Database Systems Project 2 Documentation

"We have done this assignment completely on our own except for the tools/software acknowledged in the project report. We have not copied it, nor have we given our solution to anyone else. We understand that if we are involved in plagiarism or cheating we will have to sign an official form that we have cheated and that this form will be stored in our official university records. We also understand that we will receive a grade of 0 for the involved assignment and our grades will be reduced by one level (e.g., from A to A- or from B+ to B) for our first offense, and that we will receive a grade of "F" for the course for any additional offense of any kind."

-

Shahu Ronghe

Lalji Devda

Lovelesh Colaco

Objects used in this project

a) Sequences –

1.) gen_log_num

This sequence is used to generate unique values for log table starting from 1000 and incrementing by 1.

b) Packages –

1.) myPkg

This package has all the necessary procedures-

1. show_students

   This procedure outputs the student table.

2. remove_student

This package removes a student from student table taking b# as input

3. show_courses

   This procedure outputs the courses table.

4. display_preq_by_dept

   This procedure takes in dept_code, course# as input and outputs prerequisites of the courses.

5. show_course_credits

   This procedure outputs the course_credits table.

6. show_classes

   This procedure outputs the classes table.

7. display_std_by_classid

   This procedure outputs students using classid

8. show_enrollments

   This procedure outputs the g_enrollments table.

9. student_enrollment

   This procedure outputs student enrolled in class taking b# and classid as input

10. student_drop

    This procedure remove student from graduate enrollment taking b# and classid as input

11. show_grade

    This procedure outputs the score_grade table.

12. show_preq

    This procedure outputs the prerequisites table

13. show_logs

This procedure outputs the logs table

c) Triggers

1. delete_student_enrollments

   fires before delete on students table

   deletes tuple from g_enrollment using g_B#

2. increment_class_size

   Fires after Insert on g_enrollments

   Increments class_size in classes table by 1 using classid

3. decrement_class_size

   fires after delete on g_enrollments

   decrements class_size in classes table by 1 using classid

4. student_delete_log

   fires after delete on student

   Inserts tuple into log table of (gen_num, username, system date, student table, delete operation, b#)

5. student_enrolled_log

   Fires after insert on g_enrollments

   Inserts tuple into log table of (gen_num, username, system date, g_enrollments table, insert operation, B# and classid)

6. student_drop_log

   Fires after delete on g_enrollments

   Inserts tuple into log table of (gen_num, username, system date, student table, delete operation, B# and classid)

7. class_size_update_log

   Fires after update on classes

   Insert into log table of (gen_num, username, system date, classes table, update operation, classid and class_size)

TEAM REPORT

Meetings –

April 9, Saturday

- Understanding the project and plotting down the requirements.

April 10, Sunday

- Distributing the responsibilities and project planning.

April13, Wednesday

- Setting up the project

April 17, Sunday

- Completed procedures 1,2,3

April 21, Thursday

- Completed procedures 4,5

April 23, Saturday

- Completed procedures 6,7,8

April 25, Monday

- Completed all triggers and testing with basic testcases.

April 27, Wednesday

- Completed the android app for GUI and ran final execution.

PROJECT SCHEDULE

- Week 1
  - Setting up the project and understanding various requirements and distributing roles among the group members.
- Week 2
  - Completing all the procedures/triggers i.e PL/SQL Implementation.
- Week 3
  - Completing the interface/gui and documentation part of the project.

RESPONSIBILITIES –

- Shahu Ronghe – Worked on pl/sql queries and creating Android GUI
- Lalji Devda - worked on triggers, testing and documentation.
- Lovelesh Colaco- Worked on pl/sql queries and building text interface.

SELF ASSESSMENT OF THE TEAM WORK

We worked well together.

PL/SQL CODE:

