The Due Team

HW6:Persistence and code1

Yishuo Wang
Walt Wu
Zeqing Li
Yimin Zhu

1. Persistence

We use a persistence framework Hibernate(http://hibernate.org/orm/) to handle our object/relational mapping in our project.

Hibernate Configuration file hibernate.cfg.xml is located at directory ./src/ and Annotation for JavaBean Entity is written in Java classes in this project.
Static singleton SessionManager.java located at directory ./src/db/ to create org.hibernate.SessionFactory that database transaction can be done by opening a session using sessionFactory handle.

1.1. The Hibernate configuration file

hibernate.cfg.xml file defines the Hibernate configuration information. The hibernate.cfg.xml is located under the ./src package.

The <connection.driver_class>, <connection.url>, <connection.username> and <connection.password> property elements define JDBC connection information.

The <dialect> property specifies the particular SQL variant with which Hibernate will converse. In this case we use value “org.hibernate.dialect.MySQLDialect” to indicate that we are mapping Java objects to MySQL database.

The <hbm2ddl.auto> property enables automatic generation of database schemas directly into the database. We set this value to “create-drop” to indicate that we will create new database schema each time and if there exists an database schema it will be drop automatically.

The <mapping/> element at the very end naming the annotated entity class using the class attribute.

There are many ways and options to bootstrap a Hibernate SessionFactory. For additional details, see the Native Bootstrapping topical guide.

1.2. The annotated entity Java class

The entity class in this project follows JavaBean conventions. In fact the class itself is identical to the one in The entity Java class, except that annotations are used to provide the metadata, rather than a separate mapping file.

Example 1. Identifying the class as an entity

```
@Entity
```

```
@Table( name = "Administrator" )
public class Administrator{
   ...
}
```

The @javax.persistence.Entity annotation is used to mark a class as an entity. It functions the same as the <class/> mapping element discussed in The mapping file. Additionally the @javax.persistence.Table annotation explicitly specifies the table name. Without this specification, the default table name would be ADMINISTRATOR.

Example 2. Identifying the identifier property

```
/* ./src/core/user/UserType */
@Id
@Column(name = “netId”     )
protected String netId
```

@javax.persistence.Id marks the property which defines the entity’s identifier, which represents the primary key in table.

Example 3. Identifying basic properties

```
/* ./src/core/event/Appointment */
@Temporal(TemporalType.TIMESTAMP)
@Column(name = "START_DATE")
@Basic(optional = false)
private Date startDateTime;
```

The date property needs special handling to account for its special naming and its SQL type. @javax.persistence.Temporal annotation is used for mapping MySQL data type TIMESTAMP.

Attributes of an entity are considered persistent by default when mapping with annotations, which is why we don’t see any mapping information associated with title.

2. Code Convention

In this project, our team follow the Google Java Style (https://google.github.io/styleguide/javaguide.html).

3. Implementation Status Report

Edit Testing Center Information	Use case satisfied	4%

Import Data	Use case satisfied	3%
Display Utilization	Use case partially satisfied	3%
View Appointments	Use case satisfied	3%
Cancel/edit Appointments	Use case satisfied	3%
Check-in a Student	Use case satisfied	3%
User Interface (Structure)	Finished	5%
User Manual	Partially finished	1%

Estimated percentage of progress: 25%.

4. User Manual

To make sure the application can run correctly on the server side, the user needs to download and place several .jar files into the lib directory.

The following libraries are needed for this project:

- Hibernate 4.2.2
- apache log4j 1.2.17
- gson 2.4
- MySQL Connector J
- Spring MVC 4.1.6

The user is expected to download the libraries and place the .jar files in the lib directory.

5. Test Report

Name of test case: exam

“exam” is a use case for the instructor's requests that have been approved by the administrator

Brief description, indicating the requirements, features, or use cases being tested.

Pre-condition, if any. The test case should execute successfully from any state satisfying the pre-condition. For example, a typical pre-condition is that a user with a specified role is logged in, or that the database contains specified records.

Flow of Events: a numbered list of steps, describing the inputs to the system and the expected actions by the system.

Outcome: whether the system passed or failed the test; in case of failure, give a brief description of the system's observed behavior, and explain the source of the problem, if possible.

