Case Study 2 – Student Course Enrolment System

The goal of this case study is to familiarize the developers in designing a Java stand-alone application. As part of this case study, developers need to implement student course enrolment that will help the users in creating students, courses, sorting, persisting and searching the Student Data.

Business Requirements

1. The standalone program upon executing need to show the below Menu:
    1. **Create New Student Data**
    2. **Create New Course Data**
    3. **Enroll Student to a Course**
    4. **Display Fees of a Student**
    5. **Sort Student Data**
    6. **Persist Student Data**
    7. **Show All Students with courses**
    8. **Search Students**
    9. **Exit**

2. Upon entering option 1 from the Menu, the program needs to accept the below attribute values of a Student
   Student Roll no: int (automatic generation starts from 100)
   Student First Name: String (mandatory)
   Student Last Name: String (mandatory)

   Student Roll no s should be generated automatically and must get incremented by 1 each time object of Student is created.

   [Hint: Make use of 'static' keyword in Java to generate and assign roll nos sequentially]

   [Hint: Use parametrized constructors to create instances of Student in the system.

   Parameterized constructors need to take only mandatory attributes, as for rest of the

   attributes 'setter' methods need to be used]

3. Upon entering option 2 from the Menu, the program needs to accept the below attribute values of a Course
   Course Id: int
   Course Name: String
   Fees: int

   [Hint: Use parametrized constructors to create instances of Course]

4. Upon entering option 3 from the Menu, the program needs to accept the Student Roll no to whom any course would be assigned. A Student in a system needs to be associated with only one course from available Courses.
   [Hint: Make use of an appropriate 'setter' method to assign an available Course to an existing Student in the system]

5. Modify Student class to add one more attribute DOB.
   (mark as String DD/MM/YYYY format)
   Validation should be performed on the day, month and year of the date to check if it is a valid date. If validation fails, ask user to re-enter the DOB. (Date of Birth).

[Hint: Use split() of String to extract day, month and year. Use Integer.parseInt(String) to convert String value to int value.]
[Hint: Create the Utility class having static method as isValidDate(String)]

6. Modify the system so that Courses would be created either as i. ScienceCourse or ii. CommerceCourse iii. ArtsCourse.
   [Hint: Make use of Java Inheritance to derive 3 sub classes
   ScienceCourse, CommerceCourse and ArtsCourse from parent class Course]

   ScienceCourse sub-class: -
   It should have following additional attributes
   1. Lab fees (10% extra to basic fees)

   ArtsCourse sub-class: -
   It should have following additional attributes
   1. Misc. fees (5% extra to basic fees to provide materials/equipments)

7. The ScienceCourse, CommerceCourse and ArtsCourse, all would include fees, but fee calculation would vary in all 3 types of courses.
   [Hint: Mark CalculateFees method as abstract in Course
   and override in sub classes ScienceCourse and ArtsCourse].
   For ScienceCourse, fees = basic fee + Lab fees.
   For ArtsCourse, fees = basic fee + Misc. fees.
   For CommerceCourse, fees = basic fee.

8. Upon entering option 5 from the Menu, the system needs to ask User whether sorting needs to be done on Student Names or roll numbers.
   Modify the system to store all the Students along with their Courses in an appropriate collection framework, so that Sorting would be easy on Students.
   The system should support sorting of students based on
   1. Student Names
   2. Student Roll numbers.

   [Hint: Use appropriate collection framework class supporting Sorting. Make use of Comparable and/or Comparator interfaces.]

9. Upon entering option 6 from the Menu, the system needs to ask User whether persistence needs to be achieved using File System or RDMBS.
   Enhance the system so that it would be able to persist the Student data into external storage.
   The system needs to support 2 types of storages namely
   1. File System
   2. Relational Database (RDBMS)

   [Hint: Design Contract Interface for DAO layer namely Idao.]

   Provide below abstract methods in an Interface Idao.

   1. saveAllStudents
   2. retrieveAllStudents

Provide 2 concrete classes implementing an Interface Idao, namely

1. FileStorageDao :- This class would serialize and deserialize all the Student details along with Course information, except the fees, into a flat file namely – students.ser.
   [Hint: Make use of Serializable interface and transient keyword]
2. DatabaseStorageDao :- This class would persist and retrieve all the Student details along with Course information into RDBMS. E.g. MySQL.
   [Hint: Make use of JDBC API to perform CRUD operations]

10. Upon entering option 7 from the Menu, the system needs to fetch the All Student Data, along with their courses, from the Database and would be displayed on a Console.

11. Upon entering option 8 from the Menu, the User would be asked to enter the name of Student to be searched. The relevant data would be fetched from Database and displayed to the User, supporting the Search mechanism.

    [Hint: If no record found matching student to be searched in the database, then Custom Exception namely – StudentNotFoundException needs to be thrown and appropriate Error Message needs to be displayed to the End User.]


Softra Services