G. Holden
Java Development
Database Integration, Inheritance, Enumeration, & Polymorphism
Description and Sample Test Cases

```
//
//          Program Incorporates -
//
// Database:
//     Utilizes MySQL database connection and dynamically builds SQL statements
//       across as much as three levels of inheritance, allowing modification
//       and deletion of records from database;
//     Performs verification of user email as simulation of login, immediately
//       identifying which subclass the user is, creating a new object of that
//       type, and placing all relevant fields from the Database into that object
//       to be used as the CurrentUser for the remainder of the session;
//     Uses the database to police unique email values as well as assign unique
//       user IDs (key);
//     Retrieves record sets - could easily be enhanced by use of various
//       specific WHERE queries built with user input variables to allow
//       search functionality from Database;
//
// Enumeration:
//     Uses Enumeration in both base classes - Person and Posting - to guarantee
//       data integrity within critical fields;
//     Uses Enumeration within the MySQL Database structure to allow enumerated values
//       to be processed and evaluated properly;
//
// Polymorphism:
//     Reads in a RecordSet from database, identifies subtype, builds new object
//       of that type and stores it into an ArrayList of the Base class type;
//     Iterates through ArrayList and uses overridden .toString to output all
//       elements in the proper format for their individual class.
//
// Database class:
//       Allows for cleanliness and modularity of database connection;
//       Database name, table names, port number, passwords, etc are stored in
//       variables which can be accessed and modified quickly; Variables are then used
//       to create SQL statement string and create a connection object which is passed;
//
// Inheritance:
//     PERSON: Abstract Class for all Person Profiles. Contains methods that are overriden
//          in subclasses and chained together to handle each set of fields for the
//          particular class it is in;
//         EMPLOYER:  Subclass of Person for Profiles of Employers. Allowed to View, Create,
//             Update, and Delete all job Postings; Can Create, Update, and Delete
//             own Profile;
//         APPLICANT: Abstract Class inheriting from PERSON; all Subclasses of applicant
//                can View job Postings; Can Create, Update, and Delete own Profile; .
//             STUDENT: Subclass of Applicant; Intended to contain students seeking
//                internships;
//             REGULAR: Subclass of Applicant; Intended to contain experienced job seekers;
//
//     POSTING: Abstract class for all Postings;
//       INTERNSHIP: Subclass of Posting to hold postings related to internships and
//          college students;
//       REGULAR: Subclass of Posting to hold postings related to experienced job
//          seekers;
//
// Data Validation and Formatting:
//     All user input, including email addresses, is checked for valid format and/or
//       stripped of special characters before being allowed to be assigned to fields
//       in an attempt to stop possible SQL injection;
//     Some fields, such as phone number, have output formatted by getter for
//       cleaner output and better user experience;
//
```

Test Case 1 – Email address not in Database (correctly prompts to create new account), special characters in phone number are stripped out

```java
// Date: 3/23/2018
// Name of HW:  HW# 05 Job Tracking - Object Oriented Inheritance and Database Co
//

import java.sql.*;

public class DataBase {
    // Assign common pieces of SQL statements to variables for ease of use throug
    final String DBServer = "localhost";
    final String DBPort = "3306";
    protected static final String DBSchema = "job_tracking";
    final String DBOptions ="autoReconnect=true&useSSL=false";
    final String DBUser = "root";
    final String DBPassWd = "root";
    protected static final String PEOPLE_TABLE = "profile_data";
    protected static final String POSTINGS_TABLE = "posting_data";

    // Declare java.sql connection and statement objects
    public Connection con;
    public Statement stmt;


    public DataBase (){
        try {
            // Create connection and statement objects
            this.con = DriverManager.getConnection( url: "jdbc:mysql://"+DBServer+"
                    +DBSchema+"?"+DBOptions, DBUser, DBPassWd);
            this.stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
            // For future reference, EXECUTE UPDATE command returns number of row
        } catch (Exception e) {
            System.out.println(e);
```

