Concordia University
Comp 249 - Winter 2018
Object-Oriented Programming II
Assignment 2 – Due 11:59 PM - Wednesday Feb.21, 2018

Purpose: The purpose of this assignment is to practice class inheritance, and other Object Oriented Programming concepts such as constructors, access rights, method overriding, etc. The assignment would also allow you to practice the notion of package creation. This assignment contains two parts. You need to complete part I to be able to do part II.

You have been hired by the metropolitan events planning office to write a software application that helps the office manage the year around events.

Part I

Various events can be described as follows:

- A **Event** class with the following: the *year* (XXXX: int type), *month* (YY:int type), and the *number of cities* where it will be held (int type).
- A **Festival** is an **Event** that in addition has the following: a *name* (String type- such as *Arts, Beer, Comedy, Film, Fire, Folk…etc*), *ticket price* (double type), and *duration:# of_days* (int type).
- A **Culturalfiesta** is a **Festival** that in addition has the following: *the number of spoken languages* (int type), which indicates the maximum languages spoken in this kind of festival.
- A **Musicfiesta** is a **Festival** that in addition it has the following: *number of bands* (int type).
- A **SportCompetition** is an **Event** that in addition has the following: *number of activitie* (int type) and *season name* (enumeration type that can be: *summer, fall, winter, spring*)
- A **fair** is an **Event** that in addition has the following: The number of exibitors (int type), and *type* (enumerator type that can be : career, science, trade, travel)

Part I:

1. Draw a UML representation for the above mentioned classes hierarchy. Your representation must also be accurate in terms of UML representation of the different entities and the relation between them. You must use a software to draw your UML diagrams (no hand-writing/drawing is allowed).

2. Write the implementation of the above mentioned classes using inheritance and according to the specifications and requirements given below:

   - You must have 4 different Java packages for the classes. The first package will include the **Event** class. The second package will include the **Festival**, **Culturalfiesta,** and **Musicfiesta** classes. The third package will include the **SportCompetition** class and the last package will include the **Fair** class.

- For each of the classes, you must have at least three constructors, a default constructor, a parametrized constructor (which will accept enough parameters to initialize ALL the attributes of the created object from this class) and a copy constructor. For instance the parametrized constructor of the ***Culturalfiesta*** class must accept 7 parameters to initialize the *year, month, #number_ of_cities, name, ticket price, duration, and #of languages.*

- An object creation using the default constructor must trigger the default constructor of its ancestor classes, while creation using parametrized constructors must trigger the parametrized constructors of the ancestors.

- For each of the classes, you must include at least the following methods: accessors, mutators, ***toString()*** and ***equals()*** methods (notice that you are always overriding the last two methods).

  - The ***toString()*** method must display clear description and information of the object (i.e *"This arts Culturalfiesta will be held in 2018, 07 in 3 cities, for 7 days, the ticket will cost 39.75$, and has 2 spoken language").*

  - The ***equals()*** method must first check if the passed object (to compare to) is null and if it is of a different type than the calling object. The method would clearly return false if any of these conditions is true; otherwise the comparison of the attributes are conducted to see if the two objects are actually equal. You need to add a comment indicating how effective these null verifications inside the method will be in relation to protecting your program from crashing!

- For all classes, with the exception of the **Festival** class, you are required to use the appropriate packages access rights. For the **Festival** class for example, you are required to use protected access rights.

- When accessing attributes from a base class, you must take full advantage of the permitted rights. For instance, if you can directly access an attribute by name from a base class, then you must do so instead of using a public method from that base class to access the attribute.

3. Write a driver program (that contains the main() method) that would utilize all of your classes. The driver class can be in a separate package or in any of the already existing four packages. In the main() method you must:
   - Create various objects from the 6 classes, and display all their information using the ***toString()*** method.
   - Test the equality of some to the created objects using the ***equals()*** method.
   - Create an array of 10 **Event** objects and fill that array with various objects from the 6 classes (each class must have at least one entry in that array).
   - Trace(search) that array to find the object that:
     (a) Has the least/most number of cities
     (b) Are happening on the same year (e.g. 2018)

     For both (a) and (b) display all information of the objects along with their location (index) in the array.

Part II

For the requirements of this part, you need to modify your implementation from Part I as follows:

1. The classes from part I, must now have the most restrictive (secure/protective) access rights to their attributes. Adjust your implementation from Part I accordingly, and comment on the decision about using restricted access (i.e. what are the tradeoffs).

2. In the driver program of this part, you need to add to the one from part I, another static method (add it to the above `main()` method), called ***`copyFestival()`***. The method will take as imput an array of **Event** (an array of any size) and returns an array of **Event**. That is to say, the method needs to create an array of the same length as the passed paramter one, copy all Festival from the passed array to a new array then return it. Your copy of the objects must use the copy constructors of the different listed classes.

3. In the driver program, create an array of 12 objects (must have at least one from each of the classes), then call the *`copyFestival()`* method to create a copy of the that array.

4. Display the contents of both arrays, then add some comments indicating whether or not the coping is correct. If not; you need to explain why it has not been sucessful or as you might expected.

General Guidelines When Writing Programs

- Include the following comments at the top of your source code (each .java file)
  - `// ----------------------------------------------------------`
  - `// Assignment# (include number)`
  - `// Question: (include question/part number, if applicable)`
  - `// Written by: (include student name and id if assignment done by one student or both students if assignment done by two students`
  - `// ----------------------------------------------------------`
- In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations might get a bit longer.
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy to read format.
- End your program with a closing message so that the user knows that the program has terminated.

Assignment#2 Submission

- For this assignment, you are allowed to work individually, or in a group of **2 students maximum.** You and your teammate must however be in the same section of the course.
- If working in a group, **only one** of the two members submit/upload the assignment.
- Only electronic submissions will be accepted. Zip together the source code files.
- Students will have to submit their assignment (one copy per group) using the Moodle/EAS system (please check for your section submission). Assignments must be submitted in the right DropBox/folder of the assignments. **Assignments uploaded to an incorrect DropBox/folder will not be marked and result in a zero mark. No resubmissions will be allowed. Check your section assignment method, that is you should upload your assignment to EAS (**https://fis.encs.concordia.ca/eas/**) or to Moodle (depending on your section).**
- **Naming convention for zip file:** Create one zip file, containing all source files and produced documentations for your assignment using the following naming convention:
  - The zip file should be called *a#_StudentName_StudentID*, where # is the number of the assignment and *StudentName/StudentID* is your name and ID number respectively. Use your

"official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. For example, for the first assignment, student 12345678 would submit a zip file named like: *a2_Mike-Simon_123456.zip*. If working in a group, the name should look like: *a2_Mike-Simon_12345678-AND-Linda-Jackson_98765432.zip.*

- Submit only ONE version of an assignment. If more than one version is submitted the first one will be graded and all others will be disregarded.
- **Finally, notice that a demo is required for the assignment. Failure to do your demo, or missing you reserved demo time, will result in a zero mark for the assignment regardless of your submission.** ***There will be no substitution for a missed demo time.*** **Please see course outline for further details.**

Evaluation Criteria

Part 1 (7 points)	
UML representation of class hierarchy	2 pt
Proper use of packages	1 pt
Correct implementation of the classes	1 pt
Constructors	1 pt
toString() & equals()	1 pt
Driver program & general correctness of code	1 pt
Part 2 (3 points)	
Access rights	1 pt
copyFestival() and its behaviour	2 pt
Total	**10 pts**