Procedures:

```
/* CS532 Project 2
*
*	1. Shahu Ronghe
*	2. Lovelesh Colaco
*	3. Lalji Devda
*/

SET SERVEROUTPUT ON;
SET ERRORLOGGING ON;



-- Package Declaration.
CREATE OR REPLACE PACKAGE myPkg AS

	TYPE myCursor IS REF CURSOR;

	--	FOR STUDENTS
	PROCEDURE show_students(out_cur OUT myCursor);
	PROCEDURE remove_student(bnumber IN students.B#%TYPE);
```

```
        -- FOR COURSES
        PROCEDURE show_courses(out_cur OUT myCursor);
        PROCEDURE display_preq_by_dept(out_cur IN OUT myCursor, dept_code2
IN courses.dept_code%TYPE, course_no IN courses.course#%TYPE);



        -- FOR COURSE_CREDIT
        PROCEDURE show_course_credits(out_cur OUT myCursor);


        -- FOR CLASSES
        PROCEDURE show_classes(out_cur OUT myCursor);
        PROCEDURE display_std_by_classid(out_cur IN OUT myCursor, classid2 IN
classes.classid%TYPE);


        -- FOR ENROLLMENTS
        PROCEDURE show_enrollments(out_cur OUT myCursor);
        PROCEDURE student_enrollment(bnumber IN students.B#%TYPE, classid2
IN classes.classid%TYPE);
        PROCEDURE student_drop(bnumber IN students.B#%TYPE, classid2 IN
classes.classid%TYPE);


        -- FOR GRADE
        PROCEDURE show_grade(out_cur OUT myCursor);


        -- FOR PREREQUISITES
        PROCEDURE show_preq(out_cur OUT myCursor);


        -- FOR LOGS
        PROCEDURE show_logs(out_cur OUT myCursor);
END;
/

--Package Implementation.
CREATE OR REPLACE PACKAGE BODY myPkg AS


        -- FOR STUDENTS
        -- for displaying all tuples in this table.
        PROCEDURE show_students(out_cur OUT myCursor) AS
        BEGIN
                        OPEN out_cur FOR SELECT * FROM students;
        END;



        -- delete student from students using BNumber
        PROCEDURE remove_student(bnumber IN students.B#%TYPE) AS
        chkbno char(9);
        BEGIN
                chkbno := 0;
                BEGIN
```

```
                        SELECT B# INTO chkbno FROM students WHERE B# = bnumber;
                        EXCEPTION
                                WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'The BNumber is invalid.');
                                RETURN;
                END;

                BEGIN
                        DELETE FROM students WHERE B# = bnumber;
                        COMMIT;
                END;
        END;




        -- FOR COURSES
        -- for displaying all tuples in this table.
        PROCEDURE show_courses(out_cur OUT myCursor) AS
        BEGIN
                        OPEN out_cur FOR SELECT * FROM courses;
        END;

        -- 4. finding prequisites by dept_code and course#
```

```
	PROCEDURE display_preq_by_dept(out_cur IN OUT myCursor, dept_code2 
IN courses.dept_code%TYPE, course_no IN courses.course#%TYPE) AS


		coursechk number(3);

	BEGIN

		coursechk := 0;


		-- check if dept_code and course# exist in courses to futher query 
from prequisites.

		BEGIN

			SELECT course# INTO coursechk FROM courses WHERE course# 
= course_no AND UPPER(dept_code) = UPPER(dept_code2);

				EXCEPTION

					WHEN NO_DATA_FOUND THEN 
raise_application_error(-20001, dept_code2 || course_no || ' does not exist.');

				RETURN;

		END;




		-- get all prequisites course for the given dept_code and course#.

		BEGIN

			OPEN out_cur FOR WITH preq2 (pre_dept_code, pre_course#, 
dept_code, course#) AS (SELECT pre_dept_code, pre_course#, dept_code, course# 
FROM prerequisites m WHERE UPPER(dept_code) = dept_code2 AND course# = 
course_no UNION ALL SELECT  m.pre_dept_code, m.pre_course#, m.dept_code, 
m.course# FROM prerequisites m INNER JOIN preq2 p ON p.pre_dept_code = 
m.dept_code AND p.pre_course# = m.course#)
```

```
                    SELECT CONCAT (pre_dept_code, pre_course#) AS
prerequisites FROM preq2;
                END;
        END;




        -- FOR COURSE CREDIT
        -- for displaying all tuples in this table.
        PROCEDURE show_course_credits(out_cur OUT myCursor) AS
        BEGIN
                    OPEN out_cur FOR SELECT * FROM course_credit;
        END;




        -- FOR CLASSES
        -- for displaying all tuples in this table.
        PROCEDURE show_classes(out_cur OUT myCursor) AS
        BEGIN
                    OPEN out_cur FOR SELECT * FROM classes;
```

```
    END;


    -- 3. finding students with classid.
    PROCEDURE display_std_by_classid(out_cur IN OUT myCursor, classid2 IN
classes.classid%TYPE) AS
        classidstore char(5);
    BEGIN
        classidstore := 0;
        -- check if classid is available in DB, else throw exception and return.
