**CSE 1325 – Spring 2014**

**Object-Oriented and Event-driven Programming in Java**

**Lab 2**

Design Document Due Date: **Thursday, March 6, 2013 at 11:59pm**

 **(one minute before midnight 12am)**

Lab Due Date: **Thursday, March 13, 2013 at at 11:59pm**

(see instructions on website for how to turn this in - “lab submission info”)

Grade value: 8% out of 100% for all grades

**Objective:**

The theme of this project is to design classes and methods for managing a community theater (plays, directors, actors, attendees) using object-oriented design (OOD) principles. It is expected that this project will form the basis for the rest of the projects in the semester. Please make sure you adhere to OO principles throughout the design and implementation. The projects are designed to give you an opportunity to exercise the principles taught in the course. Remember, Java is only a vehicle for supporting OOD. Make sure you fully leverage the features available in Java for your benefit as long as we’ve touched on them in class. Ask before jumping into totally new concepts for your lab.

In this project, you will be designing the data abstractions and required operations on those abstractions for managing an community theater.

Be sure to check the DEDUCTIONS section at the end of this assignment to avoid penalties

You are also required to design your program in advance before you begin writing code. You will document your design and turn in the design document at least a week before the lab assignment is due. The goal of the design document is to assist you in developing the actual program.

**Description:**

Design classes, fields (attributes) and methods for the problem description below. Make sure you understand and differentiate between public and private ( and optionally protected) fields and methods. Use static fields/methods where appropriate. Make sure you have at least one constructor (more as needed) with arguments. Each class should have appropriate get (accessor) and set (mutator) methods. Each class should have a toString method (and other formatting methods as needed).

Use a “test” class to run the application (to test/call each method defined in classes). The test class should print the following menu, accept user input and execute the menu to provide understandable output. All inputs (as appropriate for the menu) should be accepted from the common window and should prompt for the input with appropriate messages. If bounds are appropriate, please indicate it in the prompt and check for input errors. Test your source code—be sure every method works correctly.

Test class main routine should check all of the following. This means that each element below is considered a test case for your classes:

1) Sell tickets to attendees

2) Rent items for the play

3) Hire an actor or director

4) Pay an actor/director

5) Remove (fire) an actor /director

6) Display all actor/director details

7) Display all attendee details (including name, date to attend, seats)

8) Display all play details (show dates and days, total seats, number of box seats)

0) Exit program

A. The community theater class should contain the following data and methods:

Data members:

1. Has a name

2. Has a current play showing on multiple dates and days

3. Has different prices for different show days

3.a. Friday and Saturday night price

3.b. Saturday and Sunday matinee price

3.c. Sunday night price

4. Has a number of seats per show

 4.a. Regular seat cost (based on show day price)

 4.b. Box seat ticket cost (a percentage over the show day price)

5. Has a maximum yearly operating budget

6. Keep track of rented items for show (using a collection class)

Methods:

7. Sells play tickets to attendees

8. Accepts payments from attendees

9. Keeps track of how many tickets have been sold

10. Refunds ticket payments

11. Keeps track of sales of tickets as a monetary total

12. Hires actors and directors

13. Fires actor/directors

14. Keeps track of how many actors/directors are currently working on a given play

15. Pays actors/directors at the end of each week, adjusting the operating budget as necessary

16. Displays all plays, actors, attendees, etc. in a tabular formatted manner

B. Each play has the following features

EXAMPLE

1. Title Cabaret

2. Author Jay Allen

3. Year published 1972

4. Opening date March 31, 2014

5. World premier boolean

6. Play usage rental $8400.00

7. End date April 15, 2014

(Use Java Date class. E.g., - http://tinyurl.com/yhuwodk)

8. Total number of shows 11

 8.a. Friday night shows

 8.b. Saturday matinee shows

 8.c. Saturday night shows

 8.d. Sunday matinee shows

 8.e. Sunday night shows

9. List of actors

10. Director

C. Each person has the following attributes EXAMPLE

1. First name Jaya

2. Last name Smith

3. Identification number 006

4. Address 123 Sesame St., NY, NY 10001

Each actor has the following attributes EXAMPLE

1. First name Jaya

2. Last name Smith

3. Identification number 006

4. Address 123 Sesame St., NY, NY 10001

5. Hired for which play Cabaret

6. Pay scale per hour $12.00

7. Actor's Equity? boolean

8. Role Character name

9. Hours worked per week 15

Each director has the following attributes EXAMPLE

1. First name Shin

2. Last name Igwe

3. Identification number 013

4. Address 456 Sesame St., NY, NY 10001

5. Hired for which play Cabaret

6. Pay scale per hour $30.00

7. Actor's Equity? boolean

8. Up front fee $450.00

9. Percentage of house 15%

Each attendee has the following attributes EXAMPLE

1. First name Maria

2. Last name Gruen

3. Identification number 445

4. Address 987 Sesame St., NY, NY 10001

5. Attending which play Cabaret

6. Ticket price category $20.00

7.Number of tickets integer

8. Date of tickets March 19, 2014

9. Which show? (Eve or matinee) (use an enum here)

**Program Requirements**

For Lab #2 you should

1. define classes for all the described elements above,
2. use inheritance to create at least three of the classes,
3. create at least one additional class to handle money transactions,
4. create a ticket class,
5. keep data members private unless otherwise instructed
6. create get and set methods for all private data members
7. create toString and/or display methods for all classes (other than the test class)
8. create constructors and initialize objects as needed

