# Department of Computing

# Sarmad Shabbir

# BESE-5A

**SE312: Software Construction**

**Class: BESE – 5 AB**

# Lab 8: Gradebook

**Date: April 19th, 2017**

**Time: Wednesday (10:00 – 13:00), Wednesday (14:00 – 17:00)**

# Instructor: Fahad Ahmed Satti

# 

# Lab 8: Gradebook

## Introduction

In this lab, you will create a Gradebook using Spring and Hibernate. A skeleton code with persistent classes and DAOs has been provided for easy start-up. Using the skeleton code design your console based application to be able to manage grades, courses, students and teachers.

## Objectives

After performing this lab students will be able to understand:

* Spring
* Hibernate

## Tools/Software Requirement

* Solutions should be made using Java and must use the ORM Hibernate.
* You can take help from internet but remember **no plagiarism.**

**Description**

A gradebook is used to keep track of grade achieved by a student, in course contents. As a result when modeling a gradebook, courses, teachers, students, course learning outcomes and the achieved marks, against some obtained marks are all necessary. The skeleton code provided to you, uses annotations to provide persistent classes for all of these entities. Also provided for your convenience are the Data Access Objects (DAOs) for these entities which will allow you to store an entity or retrieve it.

Start by running the skeleton code by setting up a configuration file for hibernate and spring using pom.xml for building your application with Maven.

Using Spring you then have to upgrade your application and define BO (Business Objects), provide configurations, update executioners and ensure a single transactional system is being used.

Ensure your application has proper separation of concerns, with Executioner only interfacing with the BOs, which have all the important functions defined. The BOs can only talk to DAOs, which will perform the operations on your data. For exchanging data between your Executioner, BOs and DAOs you can use the defined Models.

Define the application-context.xml with basic spring and hibernate settings(avoid using hibernate.cfg.xml) and define beans for injecting only the BOs(to be injected in the executioner) and DAO(to be injected in the BOs) beans.

Next update your application completely to allow a user to add or update courses, teachers, students, contents, clos and grades. The executioner should also allow the user to either print or generate a CSV file for exporting the grades of a student, a course or course content.

**Lab Task**

Your lab task will be comprised of the following 2 parts:

1. Execute the current system.
2. Define BOs to allow addition or updating of courses, teachers, students, contents, CLOs and Grades.
3. Store all the available grades from your current courses, along with other associated information, available via LMS or CMS.
4. Allow the user to export the data by printing it on the screen or generating a CSV file for it.
5. Use only HCQL for retrieval operations.
6. The user should be able to export all the information of a graded item.
7. The user can select graded items for a course, a student, or course content.
8. Define proper separation of concerns by separating the Executioner (View), BOs, DAOs and using only Models for exchanging data between these layers.
9. Unit tests to evaluate your code.
10. Using a Version Control System (VCS) to manage your solutions.

## Deliverables

* Each submission is individual with the following composition:
  + Source Code
  + Unit Tests
  + Documentation(Introduction, Approach, Design and How to Run the application)
  + Link to the public repo on GitHub
* Convert your submission files into a zip folder and name it as given below, finally upload the zip folder to LMS.
  + Name – Registration No. – Section

## Grade Criteria

This lab will be graded on the following rubric, with minimum marks 0 and maximum marks of 24:



Documentation:

A gradebook is used to keep track of grade achieved by a student, in course contents. As a result when modeling a gradebook, courses, teachers, students, course learning outcomes and the achieved marks, against some obtained marks are all necessary. The skeleton code provided to you, uses annotations to provide persistent classes for all of these entities. Also provided for your convenience are the Data Access Objects (DAOs) for these entities which will allow you to store an entity or retrieve it.

Start by running the skeleton code by setting up a configuration file for hibernate and spring using pom.xml for building your application with Maven.

Using Spring you then have to upgrade your application and define BO (Business Objects), provide configurations, update executioners and ensure a single transactional system is being used.

Ensure your application has proper separation of concerns, with Executioner only interfacing with the BOs, which have all the important functions defined. The BOs can only talk to DAOs, which will perform the operations on your data. For exchanging data between your Executioner, BOs and DAOs you can use the defined Models.

Define the application-context.xml with basic spring and hibernate settings(avoid using hibernate.cfg.xml) and define beans for injecting only the BOs(to be injected in the executioner) and DAO(to be injected in the BOs) beans.

Next update your application completely to allow a user to add or update courses, teachers, students, contents, clos and grades. The executioner should also allow the user to either print or generate a CSV file for exporting the grades of a student, a course or course content.

The **Spring Framework** is an [application framework](https://en.wikipedia.org/wiki/Application_framework) and [inversion of control](https://en.wikipedia.org/wiki/Inversion_of_control) [container](https://en.wikipedia.org/wiki/Servlet_container) for the [Java platform](https://en.wikipedia.org/wiki/Java_platform). The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the [Java EE](https://en.wikipedia.org/wiki/Java_EE) platform.

1. Execute the current system.
2. Define BOs to allow addition or updating of courses, teachers, students, contents, CLOs and Grades.
3. Store all the available grades from your current courses, along with other associated information, available via LMS or CMS.
4. Allow the user to export the data by printing it on the screen or generating a CSV file for it.
5. Use only HCQL for retrieval operations.
6. The user should be able to export all the information of a graded item.
7. The user can select graded items for a course, a student, or course content.
8. Define proper separation of concerns by separating the Executioner (View), BOs, DAOs and using only Models for exchanging data between these layers.
9. Unit tests to evaluate your code.
10. Using a Version Control System (VCS) to manage your solutions.

**Github link: https://github.com/sarmadhi/software-construction-labs**