Version Number:
1.10 (Feb. 20, 2020)

Assignment 2

Room Booking Scheduler, Part 2

To be submitted online not later than Monday, March 2nd, 2020, 11:59 p.m.

Description:

In this lab you'll continue to add new features to the room booking scheduler. Along the way, you'll demonstrate your understanding of the following Course Learning Requirements (CLRs), as stated in the *CST8284—Object Oriented Programming (Java)* course outline:

1. Write java programming code to solve a problem, based on the problem context, including UML diagrams, using object-oriented techniques (CLR II)
2. Use basic data structures (CLR III), including implementing arrays of primitive and reference types
3. Implement inheritance and polymorphism (CLR IV)
4. Use ArrayLists to manage objects (CLR V)
5. Implement program Input/Output operations by storing objects to a file using serialization (CLR VII)
6. Produce code that has been tested and executes reliably (CLR VIII)
7. Debug program problems using manual methods and computerized tools in an appropriate manner. (CLR X)
8. Identify appropriate strategies for solving a problem (CLR XI)

Worth
5.0%
of your total mark

Assignment 2

Room Booking Scheduler, Part 2

Program Description

In this assignment you'll add additional features to the Room Booking Scheduler you started in Assignment 1. For those students who failed to submit that assignment, or for those whose submission was defective in whole or part, you may use my copy of Assignment 1, posted on Brightspace, as the starting point for your Assignment 2 submission. (Even if you're satisfied with your Assignment 1 code, you might want to look over the sample code anyway, to see how you might have implemented things differently.) Whatever your reasons, you may use the Assignment 1 code in whole or in part, *with a proper citation*, in building your Assignment 2.

I. Load the Assignment2 project and copy your existing classes to the new Project

a) Download the CST8284_Assignment2.zip file from Brightspace, and import the zip file into Eclipse, just as you have with your labs. Copy the six classes from Assignment 1 into the `cst8284.asgmt2.roomScheduler` package. Refactor each class's `package` statement to correspond to the new package name, if required.

b) The UML diagram for this assignment has been redacted to reflect the changes in Assignment 2. This may affect some of the declarations for methods you created in Assignment 1, which will need to be refactored for this assignment. So check the UML in this document carefully for changes in the declarations, access modifiers, use of static, etc.

c) Before making the additions specified in Section II, note that the same rules apply to this assignment as the last, briefly stated as:

1. Follow the UML diagram *exactly as it is written*, according to the most up-to-date version of this document. As before, you cannot add members that are not written in the UML diagram, nor can you ignore any of them either, whether they are used in code or not.
2. Most of the new methods indicated in the UML diagram are intended to be used (again, with exception of some getters and setters). Take the UML as your guide, not just of what needs to be written, but of how the pieces are connected to one another.
3. Employ best practices at all times, especially code reuse.

II. Add the following new features to your Room Scheduler

a) Replace every array with an ArrayList

Remove the `roomBookings` array and replace it with an ArrayList of type `RoomBooking` (this requirement extents to any other arrays you may have created in Assignment 1). Then change all code based on arrays and replace it with appropriate ArrayList methods instead.

In particular, remove `lastBookingIndex`, along with its getters and setters; the last used index is readily available using ArrayList's `size()` method, hence we don't need to keep track of this location anymore. Correct any other methods that were impacted by the conversion from an array to an ArrayList.

In `findBooking()`, if you haven't already done so, use an *enhanced for* loop, to search through the ArrayList of `RoomBookings`. (Note that, as always, the new ArrayList should not be referenced directly; use `getRoomBookings()` to return its reference value.

b) Add new methods to RoomScheduler

Start by adding a `deleteBooking()` method that allows the user to remove a `Booking` based on its `Calendar` date and time. (See sample output below for details.) Note that you should use existing methods for this purpose as much as possible, along with any ArrayList methods you find appropriate, rather than rewriting the same code over again. If you're practicing good code reuse, this method can be

implemented in less than a dozen lines of code using the existing `findBooking()` and `makeCalendarFromUserInput()` methods.

Similarly, add a `changeBooking()` method that allows the user to change the date and/or time of an existing `RoomBooking`. Obtain a `Calendar` object from the user, use `findBooking()` to locate and return the existing `RoomBooking` (if it exists), and then modify its current `TimeBlock` to the new date and time. Again, this should take about a dozen lines of code using the methods already available.

