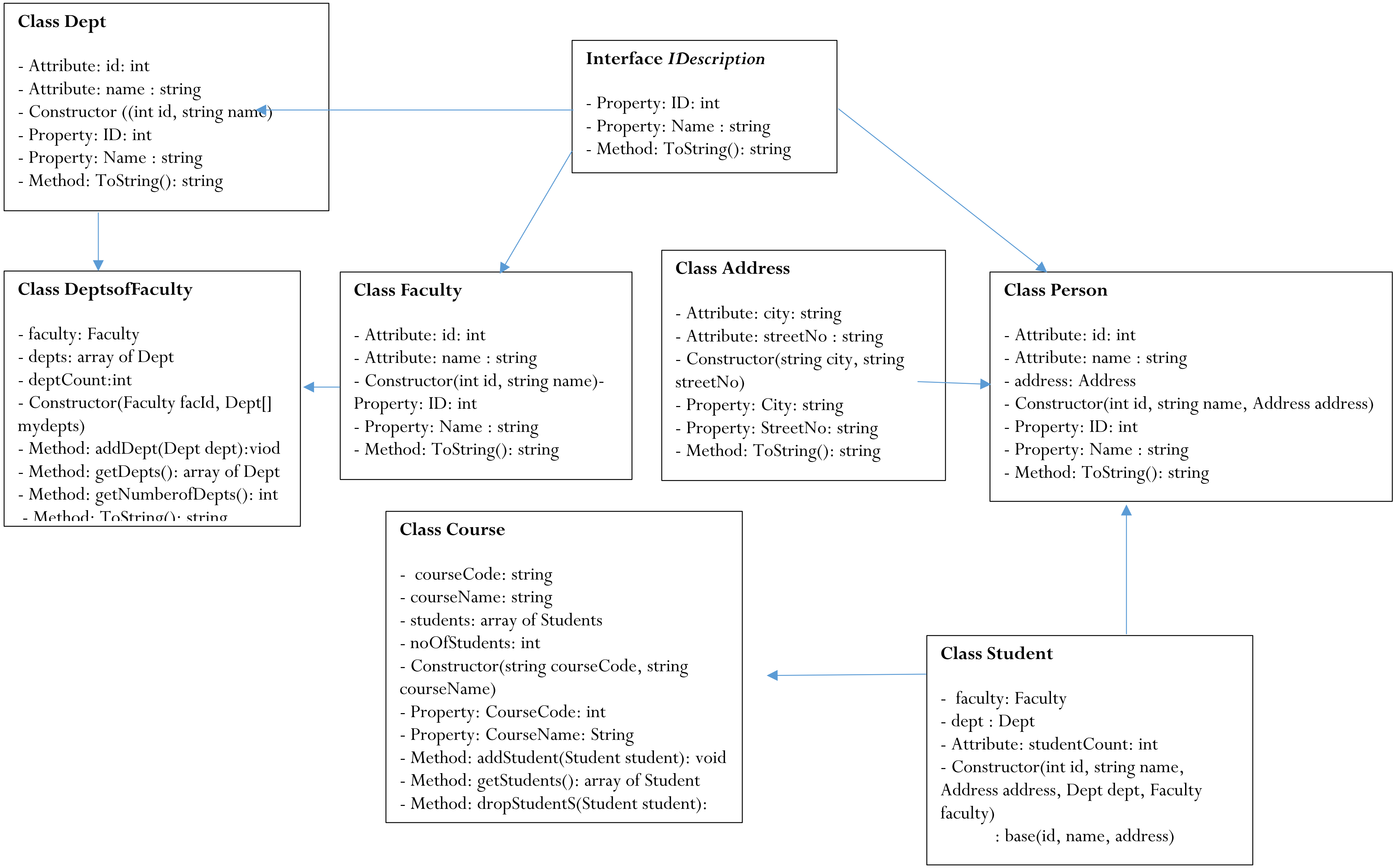
INFORMAL CONCEPTUAL VIEW FOR THE COURSE REGISTRATION PROJECT



Class Dept

- Attribute: id: int
- Attribute: name : string
- Constructor ((int id, string name)
- Property: ID: int
- Property: Name : string
- Method: ToString(): string

Interface *IDescription*

- Property: ID: int
- Property: Name : string
- Method: ToString(): string

Class DeptsofFaculty

- faculty: Faculty
- depts: array of Dept
- deptCount:int
- Constructor(Faculty facId, Dept[] mydepts)
- Method: addDept(Dept dept):viod
- Method: getDepts(): array of Dept
- Method: getNumberofDepts(): int
- Method: ToString(): string

Class Faculty

- Attribute: id: int
- Attribute: name : string
- Constructor(int id, string name)- Property: ID: int
- Property: Name : string
- Method: ToString(): string

Class Address

- Attribute: city: string
- Attribute: streetNo : string
- Constructor(string city, string streetNo)
- Property: City: string
- Property: StreetNo: string
- Method: ToString(): string

Class Person

- Attribute: id: int
- Attribute: name : string
- address: Address
- Constructor(int id, string name, Address address)
- Property: ID: int
- Property: Name : string
- Method: ToString(): string

Class Course

- courseCode: string
- courseName: string
- students: array of Students
- noOfStudents: int
- Constructor(string courseCode, string courseName)
- Property: CourseCode: int
- Property: CourseName: String
- Method: addStudent(Student student): void
- Method: getStudents(): array of Student
- Method: dropStudentS(Student student):

