**University Management System**

The University Management System is a C++ project that models the various entities within a university, including departments, courses, professors, students, and the university itself. This project aims to provide a comprehensive structure for managing university-related information and interactions.

**Class: Department**

The Department class represents a department within the university. Each department has a unique name, a list of course codes offered by the department, and a list of professors associated with the department.

* Attributes:
  + - **departmentName**: A string representing the name of the department.
    - **courseCodes**: A vector of strings containing the course codes offered by the department.
    - **professors**: A vector of pointers to Professor objects associated with the department.
  + Methods:
    - **Department**(): Constructor to initialize the department's attributes.
    - ~**Department**(): Destructor to clean up resources.

**Class: Course**

The Course class represents an individual course offered by the university. Each course has a course code, a course name, a professor teaching the course, the department offering the course, and a list of students enrolled in the course.

* Attributes:
  + **courseCode**: A string representing the course code.
  + **courseName**: A string representing the name of the course.
  + **professorTeaching**: A pointer to the Professor object teaching the course.
  + **courseDepartment**: A reference to the Department offering the course.
  + **studentsEnrolled**: A vector of pointers to Student objects enrolled in the course.
* Methods:
  + **Course**(): Constructor to initialize the course's attributes.
  + ~**Course**(): Destructor to clean up resources.

**Class: Person**

The Person class serves as the base class for both professors and students. It encapsulates common attributes and methods related to personal information.

* Attributes:
  + **name**: A string representing the person's name.
  + **age**: An integer representing the person's age.
  + **gender**: A character representing the person's gender.
  + **email**: A string representing the person's email address.
  + **phoneNumber**: A string representing the person's phone number.
* Methods:
  + **Person**(): Default constructor.
  + **Person**(string name, int age, char gender, string email, string phoneNumber): Parameterized constructor to initialize the person's attributes.
  + **virtual void getPersonInfo**() = 0;: A pure virtual method to be overridden by subclasses for displaying personal information.
  + Getter and setter methods for each attribute.
  + ~**Person**(): Destructor to clean up resources.

**Class: Professor**

The Professor class represents a professor at the university. It inherits from the Person class and adds a professor ID attribute.

* Attributes:
  + **ProfessorId**: An integer representing the professor's unique ID.
* Methods:
  + **Professor**(): Constructor to initialize the professor's attributes.
  + ~**Professor**(): Destructor to clean up resources.

**Class: Student**

The Student class represents a student at the university. Similar to the Professor class, it inherits from the Person class and adds attributes specific to students, such as a student ID and GPA.

* Attributes:
  + studentId: An integer representing the student's unique ID.
  + gpa: A double representing the student's GPA.
* Methods:
  + Student(): Constructor to initialize the student's attributes.
  + ~Student(): Destructor to clean up resources.

**Class: University**

The University class represents the entire university and manages its departments and students.

* Attributes:
  + **name**: A string representing the name of the university.
  + **departments**: A vector of pointers to Department objects associated with the university.
  + **students**: A vector of pointers to Student objects enrolled in the university.
* Methods:
  + **University**(): Constructor to initialize the university's attributes.
  + ~**University**(): Destructor to clean up resources.

This project provides a foundation for managing university-related data and relationships among various entities. It demonstrates object-oriented programming principles, inheritance, polymorphism, and encapsulation, making it a comprehensive example of a C++ application. Additional functionality can be added to perform operations such as enrolling students in courses, assigning professors to courses, and managing university resources.