        BEGIN
                SELECT classid INTO classidstore  FROM classes WHERE
classid = classid2;


                EXCEPTION
                        WHEN NO_DATA_FOUND THEN
                            raise_application_error(-20001, 'The
classid is invalid.');
                    RETURN;


        END;


        -- get all students information based on classid.
        BEGIN
            OPEN out_cur FOR SELECT DISTINCT s.B#, s.first_name,
s.last_name FROM students s, g_enrollments ge, classes c WHERE s.B# = ge.g_B#
AND ge.classid = c.classid AND c.classid = classid2;
```

```
            END;
        END;


        -- FOR ENROLLMENTS
        -- for displaying all tuples in this table.
        PROCEDURE show_enrollments(out_cur OUT myCursor) AS
        BEGIN
                    OPEN out_cur FOR SELECT * FROM g_enrollments;
        END;


        -- 5. enroll a student in class by given classid and B number.
        PROCEDURE student_enrollment(bnumber IN students.B#%TYPE, classid2
    IN classes.classid%TYPE) AS
        chkbno char(9);
        chkbno_grad char(9);
        chkclassid char(5);
        chkclassidsem char(5);
        chkclasssize char(5);
        chkenroll number(1);
        chktotalenroll number(2);
        chkpreqcount number(2);


        BEGIN
```

```
		-- check b number exist in student table.
		BEGIN
			chkbno := 0;

			SELECT B# INTO chkbno FROM students WHERE B# = bnumber;
			EXCEPTION
						WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'The B# is invalid. ' || bnumber);
						RETURN;
		END;

		-- check b number is graduate student or not.
		BEGIN
			chkbno_grad := 0;

			SELECT B# INTO chkbno_grad FROM students WHERE B# =
bnumber AND (st_level = 'master' OR st_level = 'PhD');
			EXCEPTION
						WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'This is not a graduate student.');
						RETURN;
		END;

		-- check classid exists in classes table.
		BEGIN
```

```
			chkclassid := 0;
			SELECT classid INTO chkclassid FROM classes WHERE classid = 
classid2;
			EXCEPTION
						WHEN NO_DATA_FOUND THEN 
raise_application_error(-20001, 'The classid is invalid.');
						RETURN;
		END;


		-- check if class offered in current semester.
		BEGIN
			chkclassidsem := 0;
			SELECT classid INTO chkclassidsem FROM classes WHERE year = 
'2021' AND semester = 'Spring' AND classid = classid2;
			EXCEPTION
						WHEN NO_DATA_FOUND THEN 
raise_application_error(-20001, 'Cannot enroll into a class from a previous 
semester.');


						RETURN;
		END;


		-- check whether the class is full.
		BEGIN
			chkclasssize := 0;
```

```
            SELECT classid INTO chkclasssize FROM classes WHERE classid =
classid2 AND class_size < limit;
            EXCEPTION
                        WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'The class is already full.');

                        RETURN;
        END;

        -- check student already in the class.
        BEGIN
            chkenroll := 0;
            BEGIN
                SELECT COUNT(*) INTO chkenroll FROM g_enrollments
WHERE g_B# = bnumber AND classid = classid2;
            END;

            IF(chkenroll = 1)
                    THEN raise_application_error(-20001, 'The
student is already in the class.');

                        RETURN;
            END IF;
        END;

        -- check student is not enrolled in more than 5 classes in semester.
```

```
                BEGIN

                        chktotalenroll := 0;

                        BEGIN

                                SELECT COUNT(*) INTO chktotalenroll FROM 
g_enrollments g, classes c WHERE g.classid = c.classid AND g_B# = bnumber AND 
year = 2021 AND semester = 'Spring';

                        END;

                        IF(chktotalenroll >= 5)

                                        THEN raise_application_error(-20001, 'Student 
cannot be enrolled in more than 5 classes in the same semester.');


                                                RETURN;

                        END IF;

                END;


                -- check if student completed prerequisites with C grade.

                BEGIN

                        chkpreqcount := 0;

                        BEGIN

                                SELECT COUNT(lgrade) INTO chkpreqcount FROM 
score_grade WHERE score IN (SELECT score FROM g_enrollments WHERE classid 
IN (SELECT classid FROM classes WHERE dept_code IN (SELECT pre_dept_code 
FROM prerequisites WHERE dept_code IN (SELECT dept_code FROM classes 
