1. What design principles does this code violate?
2. Without actually doing so, explain how you would refactor this code to improve its design.

Course.java

```
import java.sql.*;

public class Course {
   private String name;
   private int credits;
   static String url = "jdbc:odbc:Reggie";

   static { try { Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
           catch (Exception ignored) {} }

   public static Course create(String name, int credits)
      throws Exception
   {
     Connection conn = null;

     try {
        conn = DriverManager.getConnection(url, "", "");
        Statement statement = conn.createStatement();
        statement.executeUpdate(
           "DELETE FROM course WHERE name = '" + name + "';");
        statement.executeUpdate(
           "INSERT INTO course VALUES ('" + name
           + "', '" + credits + "');");
        return new Course(name, credits);
     } finally {
        try { conn.close(); } catch (Exception ignored) {}
     }
   }

   public static Course find(String name) {
      Connection conn = null;
      try {
         conn = DriverManager.getConnection(url, "", "");
         Statement statement = conn.createStatement();
         ResultSet result = statement.executeQuery(
            "SELECT * FROM course WHERE Name = '" + name + "';");
         if (!result.next()) return null;
```

```
            int credits = result.getInt("Credits");
            return new Course(name, credits);
        } catch (Exception ex) {
            return null;
        } finally {
            try { conn.close(); } catch (Exception ignored) {}
        }
    }

    public void update() throws Exception {
        Connection conn = null;

        try {
            conn = DriverManager.getConnection(url, "", "");
            Statement statement = conn.createStatement();

            statement.executeUpdate(
                "DELETE FROM COURSE WHERE name = '" + name + "';");
            statement.executeUpdate(
                "INSERT INTO course VALUES('" +
                name + "','" + credits + "');");
        } finally {
            try { conn.close(); } catch (Exception ignored) {}
        }
    }

    Course(String name, int credits) {
        this.name = name;
        this.credits = credits;
    }

    public int getCredits() {
        return credits;
    }

    public String getName() {
        return name;
    }
}
```

Offering.java

```
import java.sql.*;

public class Offering {
    private int id;
    private Course course;
    private String daysTimes;

    static String url = "jdbc:odbc:Reggie";

    static { try { Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
            catch (Exception ignored) {} }

    public static Offering create(Course course, String daysTimesCsv)
        throws Exception
    {
        Connection conn = null;

        try {
            conn = DriverManager.getConnection(url, "", "");
            Statement statement = conn.createStatement();

            ResultSet result = statement.executeQuery(
                "SELECT MAX(ID) FROM offering;");
            result.next();
            int newId = 1 + result.getInt(1);

            statement.executeUpdate("INSERT INTO offering VALUES ('"
                    + newId + "','" + course.getName()
                    + "','" + daysTimesCsv + "');");
            return new Offering(newId, course, daysTimesCsv);
        } finally {
            try { conn.close(); } catch (Exception ignored) {}
        }
    }

    public static Offering find(int id) {
        Connection conn = null;

        try {
            conn = DriverManager.getConnection(url, "", "");
            Statement statement = conn.createStatement();
            ResultSet result = statement.executeQuery(
```

```
            "SELECT * FROM offering WHERE ID =" + id + ";");
        if (result.next() == false)
            return null;

        String courseName = result.getString("Course");
        Course course = Course.find(courseName);
        String dateTime = result.getString("DateTime");
        conn.close();

        return new Offering(id, course, dateTime);
    } catch (Exception ex) {
        try { conn.close(); } catch (Exception ignored) {}
        return null;
    }
}

public void update() throws Exception {
    Connection conn = null;

    try {
        conn = DriverManager.getConnection(url, "", "");
        Statement statement = conn.createStatement();

        statement.executeUpdate(
            "DELETE FROM Offering WHERE ID=" + id + ";");
        statement.executeUpdate(
                "INSERT INTO Offering VALUES('" + id + "','" +
                  course.getName() + "','" + daysTimes + "');");
    } finally {
        try { conn.close(); } catch (Exception ignored) {}
    }
}

public Offering(int id, Course course, String daysTimesCsv) {
    this.id = id;
    this.course = course;
    this.daysTimes = daysTimesCsv;
}

public int getId() {
    return id;
}
```

```
    public Course getCourse() {
        return course;
    }

    public String getDaysTimes() {
        return daysTimes;
    }

    public String toString() {
        return "Offering " + getId() + ": "
+ getCourse() + " meeting " + getDaysTimes();
    }
}
```

