Some hints on Java programming

```
Compile:
#javac Classname.java
File Classname.java must look like this
//import necessary libralies e.g. java.sql.*, java.io.*;
 public class Classname //class name must the same as the file name
 {
 }
Run
#java Classname

Exception handling:
try{
    //..
}catch(SQLException e){
    //process i.e. print e.toString()
}
Get data from keyboard:
BufferedReader input = new BufferedReader(
                              new InputStreamReader(System.in));
System.out.println("Input:");
String course = input.readLine();
```

JDBC remind

```
DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
                String url =
   "jdbc:oracle:thin:@delphi.medic.chalmers.se:1521:medic1";
                String userName = ""; // Your username goes here!
                String password = ""; // Your password goes here!
Connection conn = DriverManager.getConnection(url,userName,password);

Statement stmt = conn.createStatement();
String day, hour;
//get day and hour value somewhere
String query = "SELECT course, day, hour, teacher, room \n" +
               "FROM Classes A, Courses C, Teachers T \n" +
                "WHERE course = C.name AND teacher = T.name \n" +
                "AND day = '" + day + "' AND hour = " + hour;

ResultSet rs = stmt.executeQuery(query);

while (rs.next()) {
    System.out.println("Course:" + rs.getString(1) +
    "Day:" + rs.getString(2) + "Room:" + rs.getString(5));
}
```

1. Booking lectures

```
/* get input course and day into thusly named variables */
ResultSet rs = stmt.executeQuery("SELECT room FROM Courses C, Teachers T WHERE teacher
  = T.name AND C.name ='" + course + "'");
rs.next();
String room = rs.getString(1);
String getTimes = "SELECT * FROM ClassTimes WHERE time NOT IN (" +
     "SELECT hour FROM Classes A, Courses C, Teachers T WHERE "
     + "course = C.name AND teacher = T.name AND "
     + "room = '" + room + "' AND day = '" + day + "')";
rs = stmt.executeQuery(getTimes);
if (!rs.next()) {
  rs = stmt.executeQuery("SELECT * FROM ClassTimes");
  day = "Someday";
  rs.next();
}
do {
 /* print time */
} while (rs.next());

/* get chosen time into variable hour */
stmt.executeUpdate("INSERT INTO Classes VALUES ('" + course + "','" + day + "'," +
  hour + ")")
```

2. Making room in rooms

```
String getRooms = "SELECT R.name, nrStudents " +
                  "FROM Courses C, Teachers T, Rooms R " +
                  "WHERE teacher = T.name AND room = R.name " +
                  "AND nrStudents > nrSeats";
 ResultSet rs = stmt.executeQuery(getRooms);

 while (rs.next()) {
  stmt.executeUpdate("UPDATE Rooms SET nrSeats = " + rs.getInt(2) +
                     " WHERE name = '" + rs.getString(1)) + "'";
 }
```

3. Resolve Somedays

```
String[] days = new String[6];
 ResultSet rs =
      stmt.executeQuery("SELECT * FROM ClassDays WHERE day != 'Someday'");
 int i = 0;
 while (rs.next())
   days[i++] = rs.getString(1);
```

3. Resolve Somedays (cont.)

```
while (true) {
   /* Sleep for a random time */
   String getClasses = "SELECT course, hour FROM Classes WHERE day =
   'Someday'";
   ResultSet rs = stmt.executeQuery(getClasses);
   while (rs.next()) {
     String course = rs.getString(1);
     int hour = rs.getInt(2);
     /* get a random number between 1 and 6 into variable i */
     day = days[i];
     String clashes = "SELECT * FROM Classes A, Courses C, Teachers T " +
                      "WHERE course = C.name AND teacher = T.name " +
                      "AND day = '" + day + "' AND hour = " + hour;
     ResultSet rs2 = stmt2.executeQuery(clashes);
     if (!rs2.next()) {
       stmt2.executeUpdate("DELETE FROM Classes WHERE day = 'Someday' " +
                           " AND hour = " + hour +
                           " AND course = '" + course + "'";
       stmt2.executeUpdate("INSERT INTO Classes VALUES ('"
                           + course + "','" + day + "'," + hour + ")");
      }
   }
 }
```

Transactions - remind

ACID transactions

- Atomic: whole or nothing
- Consistent: constraints are preserved
- Isolated: no interactions among transactions
- Durable: lost tolerant with system crash

Isolation levels:

-serializable: non-interference with other transactions

-read committed: allows other transactions to modify and data affected after committing

-read uncommitted: allows other transaction to modify and data affected right after modification without committing.

-Repeatable read: same as read committed, but if the transaction read more than one, at least the same tuples again are guaranteed

Assignment trigger hints

```
CREATE OR REPLACE TRIGGER CourseRegistration
INSTEAD OF INSERT ON RegisteredView
REFERENCING NEW AS new
FOR EACH ROW
DECLARE
maxNum INT;
currentNum INT;
limited INT;
BEGIN
     SELECT COUNT(*) INTO limited FROM LimitedCourses WHERE code = :new.course;
     IF limited > 0 THEN --limited course
        SELECT nrStudent INTO maxNum FROM LimitedCourses WHERE code = :new.course;
        SELECT COUNT(*) INTO currentNum
        FROM Registered
        WHERE course = :new.course;
        IF currentNum < maxNum THEN -- still has a place
           INSERT INTO Registered ...
         ELSE -- the course is full
           INSERT INTO WaitingList ...--remember to process the priority
         END IF;
     ELSE -- normal course
        INSERT INTO Registered ...
     END IF;
END;
```

Assignment trigger hints

```
CREATE OR REPLACE TRIGGER CourseUnregistration
INSTEAD OF DELETE ON RegisteredView
REFERENCING OLD AS old
FOR EACH ROW
DECLARE
firstStuInQueue Students.ID%TYPE;
waitingNum INT;
BEGIN
     DELETE FROM Registered ...;
     SELECT COUNT(student) INTO waitingNum FROM WaitingList WHERE course = :old.course;
     IF waitingNum > 0 THEN -- there are waiting students
          SELECT student INTO firstStuInQueue
          FROM WaitingList WHERE ...//first priority
          INSERT INTO Registered ... --first student in the queue
          DELETE FROM WaitingList ... --first student in the queue
          --UPDATE FROM WaitingList ...--update the priority if necessary
     END IF;
END;
```