BONUS:

Java provides mechanisms for comments and input for Javadoc. Please make sure your code contains both. Also, submit the files generated by the Javadoc for your program. Look up the convention for specifying parameters (@param) and user information for javadoc. Every method should have its meaningful description for use by someone else. Use the relevant Javadoc tags specified in -http://en.wikipedia.org/wiki/Javadoc or <http://java.sun.com/j2se/javadoc/writingdoccomments/>

**Standard Java conventions and coding style:**

Be sure to observe standard Java naming conventions and style. These will be observed across all projects for this course, hence it is necessary that you understand and follow them correctly. They include:

* Class names begin with an upper-case letter, as do any subsequent words in the class name.
* Method names begin with a lower-case letter, and any subsequent words in the method name begin with an upper-case letter. This is called "camelCase".
* 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.
* Each set of opening/closing curly braces must be in the same column.
* All user prompts must be clear and understandable
* 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 person looking at your code
* All static final variables will be upper case identifiers (i.e.constants)

In addition, ensure that your code is properly documented in terms of comments and other forms of documentation wherever necessary.

**What and How to Submit**

For Lab #2 WRITE A DESIGN DOCUMENT FIRST. The design must include

 a) all the classes you expect to write,

 b) brief (one line) descriptions of each data member of the class

 c) brief (one line) descriptions of each method (member function) of the class, and

 d) some indication of the relationships between the classes.

The design document may be written as lines of text or as diagrams (such as a diagram that start with the main function at the top and all others below it, a tree, UML, etc.) or as some combination of those, but it must include the information listed for a), b), c), and d) above. Each class should encapsulate one thing and each piece of data at a lower level should have a more specific purpose than the class that owns it. Be sure to include all the classes that are described in this lab assignment. See the website or this assignment for the DESIGN DOCUMENT due date. It is usually ONE WEEK PRIOR to the lab due date.

A Lab #2 Design Document must be turned in in order for your Lab #2 assignment to be graded.

For this lab project itself, please submit the following material using the Blackboard system at elearn.uta.edu. Please submit ONE .zip file that contains:

* Each .java file used in development with the appropriate name for the included class
* A document, called your "test plan", listing each test case that is in your test class and what the purpose is of each test
* A development notes file (standard text or word file) that describes at least three logical errors that you found while developing your code, how you solved it, and the total amount of time you spent on this project in terms of hours.

The .zip file containing your project must have the name format of: **proj2\_1000number\_firstname\_lastname**

Upload this code to Blackboard at elearn.uta.edu

**A Few Helpful Tips:**

* Netbeans by default recompiles your class every time you save your file. If you want to force everything to be recompiled, press the ‘clean and build’ button.
* Ctrl + space in most IDE’s will show a list of available members (or variables) and methods (or functions).

**Grading Rubric:**

**Code: (50%)**

 Class header and method headers for all classes (4 points)

 Comments (line comments and block comments) (3 points)

 Style (following Java conventions, indentation, consistency, meaningful identifiers, lateral separation of code from line comments, etc.) (3 points)

 Modularity (division of the problem into small parts) (5 points)

 Correct declaration of the required classes with data and methods and any other needed classes

 Community theater (4 points)

 Play (4 points)

 Actor (4 points)

 Attendee (4 points)

 Other classes (10 points)

 Test plan includes at least two tests for each of the required

 functions of sell tickets, hire, pay, fire, etc. and uses the

 display functions to verify test results (8 points)

**Output: (50%)**

 User clearly understands what is being requested for input (3 points)

 Required tasks performs correctly

 (sell tickets, hire, pay, fire, display (3), exit, and others) (15 points)

 Set functions handle inputs correctly including error checking (6 points)

 Get functions provide values safely (4 points)

 Output gives clear information to explain the values to the user (3 points)

 Output contains all the given test data (if any) and at least two additional data sets, i.e at least two other actors, two add’l attendees, one additional director, and one additional play (12 points)

 Output demonstrates all elements of test plan (7 points)

**Grading Deductions:**

 Labs which do not compile in NetBeans will receive an overall grade of 0 (zero) [-100 deduction]

 Labs which have errors in them and do not terminate normally in NetBeans will receive an overall grade of 0 (zero) [-100 deduction]

 Late submission of softcopy of design document, code and/or other documents to Blackboard will result in an overall grade of 0 (zero) UNLESS student has obtained prior instructor approval [-100 deduction]

 Use of Java language elements not yet discussed in class by the lab due date will result in potential deduction of points - discuss with instructor before using.

 Use of Java public variables (C equivalent to global variables) will result in an overall grade of 0 (zero) [-100 deduction]

 Use of unstructured control elements, like the break (except in the case of switch statements) or continue keywords, will result in an overall grade of 0 (zero) [-100 deduction]

Miscellaneous: If you have questions, e-mail Dr. T (tiernan@uta.edu) and the TA.