IT4749 – Notes and Corrections for Code Provided for U07a1

1. Because CourseRegistration class maps to a database table (learner_registration) containing an auto_increment column (registration_id), Hibernate needs a small table called hibernate_sequence to keep track of the number inserted into this column for each entry. The hibernate_sequence table can be created like this:

```
CREATE TABLE hibernate_sequence (
  next_val bigint(20) DEFAULT NULL
);
```

You also need to insert a starting value:

```
INSERT INTO hibernate_sequence(next_val) VALUES(1);
```

2. In the persistence.xml file, there are some issues that need to be addressed.

   a. The persistence-unit should use the CourseRegistrationService class:

   ```
   <persistence-unit name="CourseRegistrationService" transaction-type="RESOURCE_LOCAL">
   ```

   b. There are 2 classes that need to be declared in place of u07d1.Grade:

   ```
   <class>u07a1.Course</class>
   <class>u07a1.CourseRegistration</class>
   ```

   c. While developing, it may be helpful to see the SQL that Hibernate is running:

   ```
   <property name="hibernate.show_sql" value="true"/>
   ```

3. In the CourseRegistrationService.java file, there are some parts of the code that need correction:

   a. The HQL in the getAllCourses() method should read as shown here:

   ```
   public List<Course> getAllCourses() {
        // Query is written in Hibernate Query Language (HQL) not SQL
        String hql = "SELECT crs FROM  Course crs ORDER BY courseCode";
        TypedQuery<Course> query = em.createQuery(hql, Course.class);
        return query.getResultList();
   }
   ```

   b. The createCourseRegistration() method needs to be revised to include a database transaction:

   ```
   public CourseRegistration createCourseRegistration(String learnerID, String courseCode) {
        CourseRegistration registration = new CourseRegistration(learnerID, courseCode);
   ```

```
        em.getTransaction().begin();
        em.persist(registration);
        em.flush();
        em.getTransaction().commit();
        return registration;
    }
```

c. The getAllCourseRegistration() needs to have a parameter for the learner ID and the HQL needs updating. The revised version uses a parameter called :id to create the Hibernate version of a query with a parameter. Note how the setParameter() method is added to set the value of the parameter.

```
    public List<CourseRegistration> getAllCourseRegistrations(String id) {
        // Query is written in Hibernate Query Language (HQL) not SQL
        String hql = "SELECT reg FROM CourseRegistration reg WHERE learnerID = :id";
        TypedQuery<CourseRegistration> query = em.createQuery(hql,
            CourseRegistration.class).setParameter("id", id);
        return query.getResultList();
    }
```

4. In the code for the JavaFX application, in addition to completing the over the needed code, add a stop() method to close the Hibernate connection when the application is closed.

```
public void stop() {
    em.close();
    emf.close();
}
```

5. To control the logging level in Hibernate, use a Logger call like this in main() before the call to launch(). Be sure to import java.util.logging.Logger.

```
Logger.getLogger("org.hibernate").setLevel(Level.SEVERE);
```