WHERE classid = classid2) AND course# IN (SELECT course# FROM classes WHERE 
classid = classid2)) AND course# IN (SELECT  pre_course# FROM prerequisites 
WHERE dept_code IN (SELECT dept_code FROM classes WHERE classid = classid2) 
AND course# IN (SELECT course# FROM classes WHERE classid = classid2))) AND 
g_B# = bnumber) AND lgrade > 'C';
```

```
                END;


                IF (chkpreqcount > 0)
                        THEN raise_application_error(-20001, 'prerequisites not
satisfied.');

                                RETURN;
                END IF;
        END;



        -- all above conditions are satisfied so insert the tuple so that student
can be enrolled and trigger the events.
        BEGIN
                INSERT INTO g_enrollments VALUES (bnumber, classid2, null);
                COMMIT;
                dbms_output.put_line('Student enrolled successfully!');
        END;


    END;




    -- 6. to drop a student from g_enrollments
```

```
	PROCEDURE student_drop(bnumber IN students.B#%TYPE, classid2 IN
classes.classid%TYPE) AS

	chkbno2 char(9);
	chkbno_grad2 char(9);
	chkclassid2 char(5);
	chkclassidsem2 char(5);
	chkenroll2 number(1);
	chktotalenroll2 number(2);
	chkpreqcount2 number(2);
	chkonlyoneclass2 number(2);

	BEGIN

		-- check b number exist in student table.
		BEGIN
			chkbno2 := 0;

			SELECT B# INTO chkbno2 FROM students WHERE B# =
bnumber;
			EXCEPTION
						WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'The B# is invalid');

						RETURN;
		END;

		-- check b number is graduate student or not.
		BEGIN
			chkbno_grad2 := 0;

			SELECT B# INTO chkbno_grad2 FROM students WHERE B# =
bnumber AND (st_level = 'master' OR st_level = 'PhD');
			EXCEPTION
						WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'This is not a graduate student.');

						RETURN;
		END;

		-- check classid exists in classes table.
		BEGIN
			chkclassid2 := 0;
			SELECT classid INTO chkclassid2 FROM classes WHERE classid =
classid2;
			EXCEPTION
						WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'The classid is invalid.');

						RETURN;
		END;
```

```
        -- check student already in the class.
        BEGIN
            chkenroll2 := 0;
            BEGIN
                SELECT COUNT(*) INTO chkenroll2 FROM g_enrollments
WHERE g_B# = bnumber AND classid = classid2;
            END;

            IF(chkenroll2 = 0)
                THEN raise_application_error(-20001, 'The student is not
enrolled in the class.');

                        RETURN;
            END IF;
        END;

        -- check if class offered in current semester.
        BEGIN
            chkclassidsem2 := 0;
            SELECT classid INTO chkclassidsem2 FROM classes WHERE year
= '2021' AND semester = 'Spring' AND classid = classid2;
            EXCEPTION
                        WHEN NO_DATA_FOUND THEN
raise_application_error(-20001, 'Only Enrollment in current semester can be
dropped.');
```

```
                                        RETURN;

                    END;



                    -- check if the class is the only class in Spring 2021, then reject drop.
                    BEGIN
                         chkonlyoneclass2 := 0;
                         SELECT COUNT(*) INTO chkonlyoneclass2 FROM g_enrollments
ge where ge.classid in ( select classid from classes c where year=2021 and
semester='Spring') and g_b#=bnumber;

                         IF (chkonlyoneclass2 = 1)
                              THEN
                                   raise_application_error(-20001, 'This is the only
class for this student in Spring 2021 and cannot be dropped');
                                   RETURN;
                         END IF;
                    END;

                    -- finally drop the class for this student
                    BEGIN
                         DELETE FROM g_enrollments WHERE g_B# = bnumber AND
classid = classid2;
                         COMMIT;
```

```
        END;
END;



-- FOR GRADE
-- for displaying all tuples in this table.
PROCEDURE show_grade(out_cur OUT myCursor) AS
BEGIN
                OPEN out_cur FOR SELECT * FROM score_grade;
END;




-- FOR PREREQUISITES
-- for displaying all tuples in this table.
PROCEDURE show_preq(out_cur OUT myCursor) AS
BEGIN
                OPEN out_cur FOR SELECT * FROM prerequisites;
END;
```

```
	-- FOR LOGS
	-- for displaying all tuples in this table.
	PROCEDURE show_logs(out_cur OUT myCursor) AS
	BEGIN
			OPEN out_cur FOR SELECT * FROM logs;
	END;


END;
/
```

