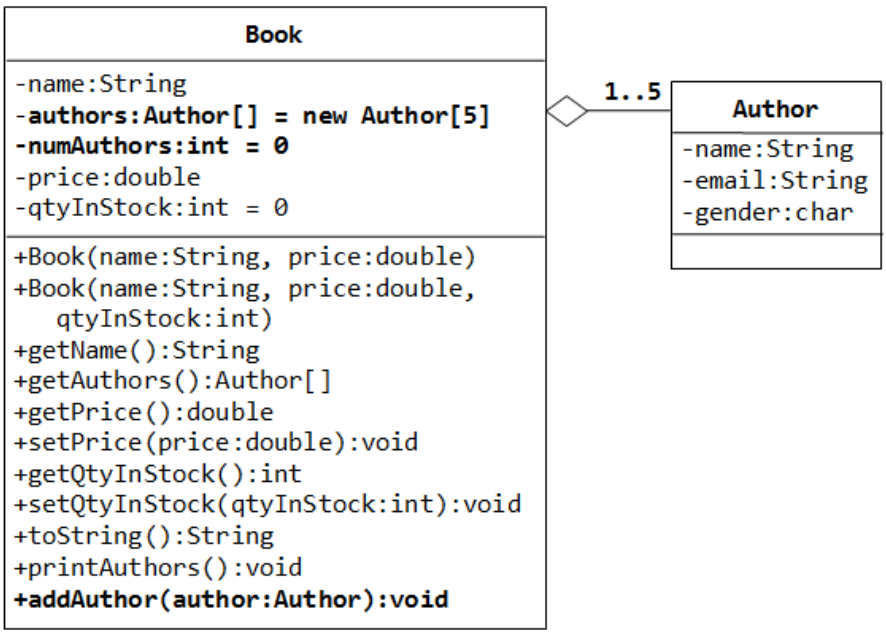
Week 4

Exercise 4.1: Author and Book Classes Once More - A Fixed-Length Array of Objects as an Instance Variable



In the above exercise, the number of authors cannot be changed once a Book instance is constructed. Suppose that we wish to allow the user to add more authors (which is really unusual but presented here for academic purpose).

We shall remove the `authors` from the constructors, and add a new method called `addAuthor()` to add the given `Author` instance to this Book.

We also need to pre-allocate an `Author` array, with a fixed length (says 5 - a book is written by 1 to 5 authors), and use another instance variable `numAuthors` (`int`) to keep track of the actual number of authors.

You are required to:

1. Modify your Book class to support this new requirement.
2. Try writing a method called `removeAuthorByName(authorName)`, that remove the author from this Book instance if `authorName` is present. The method shall return `true` if it succeeds.

```
boolean removeAuthorByName(String authorName)
```

Advanced Note: Instead of using a fixed-length array in this case, it is better to be a dynamically allocated array (e.g., `ArrayList`), which does not have a fixed length.

Exercise 4.2: MyPolynomial Class

A class called `MyPolynomial`, which models polynomials of degree-*n* (see equation), is designed as shown in the class diagram.

$$c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0$$

The class contains:

- An instance variable named `coefficients`, which stores the coefficients of the n-degree polynomial in a `double` array of size n+1, where c_0 is kept at index 0.
- A constructor `MyPolynomial(coeffs:double...)` that takes a variable number of doubles to initialize the coeffs array, where the first argument corresponds to c_0. The three dots is known as *varargs*(variable number of arguments), which is a new feature introduced in JDK 1.5. It accepts an array or a sequence of comma-separated arguments. The compiler automatically packs the comma-separated arguments in an array. The three dots can only be used for the last argument of the method.
  Hints:

```
public class MyPolynomial {

   private double[] coefficients;

   public MyPolynomial(double... coeffs) {  // varargs
      this.coeffs = coeffs;                 // varargs is treated as array
   }
   ......
}

// Test program
// Can invoke with a variable number of arguments
MyPolynomial p1 = new MyPolynomial(1.1, 2.2, 3.3);
MyPolynomial p1 = new MyPolynomial(1.1, 2.2, 3.3, 4.4, 5.5);
// Can also invoke with an array
Double coefficients = {1.2, 3.4, 5.6, 7.8}
MyPolynomial p2 = new MyPolynomial(coefficients);
```

- Another constructor that takes coefficients from a file (of the given `filename`), having this format:

```
Degree-n(int)
c0(double)
c1(double)
......
......
cn-1(double)
cn(double)
(end-of-file)
```

Hints:

```
public MyPolynomial(String filename) {
   Scanner in = null;
   try {
      in = new Scanner(new File(filename));  // open file
   } catch (FileNotFoundException e) {
      e.printStackTrace();
   }
   int degree = in.nextInt();      // read the degree
   coefficients = new double[degree+1];  // allocate the array
   for (int i=0; i<coefficients.length; ++i) {
      coeffs[i] = in.nextDouble();
   }
}
```

- A method `getDegree()` that returns the degree of this polynomial.
- A method `toString()` that returns "c_nx^n + c_{n-1}x^(n-1) + ... + c_1x + c_0".
- A method `evaluate(double x)` that evaluate the polynomial for the given x, by substituting the given x into the polynomial expression.
- Methods `add()` and `multiply()` that adds and multiplies this polynomial with the given `MyPolynomial` instance another, and returns a new `MyPolynomial` instance that contains the result.

Write the `MyPolynomial` class. Also write a unit test (called `MyPolynomialTest`) to test all the methods defined in the class.