It should be noted that while it is possible to combine the above two methods into one method—they require much of the same information, and only differ in the last step—the improvements in code and efficiency are minimal, and this is more than compensated for by the lack of clarity that results. So while code reuse is generally the rule of the day, this should be avoided in situations where the end result produces code that is more, rather than less, confusing. This is one such case.

Finally, you'll need to add 3 additional methods to the menu, `displayRoomInfo()`, `saveBookingsToFile()` and `loadBookingssFromFile()`. See parts (c) and (d) below for details of their operation.

These new features will needed to be added as new options to the menu: see the UML for the order in which the menu should be displayed. (Note that if you implemented the fixed constants correctly in Assignment 1, adding new features should not result in any major rewrites to your existing menu or code, just a renumbering of the named (fixed) constants, a new output String in for each new item in the menu, and a new `case` in the `switch` statement to call the new methods, one for each new feature added. But the original code should not need editing—*if* you implemented your code correctly in the first Assignment. See the Instructor's Assignment 1 code for an indication of how to use these constants correctly.

c) Extend the Room class as indicated.

The `cst8284.asgmt2.room` package contains an abstract class that holds `Room` information (which was missing from Assignment 1). Each `Room` has, at very minimum, one property: a room number, saved as a String (so that rooms may have labels like "A154" and "CA405"), along with concrete getters and setters as well.

Additionally, a `toString()` method is provided, one that outputs standard information about the room. But to ensure that `toString()` displays appropriate information, three of its methods, `getRoomType()`, `getSeats()` and `getDetails()`, are all declared abstract in the `Room` superclass. Thus you must override these methods in each of its subclasses to return information appropriate to the subclass.

As indicated in the UML, we've assumed that the `Room` class has three subclasses: a `Classroom`, a `ComputerLab`, and a `Boardroom`. The values returned by each subclass's concrete methods are indicated below:

`Classroom`

`seats`	DEFAULT_SEATS
`roomType`	"class room"
`details`	"contains overhead projector"

`ComputerLab`

`seats`	DEFAULT_SEATS
`roomType`	"computer lab"
`details`	"contains outlets for 30 laptops"

`Boardroom`

`seats`	16
`roomType`	"board room"
`details`	"conference call enabled"

First, you'll need to implement the three subclasses so their concrete getters return the above information. Next, modify the `RoomScheduler` constructor to take a `Room` object, as indicated in the UML. (Note that you must import the `Room` package into any class that implements a `Room` object or any of its subclasses.) Next, add the `room` field to `RoomScheduler` (see UML), add its getters and

setters, and then use the latter to save the new Room object (or more accurately, its reference value) passed to the Constructor using the setter. Then add a line to the first case statement in executeMenu(), call displayRoomInfo(), and output the appropriate subclass information using toString() when the first menu item is selected.

Finally, change the RoomSchedulerLauncher to load a new ComputerLab object, instantiated with the room number "B119", into the RoomScheduler.

Note: (1) you should not need to alter the contents of the Room class in any way; (2) All subclasses of Room should be made final; this prevents extending the concrete subclasses with additional concrete methods (such as setters, which could be used to override the default values). So making the subclasses final prevents potential abuse. You'll discover other ways to protect your code in the Level III, Design Patterns (CST8288) course; (3) As well, there are better ways to implement the displayMenu() method than we've indicated thus far. Specifically, enumeration types (or 'enums') should be used. However, this topic is not covered until CST8288, so for now, we'll need to be content with using the fixed constants indicated. (4) In an actual application, the Room information would be returned from a database, rather than 'hard-coded' into the application via RoomSchedulerLauncher, as we've done here. You'll learn how to do this in your Level IV courses. For now, we'll simulate that part of the project by instantiating a new Room subclasses.

d) Add File IO to the existing code

Using the hybrid lectures and notes as your guide, add file I/O functionality to your application. You'll need to create two new methods in RoomScheduler, one to save the RoomBooking ArrayList to a file, the other to retrieve this information. Details for both methods are provided below.

saveBookingsToFile()

