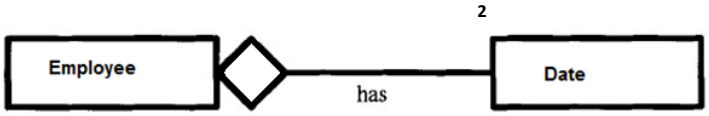
Lab Exercise (Chapter 8: Part 3)

Exercise 1: Aggregation (*has-a* relationship)

Create a aggregation classes as shown in UML diagram below:



An employee has a first name, last name, birth date and hire date.

A date class has day, month and year in integer.

Draw the class diagram and write the Java code of the Employee and Date class with the appropriate attributes, constructors (only two constructors required, one empty and one complete constructor), setters and getters.

You need to create a toString method to return the output below:

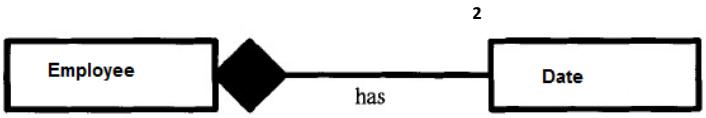
> *Employee with first name xxx last name xxx has birthday on xx/xx/xxxx and was hired on xx/xx/xxxx.*

Finally, create a test program 'TestEmployee' to test out the employee class by creating an employee with first name 'George' and last name 'Tan' who was born on 17th August 1990 and hired by the company on 25th June 2014. Then print the output. After that, update the employee last name to 'Chan' and print the output.

How about if you want to change the hired year to 2015?

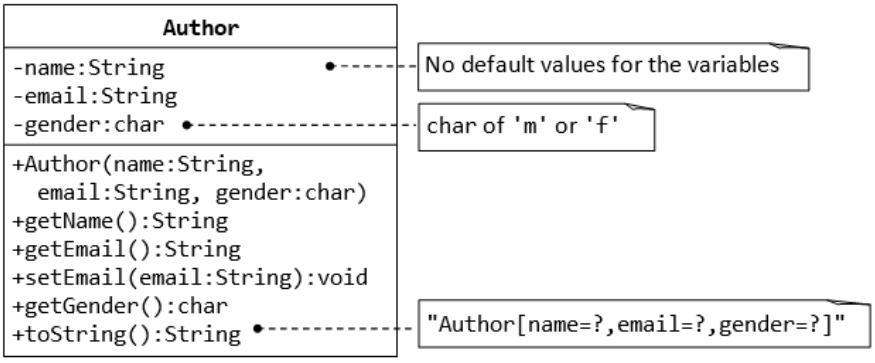
Exercise 2: Composition (*has-a* relationship)

Modify your program above to change it from aggregation to composition relationship.



Exercise 3: The Author and Book Classes (Aggregation)

In this exercise, you are going to practice the creation of aggregation relationship for the Author and Book class based on the following class diagrams:



A class called `Author` (as shown in the class diagram) is designed to model a book's author. It contains:

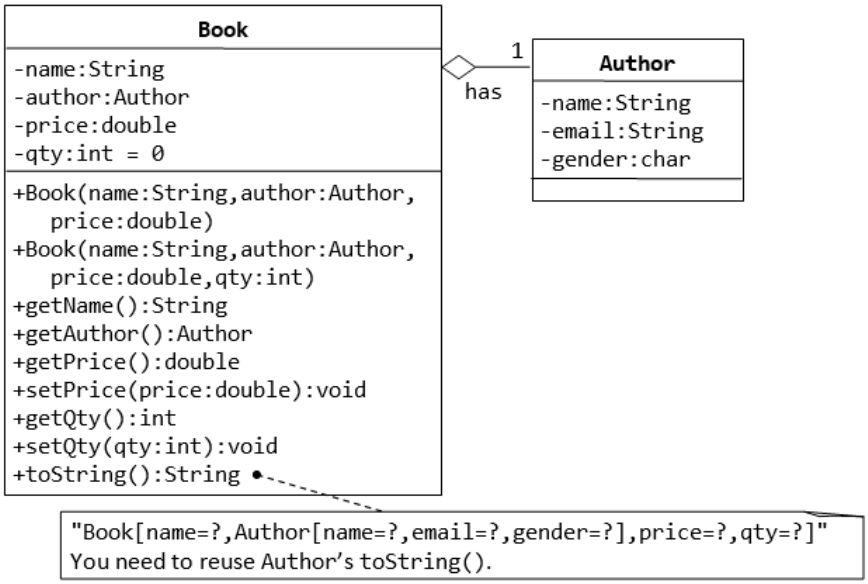
- Three `private` instance variables: `name` (`String`), `email` (`String`), and `gender` (`char` of either `'m'` or `'f'`);
- One constructor to initialize the `name`, `email` and `gender` with the given values;

```
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=?,email=?,gender=?]"`, e.g., `"Author[name=Tan Ah Teck,email=ahTeck@somewhere.com,gender=m]"`.

Write the `Author` class. Also write a *test driver* called `TestAuthor` to test all the `public` methods, e.g.,

```
Author ahTeck = new Author("Tan Ah Teck", "ahteck@nowhere.com", 'm'); // Test the constructor