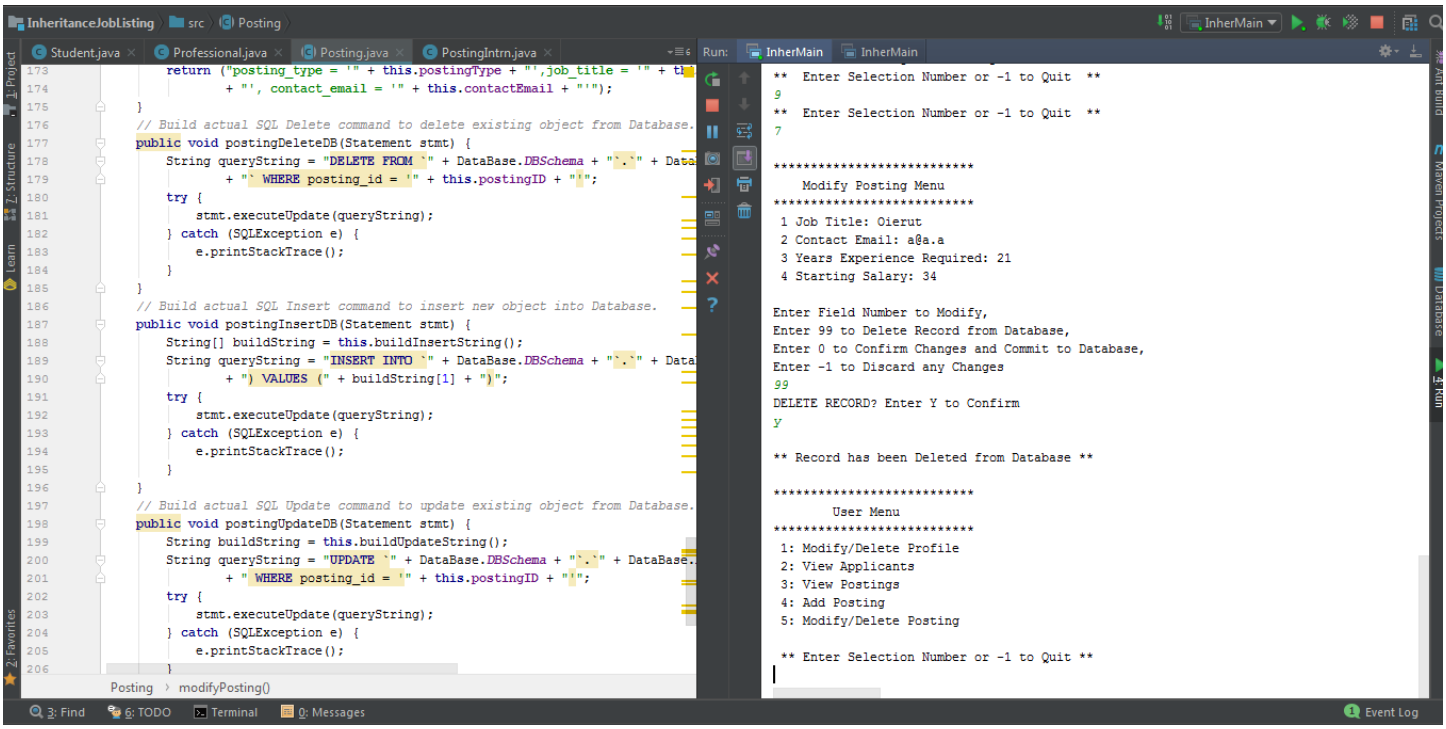
```
Please Enter Your Email Address to Log In
*****************************************
Enter Email:
gandalf@shire.net
You do not appear to have an account with us,
 ** Would you like to create an account? **
(Enter Y to create account, anything else to exit)
y
Enter Desired User Type by Number:
  1: EMPLOYER
  2: PROFESSIONAL
  3: STUDENT
1
Enter Last Name:
Grey
Enter First Name:
Gandalf
Enter 10 Digit Phone Number:
918*432*0987
Enter Company Name:
Wizards R Us
Enter Title:
Chief Operating Officer

-----------------------------------------------
**     Welcome to the Job Tracking System     **
**        Your Profile is Shown Below         **
-----------------------------------------------
Profile Type: EMPLOYER
ID Number: 0
NAME: Grey, Gandalf
Email: gandalf@shire.net
Phone: 918-432-0987
```