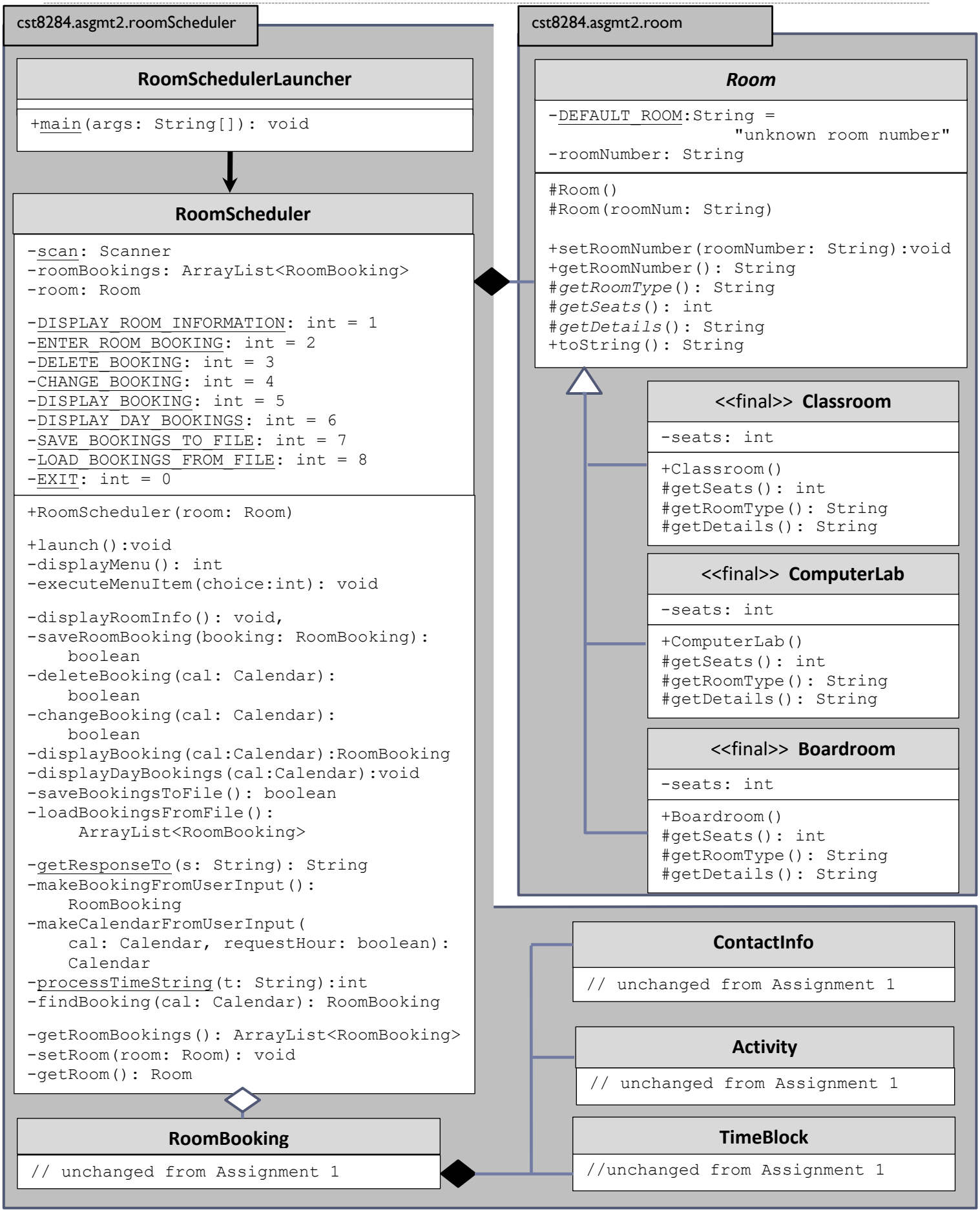
In this method, load each RoomBooking object in the ArrayList into a file named CurrentRoomBookings.book. DO NOT hardcode the file location to a particular subdirectory, as your code will not work correctly when it is transferred to another platform. Also, do not prompt the user for a filename; the CurrentRoomBookings.book file is to be used internally by your program. Simply use the default directory associated with your project (which is typically the src folder, or one of its subdirectories) to store the file.

loadBookingsFromFile()

This method performs the opposite operation, loading the file contents into the RoomBookings ArrayList. As indicated in the hybrid videos, you should use EOFException to terminate loading the ArrayList.

Students often encounter problems loading and unloading files the first time they write FileIO code. The most common cause is that the file you think you are loading from/to does not actually exist. Check carefully (using debug, of course) to ensure that there's actually a file where you think it is, otherwise expect NullPointerExceptions and other mysterious errors. Remember also that when reading objects from a file, you must cast each Object returned from the file to its appropriate class, which in this case will be RoomBooking.

