JAMHURIYA UNIVERSITY
of
SCIENCE & TECHNOLOGY

CA & SCNS Departments

Java Programming

Course Project

CC Tahlil

2024-06-20

Objective

The objective of this course project is to implement a basic Java program to manage a `class room`, its subjects, and students. The program should use classes and objects, arrays, loops, control flow, mathematical operations, and all primitive data types wherever possible.

Requirements

1. **Classes and Objects**:
    - Create classes for `Fasal`, `Student`, and `Subject`.
    - A `Fasal` should have a name, an array of `Subject` objects, and an array of `Student` objects.
    - A `Subject` should have a name and credit hours.
    - A `Student` should have an ID, name, age, an array of `Subject` objects, and an array of marks corresponding to each subject.
2. **Arrays**:
    - Use arrays to store multiple `Subject` objects and multiple `Student` objects in a `Fasal`.
    - Use arrays to store marks for subjects taken by each student.
3. **Loops**:
    - Use loops to iterate over arrays to perform operations such as displaying details, calculating total credit hours, GPA, and finding a specific student or subject.
4. **Control Flow**:
    - Use control flow statements (`if-else`, `switch`) to handle different conditions, such as checking if a student is enrolled in a particular subject.
5. **Mathematical Operations**:
    - Perform mathematical operations to calculate total credit hours, GPA, and grades.
6. **Primitive Data Types**:
    - Use all primitive data types (`byte`, `short`, `int`, `long`, `float`, `double`, `char`, `boolean`) appropriately in the program.

Project Tasks

1. Define the `Subject` Class

- Create a class named `Subject`.
- Define the following attributes:
  - `String name`
  - `int creditHours`
- Add a method to determine the grade based on marks.

2. Define the `Student` Class

- Create a class named `Student`.
- Define the following attributes:
  - `int id`
  - `String name`
  - `int age`
  - `Subject[]` subjects (array to store subjects)
  - `int[]` marks (array to store marks for each subject)
- The `Student` class should accept subjects and corresponding marks as parameters.

3. Define the `Fasal` Class

- Create a class named `Fasal`.
- Define the following attributes:
  - `String name`
  - `Subject[]` subjects
  - `Student[]` students
- Implement methods to:
  - Display Fasal details.
  - Find a student by ID.
  - Calculate average GPA of the Fasal.

4. Implement the Main Class

- Create a `Main` class with a `main` method.
- Create instances of `Subject`, `Student`, and `Fasal` classes.
- Use arrays to store multiple subjects and students.
- Use loops to iterate through subjects and students.
- Use control flow to display details, calculate GPAs, and find specific students.

Testing

- Compile and run the program to ensure it executes correctly and displays the expected output.
- Test with different sets of Fasal, subject, and student data to verify the accuracy of operations and control flow.

Submission

- Submit the Java source files (`Subject.java`, `Student.java`, `Fasal.java`, `Main.java`) in a zip file.
- Include a brief report describing the implementation, challenges faced, and any assumptions made. It should be a **PDF** file.
- The deadline of the project is 2024-06-27 $\rightarrow$ 2024-07-18.
- Each file of the program should contain student information.

Additional Notes

- Ensure the code is well-commented for readability.
- Follow Java naming conventions and best practices for coding.
- Handle any potential edge cases, such as empty arrays, appropriately.

APPLICATION PSEUDOCODE

Subject Class

```
Class Name: Subject

Attributes:
name: String
creditHours: int

Methods:
Constructor(name: String, creditHours: int):
 this.name = name
 this.creditHours = creditHours

Method determineGrade(marks: int) -> char:
 If marks >= 90:
   return 'A'
 Else if marks >= 80:
   return 'B'
 Else if marks >= 70:
   return 'C'
 Else if marks >= 60:
   return 'D'
 Else:
   return 'F'
```

Student Class

```
Class Name: Student

Attributes:
id: int
name: String
age: int
subjects: Array of Subject
marks: Array of int

Methods:
Constructor(id: int, name: String, age: int,
      subjects: Array of Subject, marks: Array of int):
 this.id = id
 this.name = name
 this.age = age
 this.subjects = subjects
 this.marks = marks

Method calculateTotalCreditHours() -> int:
  totalCreditHours = 0
  For each subject in subjects:
    totalCreditHours += subject.creditHours
  return totalCreditHours

Method calculateGPA() -> double:
 totalCreditHours = calculateTotalCreditHours()
 totalQualityPoints = 0
 For i from 0 to length of subjects:
  gradePoints = 0
  grade = subjects[i].determineGrade(marks[i])
  Switch grade:
   Case 'A':
    gradePoints = 4
   Case 'B':
    gradePoints = 3
   Case 'C':
    gradePoints = 2
   Case 'D':
    gradePoints = 1
   Case 'F':
    gradePoints = 0
  totalQualityPoints += gradePoints * subjects[i].creditHours
 return totalQualityPoints / totalCreditHours
```

Fasal Class

```
Class Name: Fasal

Attributes:
name: String
subjects: Array of Subject
students: Array of Student

Methods:
Constructor(name: String, subjects: Array of Subject, students: Array of Student):
 this.name = name
 this.subjects = subjects
 this.students = students

Method displayFasalDetails():
 Print "Fasal Name: " + name
 Print "Subjects:"
 For each subject in subjects:
  Print "- " + subject.name + " (" + subject.creditHours + " credit hours)"
 Print "Students:"
 For each student in students:
  Print "- " + student.name + " (ID: " + student.id + ", Age: " + student.age + ")"

Method findStudentById(id: int) -> Student:
 For each student in students:
  If student.id == id:
   return student
  return null

Method calculateAverageGPA() -> double:
 totalGPA = 0
 For each student in students:
  totalGPA += student.calculateGPA()
 return totalGPA / length of students
```

Main Class

```
Class Name: Main

Method main():
// Create subjects
math = new Subject("Math", 3)
science = new Subject("Science", 4)
history = new Subject("History", 2)

// Create an array of subjects
subjects = [math, science, history]

// Create student marks
std1Marks = [85, 92, 78]
std2Marks = [90, 88, 70]

// Create students
student1 = new Student(1, "Filsan", 20, subjects, std1Marks)
student2 = new Student(2, "Calas", 21, subjects, std2Marks)

// Create an array of students
students = [student1, student2]

// Create a Fasal
myFasal = new Fasal("CS101", subjects, students)

// Display Fasal details
myFasal.displayFasalDetails()

// Find and display a specific student by ID
searchId = 2
foundStudent = myFasal.findStudentById(searchId)
If foundStudent is not null:
 Print "Found Student: " + foundStudent.name + " (ID: " + foundStudent.id + ")"
 Print "Total Credit Hours: " + foundStudent.calculateTotalCreditHours()
 Print "GPA: " + foundStudent.calculateGPA()
Else:
 Print "Student with ID " + searchId + " not found."

// Calculate and display average GPA of the Fasal
averageGPA = myFasal.calculateAverageGPA()
Print "Average GPA of the Fasal: " + averageGPA
```