Triggers:

```
/* CS532 Project 2
*
*	1. Shahu Ronghe
*	2. Lovelesh Colaco
*	3. Lalji Devda
*/

-- DROP SECTION
DROP SEQUENCE gen_log_num;



-- 1. GENERATE LOG SEQUENCE WITH 1000
```

```
CREATE SEQUENCE gen_log_num
        START WITH 1000
        INCREMENT BY 1;



-- 8. All triggers.
-- student delete safe trigger
CREATE OR REPLACE TRIGGER delete_student_enrollments
BEFORE DELETE ON students
FOR EACH ROW
BEGIN
        DELETE FROM g_enrollments WHERE g_B# = :OLD.B#;
END;
/

-- update classes table to increment class size when students enrolls.
CREATE OR REPLACE TRIGGER increment_class_size
AFTER INSERT ON g_enrollments
FOR EACH ROW
BEGIN
        UPDATE classes SET class_size  = class_size + 1 WHERE classid =
:NEW.classid;
END;
/
```

```
-- update classes table to decrement class size when student drops the class.
CREATE OR REPLACE TRIGGER decrement_class_size
AFTER DELETE ON g_enrollments
FOR EACH ROW
BEGIN
	UPDATE classes SET class_size  = class_size - 1 WHERE classid = :OLD.classid;
END;
/

-- Logging triggers.
-- student delete update log
CREATE OR REPLACE TRIGGER student_delete_log
AFTER DELETE ON students
FOR EACH ROW
DECLARE
	login VARCHAR2(24);
BEGIN
	login := USER;
	INSERT INTO logs VALUES (gen_log_num.nextval, login, sysdate, 'students',
'delete', :OLD.B#);
END;
/

-- student enrolled update log
CREATE OR REPLACE TRIGGER student_enrolled_log
```

```
AFTER INSERT ON g_enrollments

FOR EACH ROW

DECLARE

        login VARCHAR2(24);

        keyvalue VARCHAR2(24);

BEGIN

        login := USER;

        keyvalue := :NEW.g_B# || ',' || :NEW.classid;

        INSERT INTO logs VALUES (gen_log_num.nextval, login, sysdate,
'g_enrollments', 'insert', keyvalue);

END;

/


-- student drop class update log

CREATE OR REPLACE TRIGGER student_drop_log

AFTER DELETE ON g_enrollments

FOR EACH ROW

DECLARE

        login VARCHAR2(24);

        keyvalue VARCHAR2(24);

BEGIN

        login := USER;

        keyvalue := :OLD.g_B# || ',' || :OLD.classid;

        INSERT INTO logs VALUES (gen_log_num.nextval, login, sysdate,
'g_enrollments', 'delete', keyvalue);
```

```
END;
/

-- class_size update log
CREATE OR REPLACE TRIGGER class_size_update_log
AFTER UPDATE ON classes
FOR EACH ROW
DECLARE
        login VARCHAR2(24);
        keyvalue VARCHAR2(24);
BEGIN
        login := USER;
        keyvalue := :OLD.classid || ',' || :NEW.class_size;
        INSERT INTO logs VALUES (gen_log_num.nextval, login, sysdate, 'classes',
'update', keyvalue);
END;
/
```