Another common problem: students frequently assume that because their program works fine on their laptop, it will work fine everywhere. But when your lab instructor runs your program, there's no guarantee that the same CurrentRoomBookings.book file will be available on their laptop. If this happens, you've lost marks, because a major component of the assignment will not execute at all. So test your code thoroughly. In particular, stress test your code by deleting any existing CurrentRoomBookings.book files to ensure that your program still works correctly at startup without the default file present.


cst8284.asgmt2.roomScheduler
RoomSchedulerLauncher
+main(args: String[]): void
RoomScheduler
-scan: Scanner
-roomBookings: ArrayList<RoomBooking>
-room: Room
-DISPLAY_ROOM_INFORMATION: int = 1
-ENTER_ROOM_BOOKING: int = 2
-DELETE_BOOKING: int = 3
-CHANGE_BOOKING: int = 4
-DISPLAY_BOOKING: int = 5
-DISPLAY_DAY_BOOKINGS: int = 6
-SAVE_BOOKINGS_TO_FILE: int = 7
-LOAD_BOOKINGS_FROM_FILE: int = 8
-EXIT: int = 0
+RoomScheduler(room: Room)
+launch():void
-displayMenu(): int
-executeMenuItem(choice:int): void
-displayRoomInfo(): void,
-saveRoomBooking(booking: RoomBooking): boolean
-deleteBooking(cal: Calendar): boolean
-changeBooking(cal: Calendar): boolean
-displayBooking(cal:Calendar):RoomBooking
-displayDayBookings(cal:Calendar):void
-saveBookingsToFile(): boolean
-loadBookingsFromFile(): ArrayList<RoomBooking>
-getResponseTo(s: String): String
-makeBookingFromUserInput(): RoomBooking
-makeCalendarFromUserInput(cal: Calendar, requestHour: boolean): Calendar
-processTimeString(t: String):int
-findBooking(cal: Calendar): RoomBooking
-getRoomBookings(): ArrayList<RoomBooking>
-setRoom(room: Room): void
-getRoom(): Room
RoomBooking
// unchanged from Assignment 1
cst8284.asgmt2.room
Room
-DEFAULT_ROOM:String = "unknown room number"
-roomNumber: String
#Room()
#Room(roomNum: String)
+setRoomNumber(roomNumber: String):void
+getRoomNumber(): String
#getRoomType(): String
#getSeats(): int
#getDetails(): String
+toString(): String
<<final>> Classroom
-seats: int
+Classroom()
#getSeats(): int
#getRoomType(): String
#getDetails(): String
<<final>> ComputerLab
-seats: int
+ComputerLab()
#getSeats(): int
#getRoomType(): String
#getDetails(): String
<<final>> Boardroom
-seats: int
+Boardroom()
#getSeats(): int
#getRoomType(): String
#getDetails(): String
ContactInfo
// unchanged from Assignment 1
Activity
// unchanged from Assignment 1
TimeBlock
//unchanged from Assignment 1

Another potential complication is *serialization*—to be discussed in class shortly. To ensure this doesn't cause you problems, add the following line to the `RoomBooking` class:

```
public static final long
            serialVersionUID = 1L;
```

While the convention is to spell a final identifier in ALL_CAPS, in this case you must use the 'camel case' format indicated above, because that is how the JVM expects it. Also, to be on the safe side, you should also add this line to any of the component classes that make up RoomBooking, including Room itself.

Last but not least: your code should automatically save the RoomBooking ArrayList, not just when the user selects the appropriate item from the menu, but whenever the program shuts down. Similarly, when `RoomScheduler` loads, if a `CurrentRoomBookings.book` file is available, it should be used to initialize the `RoomBooking` ArrayList.