Test Case 2 – Successful addition of a new job to the database

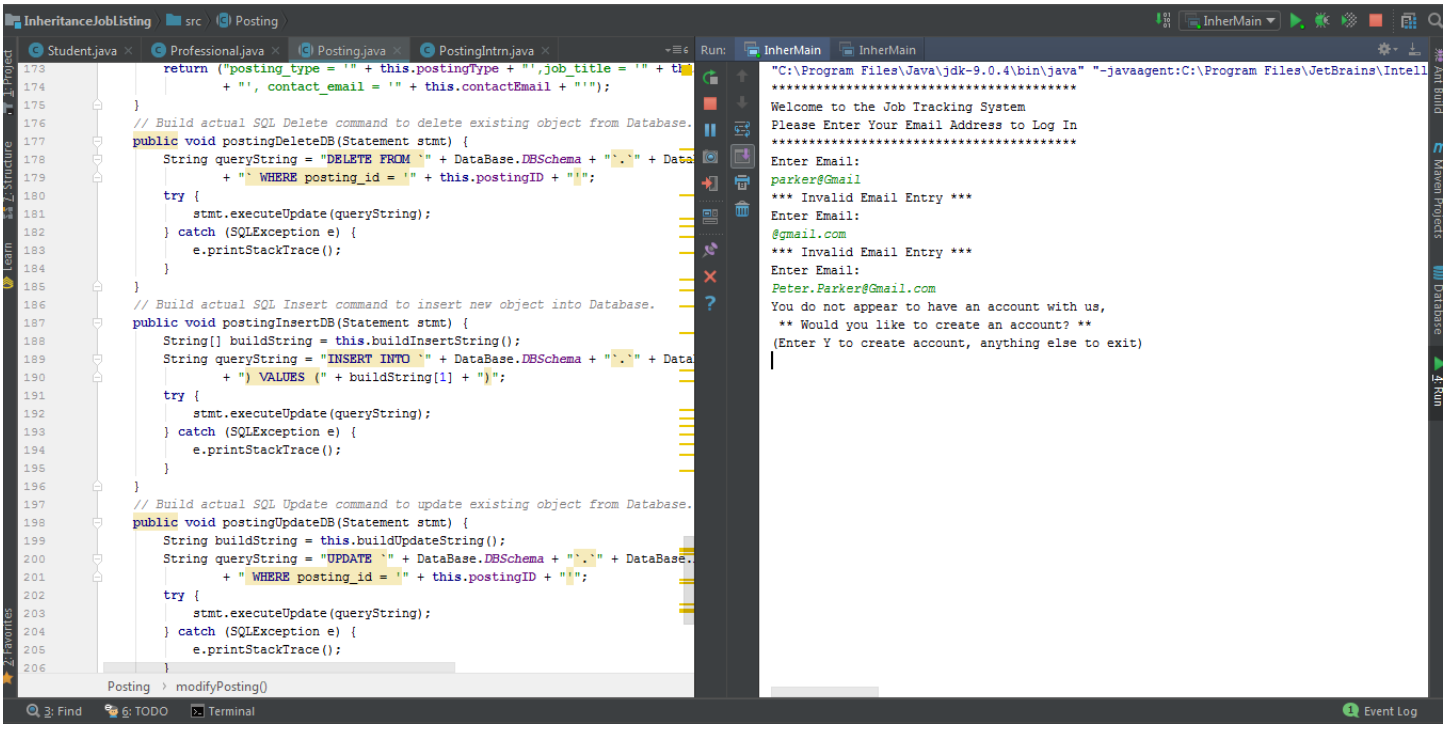
```java
// Date: 3/23/2018
// Name of HW:  HW# 05 Job Tracking - Object Oriented Inheritance and Database Co
//

import java.sql.*;

public class DataBase {
    // Assign common pieces of SQL statements to variables for ease of use throug
    final String DBServer = "localhost";
    final String DBPort = "3306";
    protected static final String DBSchema = "job_tracking";
    final String DBOptions ="autoReconnect=true&useSSL=false";
    final String DBUser = "root";
    final String DBPassWd = "root";
    protected static final String PEOPLE_TABLE = "profile_data";
    protected static final String POSTINGS_TABLE = "posting_data";

    // Declare java.sql connection and statement objects
    public Connection con;
    public Statement stmt;


    public DataBase (){
        try {
            // Create connection and statement objects
            this.con = DriverManager.getConnection( url: "jdbc:mysql://"+DBServer+"
                    +DBSchema+"?"+DBOptions, DBUser, DBPassWd);
            this.stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
            // For future reference, EXECUTE UPDATE command returns number of row
        } catch (Exception e) {
            System.out.println(e);
```

```
Company Name: Wizards R Us
Position/Title:  Chief Operating Officer

***************************
        User Menu
***************************
 1: Modify/Delete Profile
 2: View Applicants
 3: View Postings
 4: Add Posting
 5: Modify/Delete Posting

 ** Enter Selection Number or -1 to Quit **
4
Enter Desired Posting Type by Number:
  1: REGULAR
  2: INTERNSHIP
2
Enter Company Name:
Wizards R Us
Enter Name of Position:
Junior Apprentice
Enter Contact Email:
HR@Magic.Net
Enter Required Major:
Bachelors of MagicArts
Enter Required Grade Classification:
Senior

-----------------------------------------------
**   Thank You For Adding A New Job Posting   **
**    Posting Information is Shown Below      **
-----------------------------------------------
```

Test Case 3 – Successful deletion of Job Posting from Database. Entered value that did not correspond to a posting and was caught



Test Case 4 – Email regex pattern matching test. Successfully required string.string on right side of @, and also required at least one character on left side. Correctly allowed two parts on left side.



Test Case 5 – Successfully modified a field within a record and committed it back out to the Database