Homework #3

due Monday, September 24, 10:00 PM

This assignment will focus on the concepts of parsing strings for immutable classes, and using collections (especially iterators).

1 Concepts

1.1 Generics

Many of the library classes are generic so that they work with different types. We’ve already seen that with List. In this homework, we will work with several generic interfaces and classes. If you fail to give the actual generic type parameter (inside angle brackets, e.g., List<String>), Eclipse will warn you that you are using “raw types.” Raw types are not allowed for CS 351 homeworks, so make sure to take heed of the warnings.

1.2 Collections

The Java standard collection framework includes an interface Collection with a number of methods. See the textbook Figure 5.7, page 301 (3rd. ed. Fig. 5.6, p. 290), or look online.

In this homework, we are using the built-in library class TreeSet which additionally satisfies the SortedSet interface. The documentation for this interface is also online. You will only need to use the tailSet method from this interface.

1.3 Iterators

The collections classes and others provide iterators, which are more powerful than the “cursors” we implemented in the previous homework assignment. The big advantages are that one can have multiple iterators (whereas there’s just one cursor for each Sequence) and then moving an iterator does not affect the container. But since iterators are separate objects from the collections, if the collection changes, the iterator can go “stale.” Implementations try to prevent the usage of stale iterators, throwing an instance of ConcurrentModificationException but the checks are not foolproof.

Iterators have three methods:

hasNext() Returns whether any more elements remain.

next() Return the next element. May only be called if there are more elements.

remove() Removes the element previously returned by next(). If next() has not yet been called, or if the element has already been removed, then this method throws an IllegalStateException.

Each iterator moves separately through the container.

2 Concerning Updates to Duration, Time, Period and Appointment Classes

In order to facilitate importing and exporting appointments from the calendar (see below), we add a static method named fromString to each of the immutable ADTs from previous homework assignments.
The `fromString` method takes a string and returns an instance of the class it is declared in or throws a `NumberFormatException`. You will need to add this method to each of the four classes (copied into your repository) along with its javadoc. Additionally if the argument is null, the code is expected to throw a `NullPointerException`. The code should not throw any other exception.

Javadoc is written using a /** comment, and starts with a sentence describing the method. Details may be given in following sentences. Each parameter should be described with a @param tag, the return value with a @return tag and any exceptions with the @exception tag. Once you fully declare the signature of the method (including a throws declaration listing the possible exception being thrown), you can use Eclipse’s “Source > Generate Element Comment” command to give you a start.

In the case of the `Time` class, you will find that the `SimpleDateFormat` class has a useful method called `parse`. Please read the online documentation for this method. You may also need to use the `ParsePosition` class. The way the `parse` method signals errors is not the same as you need to do.

Of course the code for `Period#fromString` will call `fromString` from other classes. You will need to call `substring` to make the smaller strings. Our solution also uses the `charAt` and `indexOf` methods.

### 3 Concerning the Calendar ADT

The `Calendar` ADT is intended for the same purpose as the `ApptBook` ADT from the last assignment, but it has a completely different interface (it has different operations). In particular it follows the Java Collections framework in using iterators, and provides `SortedSet` operations (and `tailSet` in particular).

Unlike many other assignments, in this assignment you will not be acting on the data structure at all; instead you will implement `Calendar`-specific operations using other public operations:

- **doImport(Scanner, PrintWriter)** Read whole lines from the scanner and convert each to an appointment. If a number format exception is thrown, it should be caught and a message printed to the second argument. If there are no problems, nothing should be printed.

- **doExport(PrintWriter)** Write each appointment in the calendar to the print writer passed in. Make sure that each is on its own line.

- **starting(Time)** Return an iterator into the calendar starting at the time given. Unlike `setCurrent` in the previous assignment, this operation should be efficient—it shouldn’t iterate from the beginning. Instead it should use `tailSet` to efficiently start partway through the calendar.

- **removeConflicts()** Conflicts can be a problem in a calendar. Normally one might want to flag them in some way. For this assignment, we will simply remove conflicting appointments. The method will return a list of all the removed appointments.

Conflict removal should be done efficiently. As it happens, appointments cannot conflict unless one or the other one conflicts with a neighboring appointment. As a result, you can remove conflicts merely by making sure that neighboring appointments do not conflict.

### 4 What You Need To Do

You need to add `fromString` to the four immutable ADTs and finish four methods in the `Calendar` class.
5 Files

The directory homework3.git repository contains the following files:

src/TestFromString.java This class contains the tests for the fromString method for all four classes. It will have errors initially. Fix the errors by adding the methods to the ADTs.

dsrc/TestCalendar.java This class contains the tests for the four extra methods of the Calendar class.

dsrc/TestEfficiency.java Test to make sure you implemented the methods efficiently.

dsrc/UnlockTest.java Test driver that can unlock all the tests without executing them.

dsrc/edu/uwm/cs351/Calendar.java This is the skeleton file you should work with.

dsrc/edu/uwm/cs351/{Duration,Time,Period,Appointment}.java Implementations from previous homework assignments. You will need to add fromString.