III. Notes, Suggestions, Tips, and Warnings

a. As with Assignment 1, before requesting assistance, you should set breakpoints in your code at the 'last known good' location and step forward from there in debug until the error is encountered. Fix, and repeat as required. And then, if you're truly stuck, contact the instructor.

b. As before, each class must include, at the top, the following information:

```
/*  Course Name:
    Student Name:
    Class name:
    Date:
*/
```

c. Students are reminded that:
   - You should not need to use concepts that lie outside of the ideas presented in the course.
   - You *must* cite all sources used in the production of you code according to the information provided in Module00. Failure to do so *will* result in a charge of plagiarism. The one exception is the information in the course notes themselves
   - Students must be able to explain the execution of their code. If you can't explain its execution, then it is reasonable to question whether you actually wrote the code or not. Partial marks, including a mark of zero and a charge of plagiarism, may be awarded if a student is unable to explain the execution of code that he/she presumably authored.

d. The instructor's version of the code will be released at midnight, March 5th, for those students who did not complete this lab on time, or who wish to build their Assignment 3 code on top of the instructor's version (if their own effort was unstable or incomplete.) Note however, that once the 'official' version is released, it essentially nullifies any late submissions.

e. Sample data is shown at the end of this document. *Your code must be able to run using this data as written*; marks will be removed if it does not.

IV. Submission Guidelines

Your code should be uploaded to Brightspace (via the link posted) in a single zip file obtained by:

1) In Eclipse, selecting the ***project*** name (`CST8284_20W_Assignment2`)

2) right clicking on 'Export' and selecting General/Archive File; click Next;

3) in the Archive File menu make sure *all* of the project subfolders are selected (src, bin, .settings) and the 'Save in zip format' and 'Create in Directory Structure' radio buttons are selected

4) In the 'To Archive File' window, save your zip file to a location you'll remember. But make certain the name of your zip file corresponds to the following format, as outlined in Module 00:

Assignment2_Yourlastname_Yourfirstname.zip

including the underscores and capitals, but with *your* last and first name inserted as indicated. Failure to label your zip file correctly will result in lost marks.

Corrections and Addenda:

Version 1.10 (Feb. 20/2020)

Page 3: `loadBookingsFromFile()` has been corrected to indicate that objects read from the file must be cast to a `RoomBooking` object, rather than a `Room` object.
Page 4: added missing Room subclass constructors to UML diagram
Room class: output should say "30 seats;", not just "30;"
Sample output and Marking Guide added below