Class Student

- faculty: Faculty
- dept : Dept
- Attribute: studentCount: int
- Constructor(int id, string name, Address address, Dept dept, Faculty faculty)
  : base(id, name, address)

INFORMAL CONCEPTUAL DESCRIPTION FOR THE COURSE REGISTRATION PROJECT:

- There are many faculties (array of objects of the class **Faculty**) and each faculty has an id and a name.
- Each department (class **Dept**) has an id and a name.
- Each faculty has five departments, at maximum (array of objects of the class **Dept**).
- In order to relate each faculty with its departments a class called (**DeptsofFaculty**) is being added. The class consists of a reference to **Faculty** and an array of objects of the class **Dept**.
- For each person (class **Person**), we store his/her id and name. In addition, an object of the class **Address** is also stored (only one address is to be stored for each person).
- Each **Address** consists of city name and street number.
- Each **Student** is a **Person** (inherits from Person to store id and name). Additionally, we store the faculty and the department to which the student belongs. We can store students as many as we need. We keep track of the total number of students at any time.
- Each **course** has a course code, a name and many students that may register for the course. It is possible for any student to drop the course. We keep track of total number of students that are registered for each course.

COURSE REGISTRATION PROJECT:

Write the necessary C# code to fulfill the following system:

A) A **interface** called **IDescription** that obligates the classes that implement it, to have the followings:
- A property called `ID` (type int) with both set and get.
- A property called `Name` (type string) with both set and get.
- A method called `ToString`() (returns string)

B) A **class** called **Dept** describes the department to which the student belongs. The class implements **IDescription** interface and has the following members:
- A private member `id` (type int) to store an integer that represents the department id.
- A private member `name` (type string) to store the department name.
- A constructor to initialize the instances variables using their corresponding properties
- Two properties to initialize the instance members.
- An override method called `ToString()` (returns string) that will be used to print `id` and the `name` (use properties)

C) A **class** called **Faculty** describes the faculty to which the student belongs. The class implements **IDescription** interface and has the following members:
- A private member `id` (type int) to store an integer that represents the faculty id.
- A private member `name` (type string) to store the faculty name.
- A constructor to initialize the instances variables using their corresponding properties
- Two properties to initialize the instance members.
- An override method called `ToString()` (returns string) that will be used to print `id` and `name` (use properties)

D) A **class** called **DeptOfFaculty** describes the a particular faculty with its departments. The class has the following members:
- A private reference member called `faculty` (type Faculty) to store a complete faculty object.

- A private array called `depts` (type Dept) to store five departments, at most, for each faculty.
- A private non-static member `deptCounts` (type int) with a default value 0, to store the total number of departments in each faculty.
- A constructor to initialize the `faculty` and `depts` array.
- A public method called `getNumberofDepts()` (returns int) to return no. of departments.
- A public method called `getDepts` (returns Dept[]) to return departments.
- A public method called `addDept` (recieves dept of type Dept) and returns void) to allow adding a department.
- An override method called `ToString()` (returns string) that will be used to print faculty name with its departments' names.

E) A **class** called **Address** describes the person address with the following members:

- A private member `city` (type string) to describe the city.
- private member `streetNo` (type string).
- A `constructor` to initialize the instances variables using their corresponding properties
- `Two properties` to initialize the instance members.
- An override method called `ToString()` (returns string) that will be used to print `city` and `streetNo` (use properties)

F) A **class** called **Person** that describes any person. The class implements **IDescription** interface and has the following members:

- A private member `id` (type int) to store a person id.
- A private member `name` (type string) to store person name.
- A private reference member called `address` (type Address) to store a complete address object.
- A constructor to initialize the instances variables (id, name and address)
- Two properties to initialize the instance members.
- An override method called `ToString()` (returns string) that will be used to print `id` and the `name` (use properties) plus `city` and `streetNo` of address.

G) A **class** called **Student** that describes the student. The class inherits from **Person** and has the following members:

- A private reference member called `faculty` (type Faculty) to store a complete faculty object of the student.
- A private reference member called `dept` (type Dept) to store a complete department object of the student.
- A private static member `studentCount` (type int) with a default value 0. The member is used to store the total number of student objects created.
- A constructor to initialize the instances variables (`faculty` and `dept`). The constructor also must increase the static `studentCount` by 1.
- A public and static property with only get for the static `studentCount`.
- An override method called `ToString()` (returns string) that will be used to print `id` and the `name` of the student (use properties) plus his faculty name and department name.

H) A **class** called **Course** that describes the course and how students are enrolled in. The class has the following members:

- A private member `courseCode` (type string) to describe the city.
- private member `courseName` (type string).
- A private array called `students` (type Student) to store 15 students, at most, for each course.
- A private non-static member `numberOfStudents` (type int) with a default value 0, to store the total number of students who registered for a specific course.
- A constructor to initialize the instances variables (`courseCode` and `courseName`).
- Two properties to initialize the instance members `courseCode` and `courseName`.
- A public method called `getNumberofStudents()` (returns int) to return no. of students who registered so far in the course.
- A public method called `getStudents` (returns Student[]) to return students.
- A public method called `addStudent` (receives student of type Student and returns void) to allow adding a student to the array `students`.
- An override method called `ToString()` (returns string) that will be used to print `courseCode` and `courseName` with the name of the students who registered for the course.

After creating the above classes, add your faculty data with its departments. Then add three students (array of objects) and only one course. From the three students, let two of them to register for the course. Following this add a new department and let the third student to register also for the course you created.