System.out.println(ahTeck);  // Test toString()

ahTeck.setEmail("paulTan@nowhere.com");  // Test setter

System.out.println("name is: " + ahTeck.getName());     // Test getter

System.out.println("email is: " + ahTeck.getEmail());   // Test getter

System.out.println("gender is: " + ahTeck.getGender()); // Test getter
```



A class called `Book` is designed (as shown in the class diagram) to model a book written by *one* author. It contains:

- Four `private` instance variables: `name` (`String`), `author` (of the class `Author` you have just created, assume that a book has one and only one author), `price` (`double`), and `qty` (`int`);
- Two constructors:

```
public Book (String name, Author author, double price) { ..... }
```

```
public Book (String name, Author author, double price, int qty) { ..... }
```

- public methods `getName()`, `getAuthor()`, `getPrice()`, `setPrice()`, `getQty()`, `setQty()`.
- A `toString()` that returns `"Book[name=?,Author[name=?,email=?,gender=?],price=?,qty=?"`. You should reuse `Author`'s `toString()`.

Write the `Book` class (which uses the `Author` class written earlier). Also write a test driver called `TestBook` to test all the `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.,

```
// Construct an author instance
Author ahTeck = new Author("Tan Ah Teck", "ahteck@nowhere.com", 'm');
System.out.println(ahTeck);  // Author's toString()
Book dummyBook = new Book("Java for dummy", ahTeck, 19.95, 99);  // Test Book's Constructor
System.out.println(dummyBook);  // Test Book's toString()
// Test Getters and Setters
dummyBook.setPrice(29.95);
dummyBook.setQty(28);
System.out.println("name is: " + dummyBook.getName());
System.out.println("price is: " + dummyBook.getPrice());
System.out.println("qty is: " + dummyBook.getQty());
System.out.println("Author is: " + dummyBook.getAuthor());  // Author's toString()
System.out.println("Author's name is: " + dummyBook.getAuthor().getName());
System.out.println("Author's email is: " + dummyBook.getAuthor().getEmail());
// Use an anonymous instance of Author to construct a Book instance
Book anotherBook = new Book("more Java",
      new Author("Paul Tan", "paul@somewhere.com", 'm'), 29.95);
System.out.println(anotherBook);  // toString()
```

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 says `aBook`, `aBook.name` refers to the `name` of the book; whereas for an `Author`'s instance say `auAuthor`, `anAuthor.name` refers to the

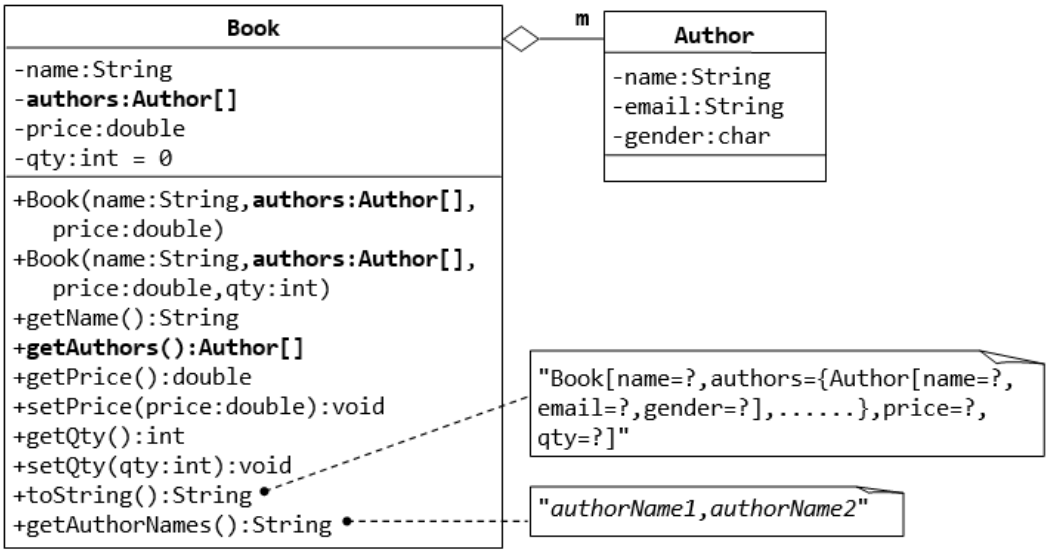
name of the author. There is no need (and not recommended) to call the variables bookName and authorName.

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 called getAuthorName(), getAuthorEmail(), getAuthorGender() in the Book class to return the name, email and gender of the author of the book. For example,

```
3. public String getAuthorName() {
4.     return author.getName();
5.        // cannot use author.name as name is private in Author class
   }
```

Exercise 4: Book and Author Classes Again - An Array of Objects as an Instance Variable



In the earlier exercise, 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.

Notes:

The constructors take an array of Author (i.e., Author[]), instead of an Author instance. In this design, once a Book instance is constructor, you cannot add or remove author.

The toString() method shall return "Book[name=?,authors={Author[name=?,email=?,gender=?],.....},price=?,qty=?]".

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 driver (called `TestBook`) to test the `Book` class.

Hints:

```
// Declare and allocate an array of Authors
Author[] authors = new Author[2];
authors[0] = new Author("Tan Ah Teck", "AhTeck@somewhere.com", 'm');
authors[1] = new Author("Paul Tan", "Paul@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()
```