Object-Oriented and Event-driven Programming (Using Java)
Instructor: Sharma Chakravarthy
Project I

Made available on:	**1/16/2014**
Due on:	**1/30/2014 (11:55Pm)**
Submit by:	**Blackboard (1 zipped folder containing all the files/sub-folders) https://elearn.uta.edu/**
Weight:	**5% of total**
Total Points:	**100**

Problem Statement:

The goal of this project is to get you started with writing a working Java program and doing some non-trivial computation using basic Java language constructs. In this project, you will not be using object-oriented concepts, but will use the structured programming that you learnt in CSE 1320. You will get familiar with writing a java program by implementing a DateTime class with several methods and constructors.

What you are asked to do:

1. Implement a number of constructors as specified in this document.

2. Implement a number of methods as specified in this document.

3. Write a main program that extensively tests and outputs results from the DateTime class. Use examples of testing from the Date and Time classes provided.

You should check your code for different input errors and handle them with appropriate output messages. For example, you should not construct/return a DateTime object that is not valid!

The output should be very clear and easy to check for correctness and understand.

Since you are familiar with C, think about how you would do it in C. We will then compare that with how it is done in Java to understand, to some extent, the advantages of Java. We will spend some time on this in the class.

Java provides mechanisms for comments and input for Javadoc. Please make sure your code contains both. Also, submit the output of Javadoc generated for your program. See examples in the classes provided to you.

In addition, you will report a few (3 to 5) important logical (not syntactic) errors that you encountered while doing this project. Furthermore, you will briefly provide an explanation on how you debugged and fixed each of these errors.

The following constructors and methods should be implemented in the file DateTime.java (provided) for this project:

1. `//constructor; constructs a DateTime object with today's date and time (implemented)`

```
public DateTime(){ … }
```

2. `//constructor; takes a string argument of the form`
   `// 01-16-2014,15:33:25:89`

```
public DateTime(String givenDateTime){ … }
```

3. `//constructor; takes 7 integer args and constructs a DateTime object with those values`

```
public DateTime(int month, int day, int year, int hr, int
            min, int sec, int hundredth) { … }
```

4. `//methods to add components of date and time`

```
public DateTime addDays(int d) { … }
public DateTime addMonths(int m) { … }
public DateTime addYears(int y) { … }
public DateTime addHours(int h) { … }
public DateTime addMinutes(int m) { … }
public DateTime addSeconds(int s) { … }
```

5. `//compare two DateTimes and return +ve, 0, -ve if`
   `//dt1 > dt2, dt1 = dt2, and dt1 < dt2`

```
public int compareTo(dt2){ … }
```

6. `//compare two DateTimes for before, after`

```
public boolean isAfter(DateTime dt2){ … }
public boolean isBefore(DateTime dt2) { … }
```

7. `Override the public void toString method to print a DateTime object legibly.`

```
public String toString() { … }
```

Optionally,

8. //Add and subtract two DateTimes returning a DateTime

```
Public DateTime addDateTime(DateTime d2) { … }
Public DateTime subtractDateTime(DateTime dt2) { … }
```

9. //return a formatted string for a DateTime object

```
public String toString() { … }
```

Coding Style:

Be sure to observe the following standard Java naming conventions and style. These will be used across all projects for this course; hence it is necessary that you understand and follow them correctly. You can look this up on the web. Remember the following:

a. Class names begin with an upper-case letter, as do any subsequent words in the class name.
b. Method names begin with a lower-case letter, and any subsequent words in the method name begin with an upper-case letter.
c. Class, instance and local variables begin with a lower-case letter, and any subsequent words in the name of that variable begin with an upper-case letter.
d. All user prompts must be clear and understandable
e. Give meaningful names for classes, methods, and variables even if they seem to be long. The point is that the names should be easy to understand for a new person looking at your code
f. Your program is properly indented to make it understandable. Proper matching of if … then … else and other control structures is important and should be easily understandable
g. Do not put multiple statements in a single line

In addition, ensure that your code is properly documented in terms of comments and other forms of documentation for generating meaningful javadoc.

Grading Scheme:

The project will be graded using the following scheme:

1. Correctness of the algorithms, test case completeness, and output clarity : 70
2. Reporting and fixing 3 to 5 logical errors in the program: 10
3. Documentation (Javadoc) and commenting of the code: 10
4. Following the coding style listed above: 10

What and How to Submit:

For this project, you will submit the following files (in one zip file) using the naming convention shown below:

1. Each java file will contain one or more related classes
2. A text file listing 3 logical (or semantic) errors and the process used to debug and fix each of them. Include the total number of hours spent on the project as well. The file should be named as: 'proj1_errors_firstname_lastname.txt'
3. All the javadoc and html files generated by the javadoc command should be in a folder named javadoc where the .java files are. Use the command `javadoc –d javadoc <your_file_name>.java` to create java documentation.
4. A script file as described in a separate document using the test cases that will be supplied for submission (see how to generate a script file on Omega)

All of the above files should be placed in a single zipped folder named as - 'proj1_firstname_lastname_Section_final'.

Only one zipped folder should be uploaded using blackboard. Look for more information on submission on the course bb site.

Your java program must be executable (without any modification by us) from the command prompt on UTA's Omega server. So, please test it on Omega before submitting it. However, the source code files can be created and/or edited on any editor that produces an ASCII text file. As I mentioned in the class, an IDE is not necessary for this and subsequent projects. If you decide to use it, please learn it on your own and make sure your code compiles and executes on Omega without need for any modification after submission (you may have to remove a few things the IDE may insert by default).

If your program does not compile or execute using the class names given (i.e., without any modifications from us), it will be returned without grading.