Sympony Project - *CST8288*

School of Advanced Technology *Computer Programmer*

Purpose

Create a project team of (max) 5 members. You will remain in your team for both Project 1 and Project 2. Your team will work together to produce the symphony system prototype (Domain Layer Model with code). Time constraints within the course may prevent developing a complete prototype; however, you will be able to develop the overall domain layer framework. Development of this prototype will give you an opportunity to apply the concepts discussed in lecture and further apply your object oriented programming and design skills.

Part I (project 1) focuses on the development of the UML diagram and implementation of most domain objects and test classes for the prototype. Part II (project 2) will focus on the application of a DAO (data access object) design pattern, server side components with Java servlets, and presentation layer components using HTML and/or JSP. A framework will be provided along with code samples to reduce the time required for project 2.

Program Description

Your task for the first project (project 1) is to take the following system description with the provided ERD and develop the UML diagram and domain object implementation (with test classes). You will need to investigate this type of system further in order to uncover additional features/functionality for you prototype. Your domain should reside in the package named symphony.domain and your test classes in the package symphony.test. Following will be a brief overview of the system and some indication of requirements; however, **as with most systems in the beginning, the specifications are incomplete at best**. You will have to conduct your own research/analysis to discover many of the requirements of such a system and thus, the complete domain layer for the application. I urge you to use your professor as the "domain expert" and the web as a research tool for systems that may have similar functionality required by the "Symphony Management System"

You domain layer implementation must include a minimum of the following…

- Design patterns:
  - Builder
  - Delegate
  - Singleton

- Refactoring to base abstract classes if appropriate
- Appropriate inheritance hierarchies
- Classes and/or interfaces where appropriate
- Controller class/interface to bridge the view layer with the domain layer and "run" (execute, manage) the business processes of the system (controller must implement the "add concert" business process and provide "null" methods for the remaining business processes of the system)

Background Notes

After completing a course in OOP and Java, you have been asked to develop a prototype OO system representing the "object view" of the existing symphony database (provided later). In preparation you have conducted an interview with the "domain expert" for the symphony orchestra. Following is a summary of the key domain abstractions (entities) discovered by your interview.

CONCERT SEASON

Concerts are arranged each season. A season has an opening date along with an indicator of the length (in days) of the concert season. A concert season consists of a number of concerts. Concerts may have different conductors, but only one conductor for any single concert.

CONCERT

A concert is a given performance of one or more compositions. Each concert has been given an identity by the committee (combination of characters and numbers). Concerts are assigned a date upon which the concert is performed. Each concert typically has more than one performance occurring on different dates.

COMPOSITION

A composition is a musical score or piece performed by the symphony at each concert. When selected by the committee and conductor, each composition is given an identification (can be a combination of characters and numbers). Compositions are created by a composer. Each compositions may contain movements. Each movement is given a number and name by the composer. Many, but not all compositions have multiple movements.

CONDUCTOR

The conductor is the person who will conduct a specific performance of the concert. Conductors are assigned a number when hired. The concert committee also records the full name, address, and telephone number of the conductor.

SOLOIST

A soloist is an artist who performs a given composition at a particular concert. Each soloist hired by the committee is given an identification number (can be characters or numbers). The committee also records the soloists area expertise (e.g. sax player, flutist...), their full name, address, and phone number.

During further discussions you discover the following...

1. A concert season schedules one or more concerts. A particular concert is scheduled for only one concert season.

2. A concert includes the performance of one or more compositions. A composition may be performed at one or more concerts, or may not be performed.

3. For each concert there is one conductor. A conductor may conduct any number of concerts, or may not conduct any concerts.

4. Each composition may require one or more soloists, or may not require a soloist. A soloist may perform one or more compositions at a given concert, or may not perform any composition. The symphony orchestra wishes to record the date when a soloist last performed a given composition.

