MANCHESTER METROPOLITAN UNIVERSITY
SCHOOL OF COMPUTING, MATHEMATICS AND DIGITAL TECHNOLOGY

ASSIGNMENT COVER SHEET

COURSE: Advanced Programming

UNIT: 6G5Z1001

LECTURER: Dr Alan Crispin

ASSIGNMENT NUMBER: 1 **Appointment System**

ASSIGNMENT TYPE:(GROUP/INDIVIDUAL)

ISSUE DATE: 16th November 2015

HAND-IN DATE: 5th February 2016

HAND-BACK DATE:

No responsibility is accepted by the School if an assignment is lost. To cover this eventuality, you are advised to take a copy of your assignment or to ensure you have the means of re-creating it.

PLAGIARISM: Students are reminded that plagiarism (copying) is a serious disciplinary matter. Checks are regularly made for misuse of the web and other existing materials. See current Regulations for Taught Postgraduate Programmes of Study

PROCEDURE FOR HANDING IN WORK: see Faculty Student Handbook. Follow any specific instructions given on the assignment specification

PENALTIES FOR LATE HAND-IN: see Regulations for Undergraduate Programmes of Study

EXCEPTIONAL FACTORS AFFECTING YOUR PERFORMANCE: see current Regulations for Taught Postgraduate Programmes of Study

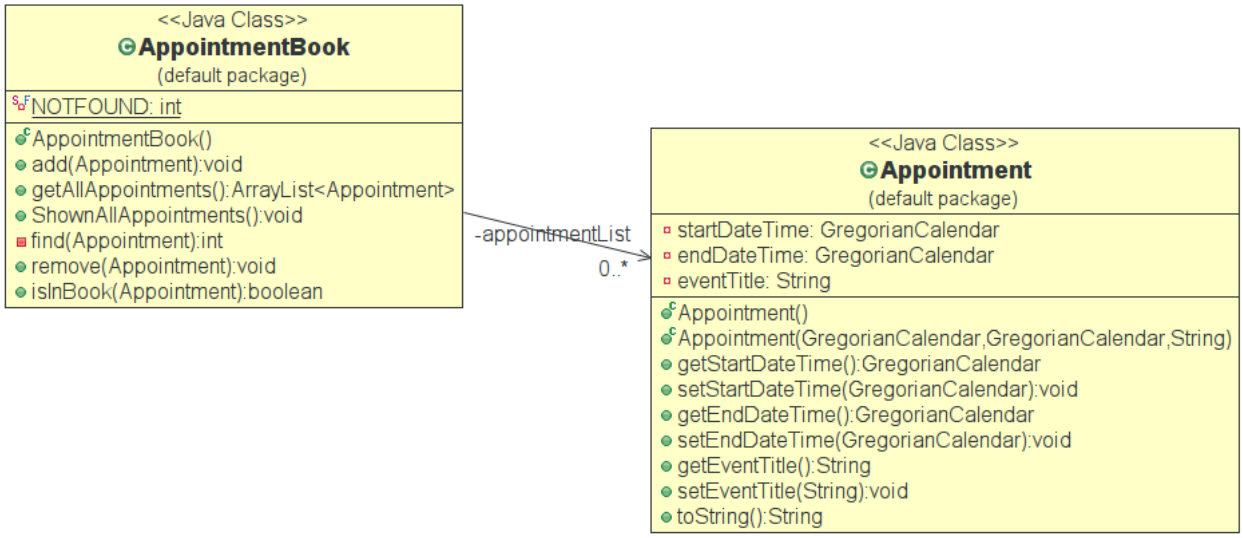
ASSESSMENT CRITERIA: see attached assignment specification

Appointment System

The aim of this assignment is to develop an appointment booking system with a graphical user interface front end. The aim is to create a calendar application which stores appointments allowing a user to add, delete, search and check for appointments. You should not use a visual editor to create the GUI but code it directly. The following provides a step by step guide to the assignment. You may, if you wish, tackle it in your own way.

Step 1: Base Classes

The assignment requires that a Swing graphical user interface is developed on top of a set of base classes called AppointmentBook and Appointment as defined in the UML diagram shown below.