Sample Output

```
Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
1
B310 is a computer lab with 30 seats; contains outlets for 30 laptops

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
2

Enter Client Name (as FirstName LastName): Kim Wong
Phone Number (e.g. 613-555-1212): 613-727-4723
Organization (optional): Algonquin College
Enter event category: CST8110 lab
Enter detailed description of event:
Event Date (entered as DDMMYYYY): 10102020
Start Time: 9:00
End Time: 11:00
Booking time and date saved.

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
2

Enter Client Name (as FirstName LastName): Jay Patel
Phone Number (e.g. 613-555-1212): 613-727-4723
Organization (optional): Algonquin College
Enter event category: CST8288
Enter detailed description of event:
Event Date (entered as DDMMYYYY): 10102020
Start Time: 3 pm
End Time: 5 pm
Booking time and date saved.

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
6

Event Date (entered as DDMMYYYY): 10102020
No booking scheduled between 8:00 and 9:00
---------------
9:00 - 11:00
Event: CST8110 lab
Description:
Contact Information: Kim Wong
Phone: 613-727-4723
Algonquin College
---------------
No booking scheduled between 11:00 and 12:00
No booking scheduled between 12:00 and 13:00
No booking scheduled between 13:00 and 14:00
No booking scheduled between 14:00 and 15:00
---------------
15:00 - 17:00
Event: CST8288
Description:
Contact Information: Jay Patel
Phone: 613-727-4723
Algonquin College
---------------
No booking scheduled between 17:00 and 18:00
No booking scheduled between 18:00 and 19:00
No booking scheduled between 19:00 and 20:00
No booking scheduled between 20:00 and 21:00
No booking scheduled between 21:00 and 22:00
No booking scheduled between 22:00 and 23:00
No booking scheduled between 23:00 and 24:00

Enter a selection from the following menu:
1. Display room information
```

```
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
4

Enter booking to change
Event Date (entered as DDMMYYYY): 10102020
Start Time: 3 pm
---------------
15:00 - 17:00
Event: CST8288
Description:
Contact Information: Jay Patel
Phone: 613-727-4723
Algonquin College
---------------
Enter New Start Time: 16:00
Enter New End Time: 18:00
Booking has been changed to:
---------------
16:00 - 18:00
Event: CST8288
Description:
Contact Information: Jay Patel
Phone: 613-727-4723
Algonquin College
---------------

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
7

Current room bookings backed up to file

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
3

Enter booking to delete
Event Date (entered as DDMMYYYY): 10102020
Start Time: 9 am
---------------
9:00 - 11:00
Event: CST8110 lab
Description:
Contact Information: Kim Wong
Phone: 613-727-4723
Algonquin College
---------------
Press 'Y' to confirm deletion, any other key to
abort: Y
Booking deleted

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
6

Event Date (entered as DDMMYYYY): 10102020
No booking scheduled between 8:00 and 9:00
No booking scheduled between 9:00 and 10:00
No booking scheduled between 10:00 and 11:00
No booking scheduled between 11:00 and 12:00
No booking scheduled between 12:00 and 13:00
No booking scheduled between 13:00 and 14:00
No booking scheduled between 14:00 and 15:00
No booking scheduled between 15:00 and 16:00
---------------
16:00 - 18:00
Event: CST8288
Description:
Contact Information: Jay Patel
Phone: 613-727-4723
Algonquin College
---------------
No booking scheduled between 18:00 and 19:00
No booking scheduled between 19:00 and 20:00
No booking scheduled between 20:00 and 21:00
No booking scheduled between 21:00 and 22:00
No booking scheduled between 22:00 and 23:00
No booking scheduled between 23:00 and 24:00

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
8

Current room bookings loaded from file

Enter a selection from the following menu:
```

```
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
6

Event Date (entered as DDMMYYYY): 10102020
No booking scheduled between 8:00 and 9:00
---------------
9:00 - 11:00
Event: CST8110 lab
Description:
Contact Information: Kim Wong
Phone: 613-727-4723
Algonquin College
---------------
No booking scheduled between 11:00 and 12:00
No booking scheduled between 12:00 and 13:00
No booking scheduled between 13:00 and 14:00
No booking scheduled between 14:00 and 15:00
No booking scheduled between 15:00 and 16:00
---------------
16:00 - 18:00
Event: CST8288
Description:
Contact Information: Jay Patel
Phone: 613-727-4723
Algonquin College
---------------
No booking scheduled between 18:00 and 19:00
No booking scheduled between 19:00 and 20:00
No booking scheduled between 20:00 and 21:00
No booking scheduled between 21:00 and 22:00
No booking scheduled between 22:00 and 23:00
No booking scheduled between 23:00 and 24:00

Enter a selection from the following menu:
1. Display room information
2. Enter a room booking
3. Remove a room booking
4. Change a room booking
5. Display a booking
6. Display room bookings for the whole day
7. Backup current bookings to file
8. Load current bookings from file
0. Exit program
0

Exiting Room Booking Application
```

Assignment 2 Marking Guide

Requirement	Mark
The submitted zip file is correctly labelled, and contains all project-related files in the package indicated, along with all expected classes, as outlined in the Assignment 2 document in Section IV, Submission Guidelines.	**/2**
Code loads, compiles and executes in Eclipse, with no 'red dots' in the left hand column in Eclipse and no errors generated during execution. Note that failure to submit an executable program may impact your marks in the remainder of this assessment, since most of the marking depends upon being able to test your code by its execution. So even if the underlying code is correct, if it can't be run, it can't be marked—you'll get 0.	**/3**
Replaced arrays with ArrayLists, along with all supporting methods and fields	**/3**
Program executes correctly, as demonstrated by having output consistent with the sample output shown in the latest version of the Assignment 2 document. Note that your code must be able to display the output shown *in its entirety*, with no 'Scanner bugs'.	**/6**
New package added, containing the abstract `Room` class provided, along with the three subclasses specified in the UML diagram. Your code should be able to load any of these subclasses in the `RoomSchedulerLauncher`, and store them polymorphically in the `Room` field in `RoomScheduler`, allowing `Room`'s `toString()` method to be used to output information. The `Room` information should be correctly displayed by selecting the first item from the menu.	**/6**
`deleteBooking()` and `changeBooking()` work as required	**/4**
`saveBookingsToFile()` and `loadBookingsFromFile()` save and load information as required, including loading and saving automatically on startup and shutdown, respectively.	**/6**
MINUS: late penalty; failure to cite sources, or missing documentation; private information not kept secure through data hiding; diagnostic strings output to the console, abnormal termination, exceptions thrown under certain circumstances; unusual, abnormal and erratic features displayed during execution—simply put, your program is the code equivalent of Donald Trump's presidency—etc.	
Total:	**/30**