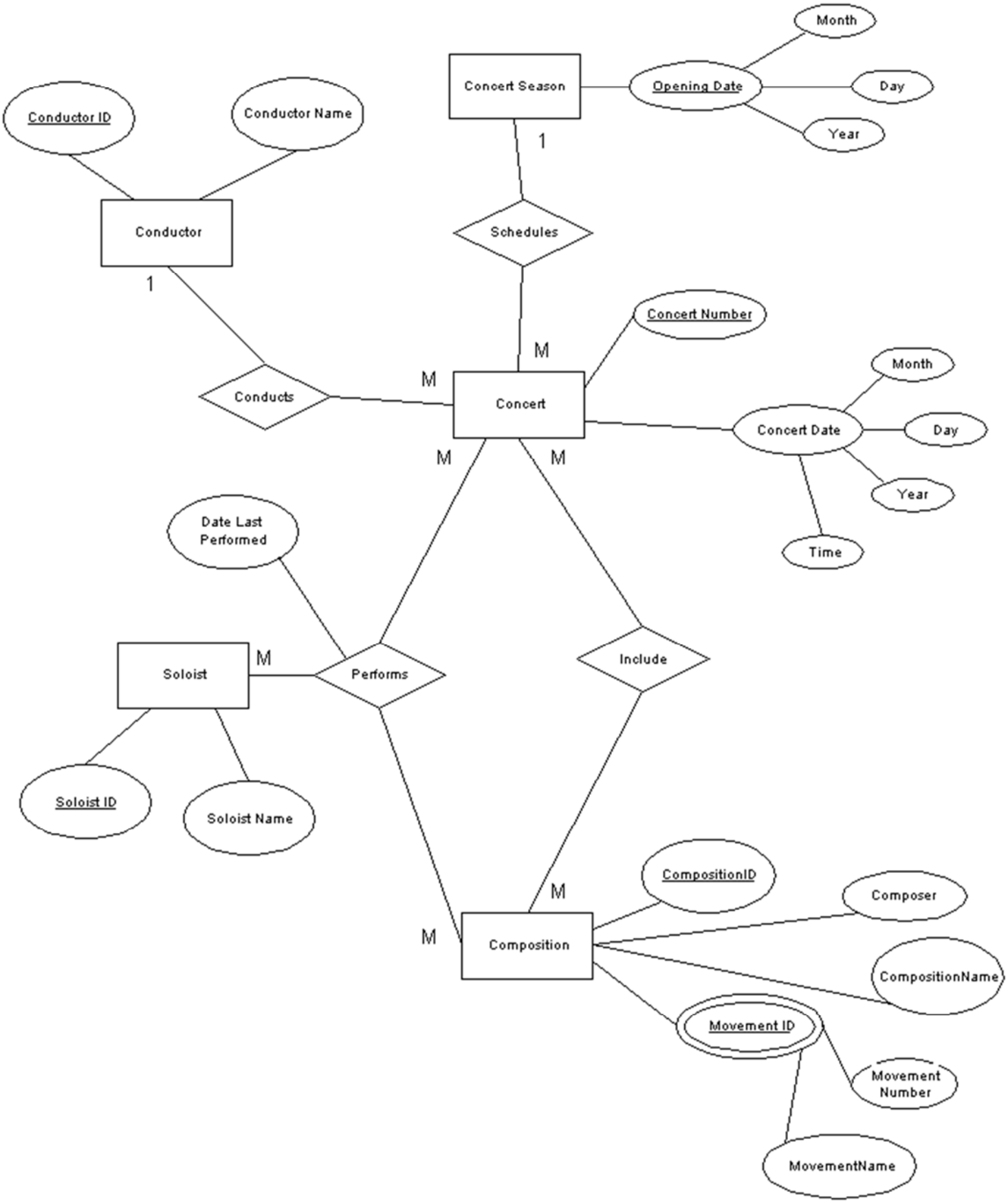
Note:

More information may be discovered in discussions with the “domain expert” and application of your investigative skills as applied to the internet for symphony management systems.

Based on the above information the following (intial) ERD was constructed....

Entity Relationship Diagram (not normalized and not necessarily complete, however complete for the information from above)

Symphony Orchestra ERD



Program Documentation

You code must adhere to the published Java standards discussed in class and provide the necessary JavaDocs for you application along with appropriate test classes. There is no requirement to provide any kind of "view" layer or "data management" layer.

Submission Requirements

Submit your java classes (for the classes on your UML diagram) including your test classes in a jar file with the following naming structure: (further requirements are posted on blackboard under the description of the assignment)

CST8288-W16Project1_<name1>-<name2>BIN.jar

Your source code in a jar file with the following name:

CST8288-W16Project1_<name1>-<name2>_SRC.jar

Your documentation (javadoc's) and UML diagram (as a jpeg) image, in a jar file with the following name:

CST8288-W16Project1_<name1>-<name2>_DOC.jar