The AppointmentBook class has an ArrayList which can store 0 or many (0..*) appointments. Each appointment has a start date and time, an end date and time and an event title. When developing the base classes, a Controller class should be used as a test harness and have a main() method. So, for example, you could test the above classes with the following code.

```
Appointment a1 = new Appointment(new GregorianCalendar(2015, 8+1, 14, 10,30), new
GregorianCalendar(2015, 10, 14,11,30), "Dentist");

Appointment a2 = new Appointment(new GregorianCalendar(2015, 8+1, 20,9,00), new
GregorianCalendar(2015, 10, 20,10,10), "OOWP Lecture");

Appointment a3 = new Appointment(new GregorianCalendar(2015, 8+1, 21,14,00), new
GregorianCalendar(2015, 10, 21,16,00), "Tutorial");

AppointmentBook appBook = new AppointmentBook();
appBook.add(a1);
appBook.add(a2);
appBook.add(a3);
appBook.ShowAllAppointments();
appBook.remove(a1);
appBook.ShowAllAppointments();

System.out.println("Find Test: "+ appBook.isInBook(a2));
System.out.println("Find Test: "+ appBook.isInBook(a1));
```

Notice that the Appointment class makes use of the GregorianCalendar available in the package java.util. Use the Java API documentation to find out how to use this class and how to use its methods. You should ensure that an appointment that conflicts with a previously entered appointment is displayed as a conflict so that the user can take appropriate action. All classes developed must be fully documented using Javadoc.

Step 2: Creating a Graphical User Interface

To create a Swing graphical user interface you will need a MainForm class which extends JFrame and contains an AppointmentBook object as shown in the code example below. In this example the AppointmentBook object is called “appBook”.

```
import javax.swing.*;

public class MainForm extends JFrame
{
  AppointmentBook  appBook = new AppointmentBook();

public MainForm()
  {
    super("Appointment System Assignment");
  }
}
```

To invoke the graphical user interface you need to develop a controller class which has a main method for creating and displaying a MainForm object.

```
public class ControllerGUI
{
/**
          * @author Alan Crispin
          */
public static void main(String[] args){
//initiate the creation of the GUI on the event dispatching thread
 javax.swing.SwingUtilities.invokeLater(new Runnable() {
       public void run() {
       MainForm f = new MainForm();
       f.setSize(600,400);
       f.setVisible(true);
       }
       });
  }
}
```

Now design the main form so that it allows appointments to be added, deleted, searched, checked and displayed. Add buttons for navigation to allow a user to cycle through appointments both forwards and backwards individually. You should design your user interface so that it is functional and user friendly. It should provide a means of displaying daily and monthly appointments.

Step 3. Saving and Opening Appointments

The next step is to develop menu items to save and open (load) appointment data stored in the ArrayList. Develop methods to save appointment data from the ArrayList as

1) plain text (.txt)
2) comma separated value (.csv) format
3) iCalendar format (.ics).

Use file chooser components when loading and saving data.

Step 4. Advanced Features

There are many ways in which the core appointment book system can be expanded as seen with commercial products. For example, the system could send email and SMS text message reminders

for critical appointments, automatically synchronise with the Google Calendar API, use a database in place of an ArrayList and incorporate a web front (hint: research Java Servlets). Once you have the core elements developed, research and code an advanced feature such as one of those suggested above.

Hand In:

You are required to up-load your Java source files in zip format to Moodle and a short two page report containing a screen shot of your graphical user interface, the console testing undertaken with respect to your base classes and your advanced feature. Upload your assignment with the format

surname_studentnumber.zip

You may be asked to demonstrate your work to your laboratory tutor.

Assessment Criteria

Mark Scheme

In what follows, you must satisfy all of the criteria for one classification before attempting to satisfy the next highest (that is, work upwards from the pass grade in terms of which features you provide). The position within a classification band will be determined by the overall quality of the code supplied, commenting and presentation.

Pass mark (40-49%) The appointment base classes (e.g. AppointmentBook and Appointment) must be written and tested at the console using the Controller class as a test harness. You must demonstrate that you have checked for appointment conflicts and exceptions that may arise. Your code must compile and execute. Your code should be commented using Javadoc.

Lower Second (50-59%) In addition to the requirements in the previous stage your system must incorporate a graphical user interface which allows appointments to be added, deleted, searched and displayed. Your graphical user interface should incorporate navigation buttons (or other mechanism) to allow a user to cycle through appointments, both forwards and backwards, individually. It should also allow a user to display daily and monthly appointments.

Upper Second (60-69%) In addition to the requirements in the previous stages your system must incorporate menu items to allow appointment data to be saved to disk as either a text file (.txt) or comma separated values (.csv).

First (70% plus) In addition to the requirements in the previous stages you should save appointments in the iCalendar format. This is a computer file format which allows internet users to send meeting requests and tasks to other internet users, via email, or sharing files with an extension of .ics. You must make an attempt at an advance feature.

Distinctive (80%+) Everything for a First, plus an additional feature such as the use of a database instead of an ArrayList or the incorporation of a web front end. The additional feature will be assessed on both its complexity and usefulness. You will be rewarded also for any other features which are not specified above but which sensibly improve the software and the use of JUnit testing of the base classes.

Dr Alan Crispin (16/9/15)