# OO Programming Sem 3 2015\_16 CA #1

Please Note: The document, as it stands, constitutes a portion of the CA. Further details will be added soon. A marking scheme will be made available when the full CA is disseminated.

Due: End of week 6 (detail on the submission process to follow).

NOTES:

Note 1: The work logs are a vital component of the overall project. They must be filled in on a week-by-week basis. A template document is available in the Resources sub-folder.

Note 2: Consistent indentation and coding conventions required (e.g. all class names to start with uppercase letter). I have highlighted some of the pertinent conventions in the document in folder “Coding Style Guidelines”.

Note 3: **JavaDoc comments and generated html files required**. I will briefly explain how to do this shortly.

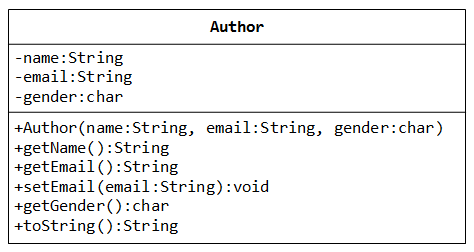
Note 4: A cover sheet must be filled in and signed to complete your submission.

Document Status (29/09/2015):

First three parts specified. Please note that the relationship between the Book and Author classes is called “Composition”, which simply means that the instance fields of an object – e.g. a Book object – may themselves be objects (an Author object). We’ve already seen this many times, since Strings are objects and lots of classes have had String instance fields.

## Part 1: The Author and Book Classes

Note: you should store your classes and testers in a folder called Part1.  
**For subsequent parts you should also have separate folders** (Part2, etc). Some of the classes you create in one part may be copied into another (and perhaps modified for the purposes of that part).



A class called Author is designed as shown in the class diagram. It contains:

* Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');
* One constructor to initialize the name, email and gender with the given values;

public Author (String name, String email, char gender) {......}

(There is no default constructor for Author, as there are no defaults for name, email and gender.)

* public getters/setters: getName(), getEmail(), setEmail(), and getGender();  
  (There are no setters for name and gender, as these attributes cannot be changed.)
* A toString() method that returns "*author-name (gender) at email*", e.g., "Dermot Hegarty (m) at dhegarty@somewhere.com".

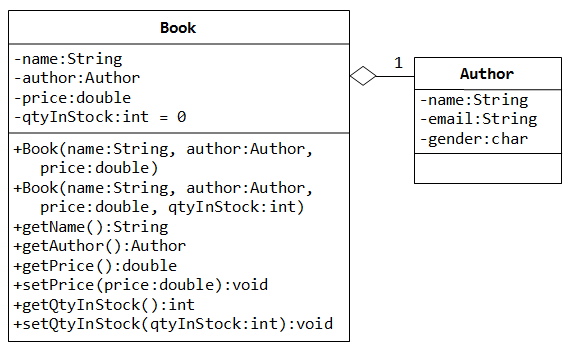
Write the Author class. Also write a *test program* called TestAuthor to test the constructor and public methods. Among other things, try changing the email of an author, e.g.,

Author anAuthor = new Author("Dermot Hegarty", "dhegarty@somewhere.com", 'm');

System.out.println(anAuthor); // calls toString()

anAuthor.setEmail("hegarty\_d@hotmail.com")

System.out.println(anAuthor);



A class called Book is designed as shown in the class diagram. It contains:

* Four private instance variables: name (String), author (of the class Author you have just created, assume that each book has one and only one author), price (double), and qtyInStock (int);
* Two constructors:
* public Book (String name, Author author, double price) {...}
* public Book (String name, Author author, double price, int qtyInStock) {...}
* public methods   
  getName(), getAuthor(), getPrice(), setPrice(), getQtyInStock(), setQtyInStock().
* toString() that returns "*'book-name' by author-name (gender) at email*".  
  (Take note that the Author's toString() method returns "*author-name (gender) at email*".)

Write the class Book (which uses the Author class written earlier). Also write a test program called TestBook to test the constructor and public methods in the class Book. Take note that you have to construct an instance of Author before you can construct an instance of Book. E.g.,

Author anAuthor = new Author(......);

Book aBook = new Book("Java for dummy", anAuthor, 19.95, 1000);

//Or use an anonymous instance of Author –NOTE:you must become familiar with this!