Schedule.java

```
import java.util.*;
import java.sql.*;

public class Schedule {
    String name;
    int credits = 0;
    static final int minCredits = 12;
    static final int maxCredits = 18;
    boolean overloadAuthorized = false;
    ArrayList schedule = new ArrayList();

    static String url = "jdbc:odbc:Reggie";
    static { try { Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
            catch (Exception ignored) {} }

    public static void deleteAll() throws Exception {
        Connection conn = null;

        try {
            conn = DriverManager.getConnection(url, "", "");
            Statement statement = conn.createStatement();

            statement.executeUpdate("DELETE * FROM schedule;");
        } finally {
            try { conn.close(); } catch (Exception ignored) {}
        }
```

```
}

public static Schedule create(String name) throws Exception {
    Connection conn = null;
    try {
        conn = DriverManager.getConnection(url, "", "");
        Statement statement = conn.createStatement();

        statement.executeUpdate(
           "DELETE FROM schedule WHERE name = '" + name + "';");
        return new Schedule(name);
    } finally {
        try { conn.close(); } catch (Exception ignored) {}
    }
}

public static Schedule find(String name) {
    Connection conn = null;

    try {
        conn = DriverManager.getConnection(url, "", "");
        Statement statement = conn.createStatement();
        ResultSet result = statement.executeQuery(
            "SELECT * FROM schedule WHERE Name= '" + name + "';");

        Schedule schedule = new Schedule(name);

        while (result.next()) {
            int offeringId = result.getInt("OfferingId");
            Offering offering = Offering.find(offeringId);
            schedule.add(offering);
        }

        return schedule;
    } catch (Exception ex) {
        return null;
    } finally {
        try { conn.close(); } catch (Exception ignored) {}
    }
}

public static Collection all() throws Exception {
```

```
    ArrayList result = new ArrayList();
    Connection conn = null;

    try {
        conn = DriverManager.getConnection(url, "", "");
        Statement statement = conn.createStatement();
        ResultSet results = statement.executeQuery(
            "SELECT DISTINCT Name FROM schedule;");

        while (results.next())
            result.add(Schedule.find(results.getString("Name")));
    } finally {
        try { conn.close(); } catch (Exception ignored) {}
    }

    return result;
}

public void update() throws Exception {
    Connection conn = null;

    try {
        conn = DriverManager.getConnection(url, "", "");
        Statement statement = conn.createStatement();

        statement.executeUpdate(
             "DELETE FROM schedule WHERE name = '" + name + "';");

        for (int i = 0; i < schedule.size(); i++) {
            Offering offering = (Offering) schedule.get(i);
            statement.executeUpdate(
                "INSERT INTO schedule VALUES('" + name + "','"
                + offering.getId() + "');");
        }
    } finally {
        try { conn.close(); } catch (Exception ignored) {}
    }
}

public Schedule(String name) {
    this.name = name;
}
```

```
public void add(Offering offering) {
    credits += offering.getCourse().getCredits();
    schedule.add(offering);
}

public void authorizeOverload(boolean authorized) {
    overloadAuthorized = authorized;
}

public List analysis() {
    ArrayList result = new ArrayList();

    if (credits < minCredits)
        result.add("Too few credits");

    if (credits > maxCredits && !overloadAuthorized)
        result.add("Too many credits");

    checkDuplicateCourses(result);

    checkOverlap(result);
    return result;
}

public void checkDuplicateCourses(ArrayList analysis) {
    HashSet courses = new HashSet();
    for (int i = 0; i < schedule.size(); i++) {
        Course course = ((Offering) schedule.get(i)).getCourse();
        if (courses.contains(course))
           analysis.add("Same course twice - " + course.getName());
        courses.add(course);
    }
}

public void checkOverlap(ArrayList analysis) {
  HashSet times = new HashSet();

  for (Iterator iterator = schedule.iterator();
      iterator.hasNext();)
  {
      Offering offering = (Offering) iterator.next();
```

```
            String daysTimes = offering.getDaysTimes();
            StringTokenizer tokens = new StringTokenizer(daysTimes, ",");
            while (tokens.hasMoreTokens()) {
                String dayTime = tokens.nextToken();
                if (times.contains(dayTime))
                    analysis.add("Course overlap - " + dayTime);
                times.add(dayTime);
            }
        }
    }

    public String toString() {
        return "Schedule " + name + ": " + schedule;
    }
}
```