setStart(LocalDateTime.now());

nowObject.setEnd(LocalDateTime.now());

// assuming you have made a getEndTime()

LocalDateTime now = LocalDateTime.now();

while(temp != null && (temp.getEndTime().isEqual(now) || temp.getEndTime().isBefore(now)))

temp = nem.getNextItem();

Reply to this followup discussion

**Start a new followup discussion**