# CPSC 1110 – LAB 04

More Inheritance (Chapter 9) and List Processing

This lab will deal with implementing a simulation of a simple calendar with appointments using the concepts of inheritance and subclassing. This is problem P9.1 from the book, with some modifications and additions. We will have an Appointment super class, and three types of appointments that inherit from the super class. Additionally, we will have a Calendar class that will simply be an ArrayList of Appointment objects. You may use BlueJ or Eclipse to complete the lab. (If you want to use some other IDE please talk to me about it). **PLEASE COMMENT YOUR CODE.** You will have points taken off if you do not comment your code. You can see sample comments in my starter code for how you should comment your code. Keep your code neat.

Zip all your .java files as well as a PDF file containing your output to submit to UTC Learn.

**Some useful links:**

BlueJ tutorial [www.bluej.org/tutorial/tutorial-201.pdf](http://www.bluej.org/tutorial/tutorial-201.pdf)

Java tutorial home page: <http://docs.oracle.com/javase/tutorial/>

Start here: <http://docs.oracle.com/javase/tutorial/java/index.html>

Arrays <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>

Array Lists <http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html>

Inheritance <http://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html>

**Some helpful tips:**

1. Compile often – do it.
2. You are responsible for a larger amount of the design process in this lab. You need to draw on paper how you want your subclasses to work and plan ahead before you actually write code.
3. It may be helpful to use the Debugger or print statements to check your work. (always).
4. We will have a demo/tester class for this lab called AppointmentDemo. You will need to fill in this class with enough tests to verify that your project works as expected. I have given you sample code to see how to use the public interface of the classes you will be implementing.
5. There are three types of Appointments we are creating, Onetime (Occurs one time on a single date), Daily (occurs every day after the given start date, including the given start date), and Monthly (Occurs on the same date each month, i.e. 1/1/2000, 2/1/2000, 3/1/2000).
6. Read Chapter 9, including the Special Topics and Common Errors.

## Tasks: Follow the directions below to complete your lab assignment

For today's lab we will be completing **Exercise P9.3** from the book (with modifications/additions).

**P9.3** Implement a superclass Appointment and subclasses Onetime, Daily, and Monthly. An appointment has a description (for example, “see the dentist”) and a date (use int's to store the date). Write a method occursOn(int year, int month, int day) that checks whether the appointment occurs on that date. For example, for a monthly appointment, you must check whether the day of the month matches. (Think about what you need to check for the other types of appointments. Ask if you have questions).

***(Skip the remaining problem description listed in the book, and follow these directions below).***

Additionally, you will need to implement a class called Calendar. This will contain an ArrayList of Appointment objects (The list is an instance variable of the Calendar class). You will need to provide the following methods for your Calendar class.

A no argument constructor

public Calendar(){...}

This constructor will initialize your ArrayList instance variable to an empty list.

/\*\*

\* A method to add an appointment to the calendar

\* **@param** apt – the appointment object to add to the calendar.

\*/

**public** **void** add(Appointment apt) {…}

/\*\*

\* A method to remove an appointment from the calendar.

\* This method uses the occursOn() method from the public

\* interface for the Appointment class. Therefore, if parameters

\* are entered that occur after a start date for a given Daily

\* appointment

\* the Daily appointment will be removed as well. (Because occursOn() \* willreturn true in this case). This is a limitation we will

\* accept for now.

\* **@param** year - the year of the appointment to remove

\* **@param** month - the month of the appointment to remove

\* **@param** day - the day of the appointment to remove

\*/

**public** **void** remove(**int** year, **int** month, **int** day) {

//this method needs to iterate over your list of appointments

//and remove elements who's occursOn() method return true

//when passed the parameters above.

}

/\*\*

\* Method to return a string representation of this Calendar object.

\* Overrides the Object method toString (see page 448 in text).

\* (also see page 453 Special Topic 9.6)

\* **@return** a String representation of the Calendar object.

\*/

**public** String toString() {

String ret = "";

//this method needs to iterate over your list of appointments

//and construct the return string

//make sure to put each appointment on its own line

//by using “\n”

**return** ret;

}

Note that you also need to create a toString method for your Appointment class. (Your subclasses will inherit this version of the method).

Make sure to add accessors to your Appointment class for getYear, getMonth, getDay, and getDescription.

Here is some sample output from a working project using the AppointmentDemo.java.txt file posted on UTC Learn. You will need to create additional tests to prove that your project works correctly.

Before removal of appointment

Daily[Brush your teeth. Date: 8/13/2000]

Monthly[Visit grandma. Date: 5/20/2003]

Onetime[Dentist appointment. Date: 11/2/2004]

Onetime[Trick or Treat. Date: 10/31/2004]

Monthly[Dentist appointment. Date: 11/2/2004]

Onetime[Dentist appointment. Date: 11/2/2004]

After removal of 11/2/2004

Monthly[Visit grandma. Date: 5/20/2003]

Onetime[Trick or Treat. Date: 10/31/2004]

***IMPORTANT!!*** To make it easier to test/grade your program, make your method/class names to be the same. This will allows to make a single set of test code which can be run on everyone's methods. Method names and class names are detailed in this document above.

When you are finished with your project, take a screen-shots or text capture of your final output.

## To Turn In via UTC Learn

You should turn in 1 .ZIP file containing your java files and a PDF document with screen-shots (or text) of your output. 1 file should be uploaded to UTC Learn. ***IMPORTANT!!!*** You should name your file in the following manner. lastname-firstname-lab04.zip. So John Smith would submit smith-john-lab04.zip.