BUỔI 06
OOP: TÍNH ĐÓNG GÓI

MỤC TIÊU

- Hiểu được tính Đóng gói trong Lập trình hướng đối tượng
- Nắm vững cú pháp tạo và định nghĩa class trong C#

Bài tập 1. Class **Student**

Xây dựng class **Student** để biểu diễn các đối tượng sinh viên theo bản mô tả sau:

Student
– studentId: string – name: string – birthYear: int – gender: bool – stdClass: string
+ Student() + Student(string studentId, string name, int birthYear, bool gender, string stdClass) + GetStudentId(): string + SetStudentId (string newId): void + GetName():string + SetName(string newName): void + GetBirthYear(): int + SetBirthYear(int newbirthYear): void + GetGender(): bool + SetGender(bool newGender): void + GetStdClass(): string + SetStdClass(int newStdClass): void + Input(): void + GetAge(): int + PrintInfo(): void

- Attributes:
  - studentId: string
  - name: string
  - birthYear: int
  - gender: bool
  - stdClass: string
- Constructors:
  - Default constructor tạo một sinh viên không có các thông tin ban đầu: Student()
  - Khởi tạo một sinh viên với đầy đủ các thông tin ban đầu *(*birthYear *cần được kiểm tra tính hợp lệ* 1900 ≤birthYear ≤*năm hiện tại, nếu không hợp lệ thì thông báo và set về giá trị mặc định là 1900)*:
    Student(string studentId, string name, int birthYear, bool gender, string stdClass)

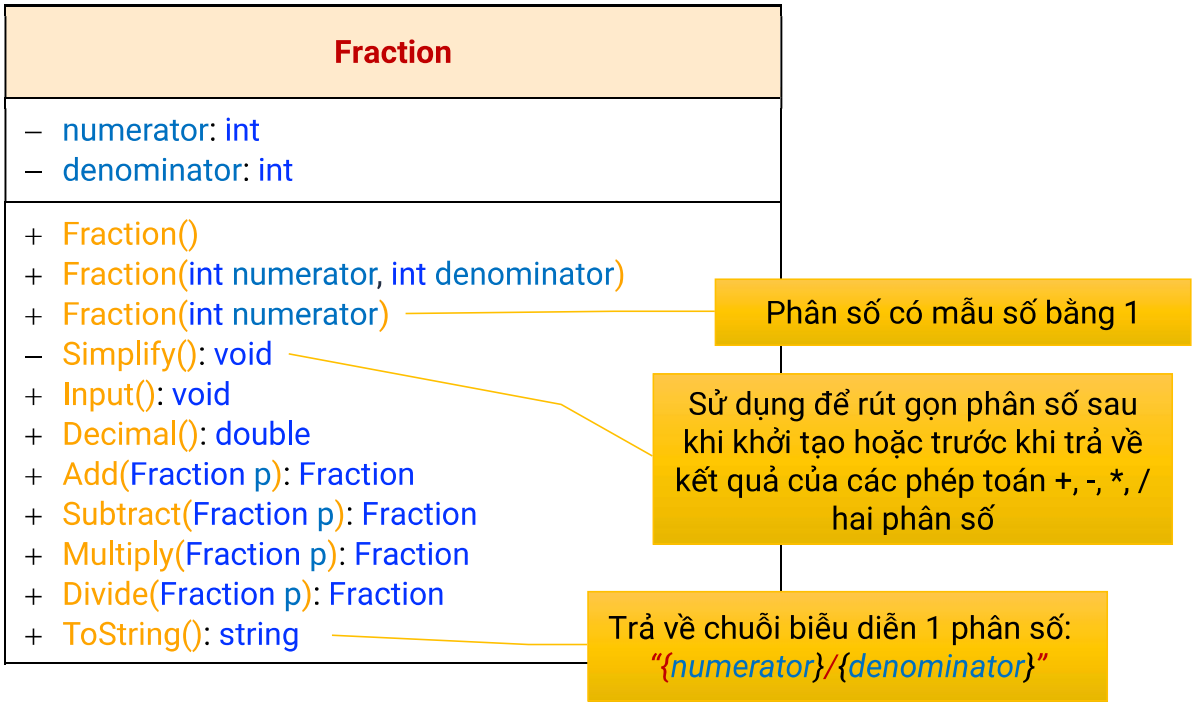
- Methods:
  - Các phương thức Get và Set để truy xuất và thiết lập giá trị cho các thuộc tính của sinh viên. Đối với phương thức Set thuộc tính năm sinh cần được kiểm tra tính hợp lệ *1900* ≤ birthYear ≤ *năm hiện tại*, nếu không hợp lệ thì thông báo và set về giá trị mặc định là 1900)
  - Input(): Thực hiện nhập các thông tin cho sinh viên. Với năm sinh yêu cầu người dùng nhập cho đến thỏa điều kiện *1900* ≤ birthYear ≤ *năm hiện tại*.
  - GetAge(): Trả về tuổi hiện tại của sinh viên.
  - PrintInfo(): In ra tất cả thông tin của sinh viên:

```
Student ID: { studentId }
Name: { name }
Birth Year: { birthYear }
Gender: {Nam/Nữ}
Class: {stdClass}
```

Viết hàm Main để khởi tạo các đối tượng sinh viên và thực hiện các phương thức đã xây dựng.