JAVA CODE:

```
package src.dbtesting;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;


import oracle.jdbc.OracleTypes;
import oracle.jdbc.pool.OracleDataSource;


public class OurTesting {


	/**
	 * Main function to start the menu driven program.
	 *
	 * @param args arguments to pass to the program. args[0] is username and
args[1]
	 *           is password.
	 * @throws SQLException
	 */
	public static void main(String[] args) throws SQLException {
		if (args.length != 2) {
			System.out.println("Invalid arguments. Enter 2 arguments as
username and password.");
			return;
		}
		Connection conn = null;
		OracleDataSource ds = null;
		Scanner sc = new Scanner(System.in);
		try {
```

```
                        ds = new oracle.jdbc.pool.OracleDataSource();
                        // database source location

ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:ACAD111");


                        String username = args[0];
                        String password = args[1];


                        try {
                                // database connection with given username and
password.
                                conn = ds.getConnection(username, password);
                                System.out.println("\nSuccessfully connected to Oracle
database");
                        } catch (Exception e) {
                                System.err.println("Connection Error: " +
e.getMessage());
                                System.exit(1);
                        }
                }


                catch (SQLException ex) {
                        System.out.println("\n*** SQLException caught ***\n" +
ex.getMessage());
                        System.exit(1);
                } catch (Exception e) {
                        System.out.println("\n*** other Exception caught ***\n" +
e.getMessage());
                        if (conn != null) {
                                conn.close();
                        }
                        System.exit(1);
                }

                while (true) {

                        try {
                                System.out.print("\n******* WELCOME TO STUDENT
SERVICE SYSTEM *******\n");
                                System.out.print("\nEnter '1' to show all the tables\n");
                                System.out.print(
                                                "Enter '2' to List B#, First name and Last
name of every student in class using classid\n");
                                System.out.print("Enter '3' to check the prerequisite
courses for a given course\n");
                                System.out.print("Enter '4' to enroll graduate student
into class\n");
                                System.out.print("Enter '5' to drop graduate student
from class\n");
                                System.out.print("Enter '6' to delete student from
students table\n");
                                System.out.print("Enter '7' to exit\n");
                                int option = sc.nextInt();
                                CallableStatement call;

                                if (option == 7) {
                                        System.out.println("Database disconnected!
Thank You! Exiting now");
                                        conn.close();
                                        System.exit(0);
                                }
                                boolean innerFlag = true;
                                switch (option) {
                                case 1:
                                        while (innerFlag) {
                                                System.out.println("Select option for table
");
                                                System.out.println("Enter '1' for Students
table");
                                                System.out.println("Enter '2' for Courses
table");
                                                System.out.println("Enter '3' for Course
Credit table");
                                                System.out.println("Enter '4' for Classes
table");
                                                System.out.println("Enter '5' for
G_enrollments table");
```

```
                                        System.out.println("Enter '6' for
Score_Grade table");
                                        System.out.println("Enter '7' for
Prerequisites table");
                                        System.out.println("Enter '8' for Logs
table");
                                        System.out.println("Enter '9 to go back");
                                        System.out.println("Enter '0' to exit");

                                        int op = sc.nextInt();

                                        switch (op) {

                                        case 1:
                                             call = conn.prepareCall("begin
myPkg.show_students(?); end;",

ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
                                             call.registerOutParameter(1,
OracleTypes.CURSOR);
                                             call.execute();
                                             printResult((ResultSet)
call.getObject(1));
                                             call.close();

                                             break;
                                        case 2:
                                                call = conn.prepareCall("begin
myPkg.show_courses(?);end;");
                                                call.registerOutParameter(1,
OracleTypes.CURSOR);
                                                call.execute();
                                                printResult((ResultSet)
call.getObject(1));
                                                call.close();
                                                break;

                                        case 3:
                                                call = conn.prepareCall("begin
myPkg.show_course_credits(?);end;");
                                                call.registerOutParameter(1,
OracleTypes.CURSOR);
                                                call.execute();
                                                printResult((ResultSet)
call.getObject(1));
                                                call.close();
                                                break;

                                        case 4:
                                                call = conn.prepareCall("begin
myPkg.show_classes(?);end;");
```

```
                                        call.registerOutParameter(1,
OracleTypes.CURSOR);
                                        call.execute();
                                        printResult((ResultSet)
call.getObject(1));
                                        call.close();
                                        break;

                                case 5:
                                        call = conn.prepareCall("begin
myPkg.show_enrollments(?);end;");
                                        call.registerOutParameter(1,
OracleTypes.CURSOR);
                                        call.execute();
                                        printResult((ResultSet)
call.getObject(1));