Book anotherBook = new Book("more Java for dummy", new Author(......), 29.95, 888);

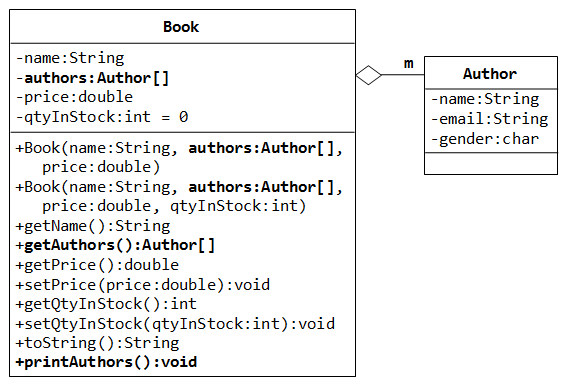
Take note that both Book and Author classes have a variable called name. However, it can be differentiated via the referencing instance. For a Book instance (let’s call it aBook), aBook.name refers to the name of the book; whereas for an Author's instance (let’s call it anAuthor), anAuthor.name refers to the name of the author. There is no need (and not recommended) to call the variables bookName and authorName.

TRY:

1. Printing the name and email of the author from a Book instance. (Hint: aBook.getAuthor().getName(), aBook.getAuthor().getEmail()).
2. Introduce new methods (and test, of course) called getAuthorName(), getAuthorEmail(), getAuthorGender() in the Book class to return the name, email and gender of the author of the book. For example,

public String getAuthorName() { ...... }

**Part 2: Book and Author Classes Again - An Array of Objects as an Instance Variable**



In the previous part the assumption was that a book is written by one and only one author. In reality, a book can be written by one or more author. Modify the Book class to support one or more authors by changing the instance variable authors to an Author array. Reuse the Author class written earlier.

Notes:

* The constructors take an array of Author (i.e., Author[]), instead of an Author instance.
* The toString() method shall return "book-name by *n* authors", where *n* is the number of authors.
* A new method printAuthors() to print the names of all the authors.

You are required to:

1. Write the code for the Book class. You shall re-use the Author class written earlier.
2. Write a test program (called TestBook) to test the Book class. Note: Your test program should make sure that you test the getAuthors() method.

Hints:

// Declare and allocate an array of Authors

Author[] authors = new Author[2];

authors[0] = new Author("Jim Jones", "jjones@somewhere.com", 'm');

authors[1] = new Author("Mike Conroy", "mconroy@nowhere.com", 'm');

// Declare and allocate a Book instance

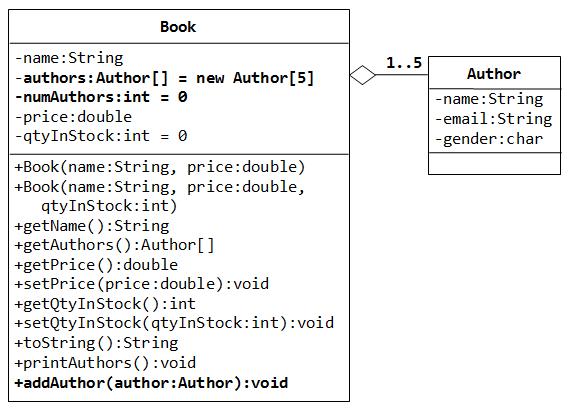
Book javaDummy = new Book("Java for Dummy", authors, 19.99, 99);

System.out.println(javaDummy); // toString()

System.out.print("The authors are: ");

javaDummy.printAuthors();

**Part 3: Book and Author 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.  
   Hints:
2. public class Book {
3. // private instance variable
4. private Author[] authors = new Author[5]; // declare and allocate the array
5. // BUT not the element's instance
6. private int numAuthors = 0;
7. ......
8. ......
9. public void addAuthor(Author author) {
10. authors[numAuthors] = author;
11. ++numAuthors;
12. }
13. }
15. // Test program
16. Book javaDummy = new Book("Java for Dummy", 19.99, 99);
17. System.out.println(javaDummy); // toString()
18. System.out.print("The authors are: ");
19. javaDummy.printAuthors();
21. javaDummy.addAuthor(new Author("Jim Jones", "jjones@somewhere.com", 'm'));
22. javaDummy.addAuthor(new Author("Mike Conroy", "mconroy@somewhere.com", 'm'));
23. System.out.println(javaDummy); // toString()
24. System.out.print("The authors are: ");
25. javaDummy.printAuthors();
26. Write a method called removeAuthorByName(authorName), that removes the author from this Book instance if authorName is present. The method shall return true if it succeeds. As always, test your code (make sure that you test for both basic situations here).

boolean removeAuthorByName(String authorName)