Bài tập 2. Class **Fraction**

Xây dựng class **Fraction** để biểu diễn phân số với các thuộc tính numerator (tử số) và denominator (mẫu số) nguyên.



- Attributes:
  - numerator: int
  - denominator: int
- Properties:
  - Numerator: bao gồm thao tác get truy xuất tử số và set để thiết lập giá trị cho tử số
  - Denominator: bao gồm thao tác get truy xuất mẫu số và set để thiết lập giá trị cho mẫu số (kiểm tra nếu giá trị mẫu số = 0 thì thông báo cho người dùng và set giá trị mặc định cho mẫu số 1)
- Constructors:
  - Default constructor tạo phân số có tử số = 0, mẫu số =1: Fraction()
  - Khởi tạo phân số với giá trị tử số và mẫu số cho trước *(giá trị mẫu số được kiểm tra như phía trên)*:

    Fraction(int numerator, int denominator)
  - Khởi tạo phân số với giá trị tử số cho trước, mẫu số = 1:

    Fraction(int numerator)

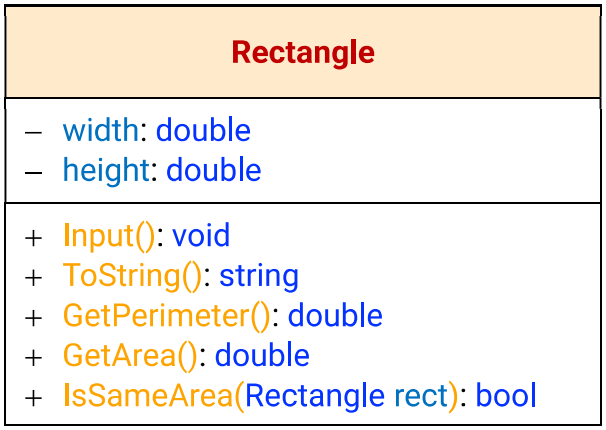
  Yêu cầu: Các phân số sau khi được khởi tạo đều phải được rút gọn
- Methods:
  - Simplify(): Tối giản phân số
  - Input(): Thực hiện nhập tử số và mẫu số cho phân số. Với mẫu số yêu cầu người dùng nhập cho đến khi mẫu số khác 0. Phân số sau khi nhập xong sẽ được rút gọn.
  - Decimal(): Trả về giá trị thập phân của phân số
  - Add(Fraction p): Trả về một phân số là tổng của đối tượng phân số hiện hành với một phân số p. Phân số kết quả phải được rút gọn trước khi trả về.
  - Subtract(Fraction p): Trả về một phân số là hiệu của đối tượng phân số hiện hành với một phân số p. Phân số kết quả phải được rút gọn trước khi trả về.
  - Multiply(Fraction p): Trả về một phân số là tích của đối tượng phân số hiện hành với một phân số p. Phân số kết quả phải được rút gọn trước khi trả về.
  - Divide(Fraction p): Trả về một phân số là thương của đối tượng phân số hiện hành với một phân số p. Phân số kết quả phải được rút gọn trước khi trả về.
  - ToString(): Trả về chuỗi biểu diễn phân số dưới dạng: *"numerator/denominator"*

Viết hàm Main để khởi tạo các đối tượng phân số và thực hiện các phép toán trên các đối tượng này.