                                        call.close();
                                        break;

                                case 6:
                                        call = conn.prepareCall("begin
myPkg.show_grade(?);end;");
                                        call.registerOutParameter(1,
OracleTypes.CURSOR);
                                        call.execute();
                                        printResult((ResultSet)
call.getObject(1));
```

```
                    call.close();
                    break;

                case 7:
                    call = conn.prepareCall("begin
myPkg.show_preq(?);end;");
                    call.registerOutParameter(1,
OracleTypes.CURSOR);
                    call.execute();
                    printResult((ResultSet)
call.getObject(1));
                    call.close();
                    break;

                case 8:
                    call = conn.prepareCall("begin
myPkg.show_logs(?);end;");
                    call.registerOutParameter(1,
OracleTypes.CURSOR);
                    call.execute();
                    printResult((ResultSet)
call.getObject(1));
                    call.close();
                    break;

                case 9:
```

```
                                    innerFlag = false;
                                    break;

                                case 0:
                                    System.out.println("Database
disconnected! Thank You! Exiting now");
                                    conn.close();
                                    sc.close();
                                    System.exit(0);
                                    break;

                                default:
                                    System.out.println("Invalid option");
                                    break;
                                }
                            }
                            break;

                        case 2:
                            // list the students enrolled in a class.
                            System.out.println("Enter class id");
                            String classid = sc.next();

                            call = conn.prepareCall("begin
myPkg.display_std_by_classid(?,?); end;");
```

```
                                        call.registerOutParameter(1,
OracleTypes.CURSOR);
                                        call.setString(2, classid);
                                        call.execute();
                                        printResult((ResultSet) call.getObject(1));
                                        call.close();
                                        break;

                                case 3:
                                        // list the prerequisites using the dept_code and
course number.
                                        System.out.println("Enter department code");
                                        String deptc = sc.next();

                                        System.out.println("Enter course number");
                                        String course_no = sc.next();

                                        call = conn.prepareCall("begin
myPkg.display_preq_by_dept(?,?,?); end;");
                                        call.registerOutParameter(1,
OracleTypes.CURSOR);
                                        call.setString(2, deptc.toUpperCase());
                                        call.setString(3, course_no);
                                        call.execute();
                                        printResult((ResultSet) call.getObject(1));
                                        call.close();
```

```
                    break;

                case 4:
                    // enroll student into a course using a bNumber
and classid.
                    System.out.println("Enter B#");
                    String bnum = sc.next();

                    System.out.println("Enter classid");
                    String classid2 = sc.next();

                    call = conn.prepareCall("begin
myPkg.student_enrollment(?,?); end;");
                    call.setString(1, bnum);
                    call.setString(2, classid2);
                    call.execute();
                    call.close();
                    System.out.println("Student '" + bnum + "' has
been enrolled in class: " + classid2);
                    break;

                case 5:
                    // drop student course from g_enrollments table.
                    System.out.println("Enter B#");
                    String bnum2 = sc.next();
```

```
                                        System.out.println("Enter classid");
                                        String classid3 = sc.next();
                                        call = conn.prepareCall("begin
myPkg.student_drop(?,?); end;");
                                        call.setString(1, bnum2);
                                        call.setString(2, classid3);
                                        call.execute();
                                        call.close();
                                        System.out.println("Student '" + bnum2 + "' has
been dropped from class: " + classid3);
                                        break;

                                case 6:
                                        // remove student from student table and entry
from g_enrollments table.
                                        System.out.println("Enter B#");
                                        String bnum3 = sc.next();
                                        call = conn.prepareCall("begin
myPkg.remove_student(?); end;");
                                        call.setString(1, bnum3);
                                        call.execute();
                                        call.close();
                                        System.out.println("Student '" + bnum3 + "' has
been deleted successfully!");
                                        break;
```

```
				}

			}

			catch (SQLException ex) {
				System.out.println("\nERROR: " +
ex.getMessage().split("\n")[0].split(":")[1]);

			} catch (Exception e) {
				System.out.println("Other Exception caught: " +
e.getMessage());
				if (conn != null) {
					conn.close();
				}
			}
		}
	}

	/**
	 * prints the query data in tabular format.
	 *
	 * @param resultSet contains the data of the query.
	 * @throws SQLException
	 */
	private static void printResult(ResultSet resultSet) throws SQLException {
```

```
                    DBTablePrinter.printResultSet(resultSet);


                    // legacy printing code. not well formatted.
                    /*
                     * ResultSetMetaData rsmd = resultSet.getMetaData(); int
columnsNumber =
                     * rsmd.getColumnCount();
                     *
                     * for (int i = 1; i <= columnsNumber; i++) {
                     * System.out.print(rsmd.getColumnName(i)+ "\t"); }
System.out.println("");
                     *
                     * while (resultSet.next()) { for (int i = 1; i <= columnsNumber; i++) {
String
                     * columnValue = resultSet.getString(i);
System.out.print(columnValue + "\t\t");
                     * } System.out.println(""); }
                     */
          }


}
```