Test Case 1.1 Import Data

Description	Read .csv files including user class and rosters
Precondition	1. Login as the Administrator 2. Appropriate name for .csv file
Flow of Event	1. System prompts user for term and folder name 2. User specifies a term and name of the folder 3. System specifies the type of information and stores the data
Outcome:	---------- importData ----------" | -user/class/roster information: ; Then system display the specific information it saves

Test Case 1.2 Add exam

Description	“exam” is an use case for the instructor's requests that have been approved by the administrator. It might be “Course” type or “Ad Hoc” type
Precondition	1. Connecting Database 2. Instructor sends exam request to the testing center
Flow of Event	1. System prompts user to input information of the exam for course. 2. Users provides the information of the exam, including start date, end date, start time, end time, exam name, exam Id, type of the exam(Course or Ad hoc), Duration, etc. 3. Request succeed and System saves and updates the newest information.
Outcome:	---------- add Exam info----------" | -Exam Id: " + exam.getExamId()); | -Exam Name: " + exam.getExamName()); | -Type: " + exam.getType()); | -Start Date Time: " + exam.getStartDateTime()); | -End Date Time" + exam.getEndDateTime()); | -Duration: " + exam.getDuration());

Test Case 1.3 View exam

Description	View all the exam that Testing center has, and their information.

Precondition	1. connecting database 2. have access to view 3. press the view exam button
Flow of Event	1. User press view exam button 2. System gets the instruction and read exam Table and output all the exam one by one with the format that we set. 3. UI shows out all the exam information
Outcome:	---------- view Exam1 info----------" | -Exam Id: " + exam1.getExamId()); | -Exam Name: " + exam1.getExamName()); | -Type: " + exam1.getType()); | -Start Date Time: " + exam1.getStartDateTime()); | -End Date Time" + exam1.getEndDateTime()); | -Duration: " + exam1.getDuration()); ---------- view Exam2 info----------" | -Exam Id: " + exam1.getExamId()); | -Exam Name: " + exam2.getExamName()); | -Type: " + exam2.getType()); | -Start Date Time: " + exam2.getStartDateTime()); | -End Date Time" + exam2.getEndDateTime()); | -Duration: " + exam2.getDuration()); . . .

Test Case 1.4 View Actual Utilization

Description	The actual utilization for the testing center
Precondition	Testing center has have approved exam request Student has sent appointments to do the exam in the exam range press view actual utilization button
Flow of Event	1. user press view actual utilization button to tell system we need this 2. System get the instruction and get the corresponding data to count the actual utilization 3. System output the actual utilization 4. connecting database

Outcome:	---------- View Actual Utilization----------" | -Actual Utilization-: " + util.countUtilzActual()+"%") ---------- View Actual Utilization----------"

Test Case 1.5 View Expected Utilization

Description	The Expected utilization for the testing center
Precondition	1. Testing center has have approved exam request 2. Student has sent appointments to do the exam in the exam range 3. We have set the "day" of the testing center, and the gap between exams. 4. press view expected utilization button 5. connecting database
Flow of Event	1. user press view expected utilization button to tell system we need this 2. System get the instruction and get the corresponding data to count the expected utilization 3. System output the expected utilization
Outcome:	---------- View Expected Utilization----------" | Expected Utilization-: " + util.countUtilzExpection()+"%") ---------- View Expected Utilization----------"

Test Case 1.6 View Appointments

Description	Administrator view the all appointments that students send to testing center, and they are separated by the each exam(approved request)
Precondition	1. connecting database 2. Instructors have had the approved request, and students have had the appointment to do the test. 3. administrator press the view appointments
Flow of Event	1. Administrator enter a specific date 2. System displays all appointments on that date 3. Administrator selects "modify" or "cancel"
Outcome:	All the information of the appointments on the specific date

Test Case 1.7 Cancel Appointments

Description	administrator cancel an appointment
Precondition	1. connecting database 2. Instructors have had the approved request 3. choose the specific appointment and press cancel button

Flow of Event	1. Administrator presses “cancel” 2. System displays checkbox for selection 3. Administrator selects appointments to be deleted and presses cancel 4. System deletes selected appointments
Outcome:	A message sends from system to show us we have deleted selected appointments.

Test Case 1.8 Check-in a student for an appointment

Description	Students go to take exam and swipe card to show that they come to test.
Precondition	1. connecting database 2. Students have made appointment for the test 3. Students go to attend test
Flow of Event	1. student go to test 2. student swipe the student ID card or show instructor he comes 3. System get the information that student comes and update the data
Outcome:	System shows mout that a student comes to attend test

6. Contribution

Walt Wu:

1. User interface structure design and implementation
2. Partial functionality realization
3. Active discussion on the project.

Yishuo Wang:

1. Implement most functionality coding part
2. Implement some of the test coding part
3. Writing all the test report for each testing
4. Active discussion for the whole project

Yimin Zhu:

1. Implement functionality and testing coding part for data collection and testing center info
2. Write test cases for data collection and testing center info
3. Active discussion for the whole project

Zeqing Li:

1. Implement Hibernate configuration and setup remote MySQL server.
2. Implement Functionality for Administrator including:
    a. View appointments at a specific time

    b. Cancel an appointment
    c. Edit an appointment
3. Write test cases for Administrator
4. Active discussion for the whole project
5. Write Persistence Layer Documentation
6. Configure Log File Configuration.