Bài tập 3. Class **Rectangle**

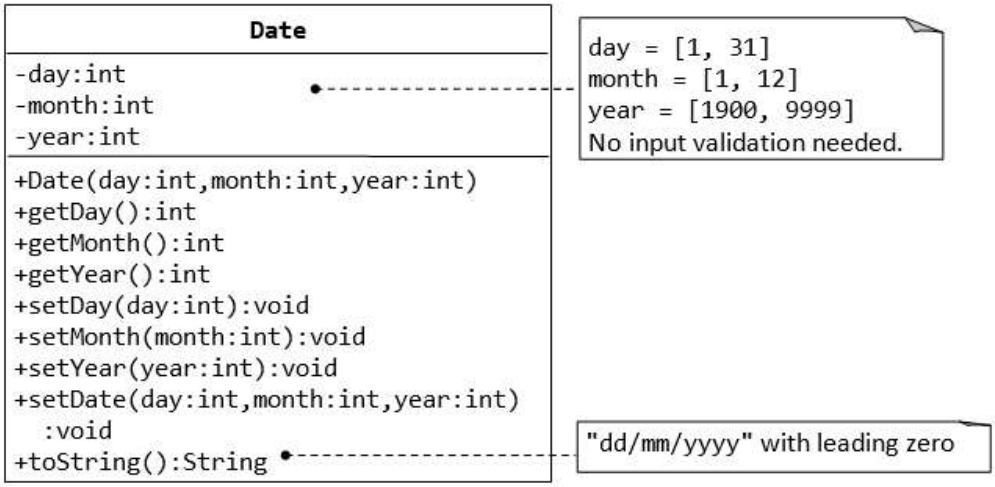
Xây dựng class **Rectangle** để biểu diễn các đối tượng hình chữ nhật theo bản mô tả sau:

(Sinh viên suy nghĩa các ràng buộc và các thuộc tính/phương thức cần thiết để mô tả được đối tượng đúng với thực tế. Ví dụ: chiều dài, chiều rộng của hình chữ nhật phải lơn hơn 0)



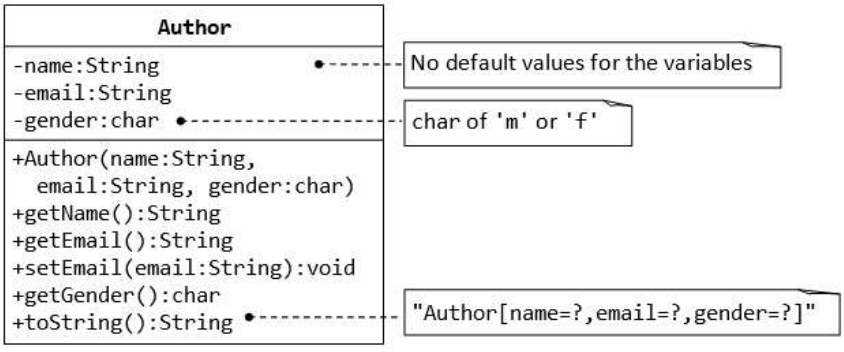
Bài tập 4. Class **Date**

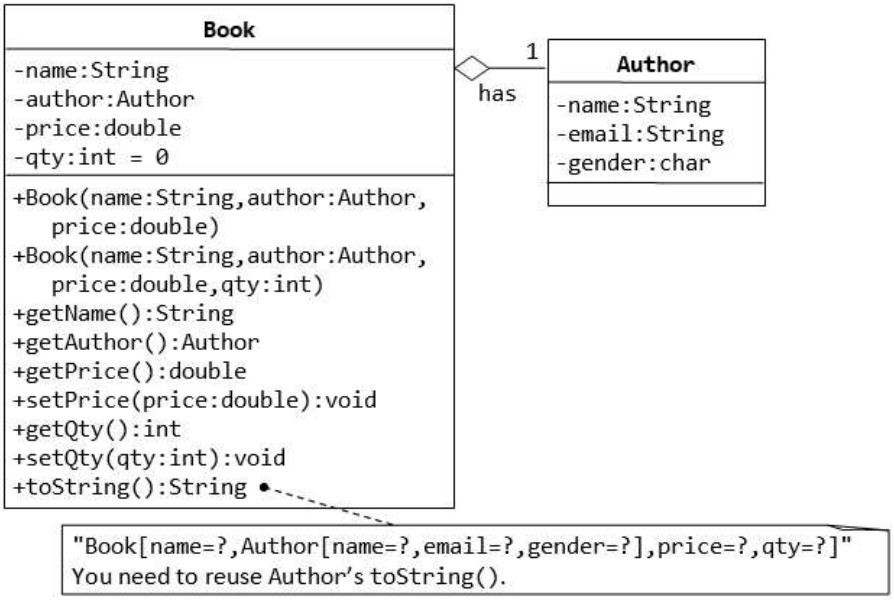
Xây dựng class **Date** để biểu diễn ngày tháng theo bản mô tả sau:



Bài tập 5. Class **Book**

Xây dựng class **Author** để biểu diễn tác giả và class **Book** để biểu diễn quyển sách theo mô tả sau:




Book
-name:String
-author:Author
-price:double
-qty:int = 0
+Book(name:String,author:Author,
price:double)
+Book(name:String,author:Author,
price:double,qty:int)
+getName():String
+getAuthor():Author
+getPrice():double
+setPrice(price:double):void
+getQty():int
+setQty(qty:int):void
+toString():String
1
has
Author
-name:String
-email:String
-gender:char
"Book[name=?,Author[name=?,email=?,gender=?],price=?,qty=?]"
You need to reuse Author's toString().