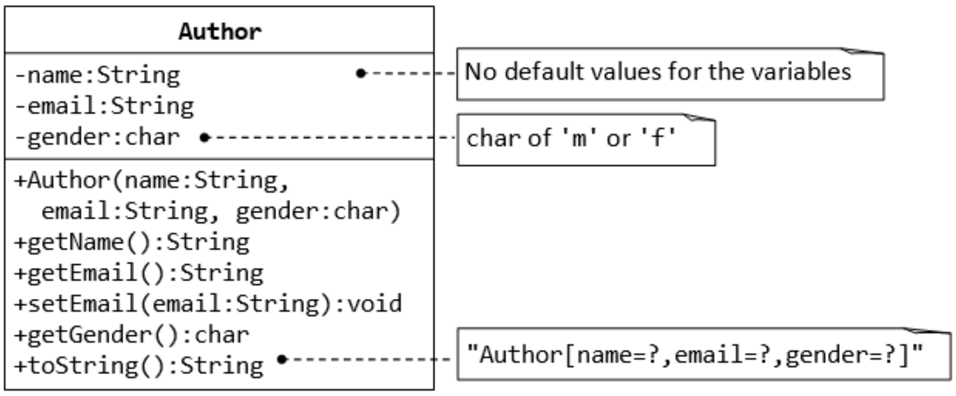
Exercise 07

Assignment 1 - Aggregation and composition


Author
-name:String
-email:String
-gender:char
+Author(name:String, email:String, gender:char)
+getName():String
+getEmail():String
+setEmail(email:String):void
+getGender():char
+toString():String
No default values for the variables
char of 'm' or 'f'
"Author[name=?,email=?,gender=?]"

This first exercise shall lead you through all the concepts involved in OOP Composition/Aggregation.

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;

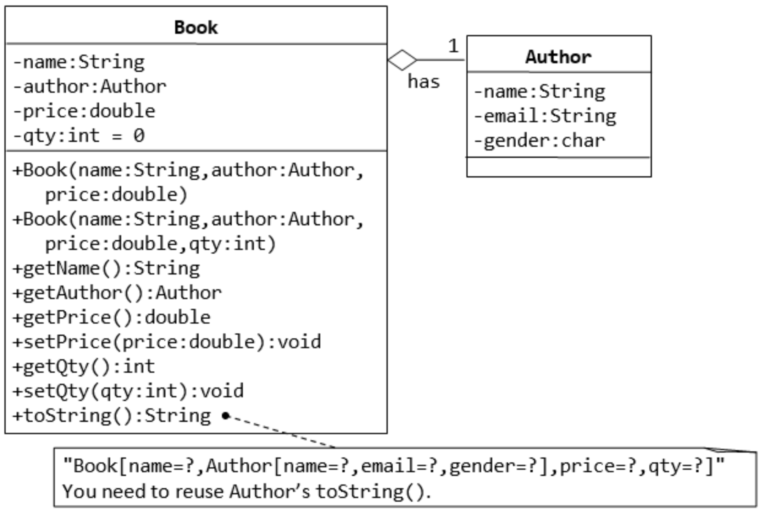
```
public Author (String name, String email, char gender) {......}
```

  (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 class* called TestAuthor to test all the public methods.


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
has
1
Author
-name:String
-email:String
-gender:char
"Book[name=?,Author[name=?,email=?,gender=?],price=?,qty=?]"
You need to reuse Author's toString().

Next, 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) { ...... }
```

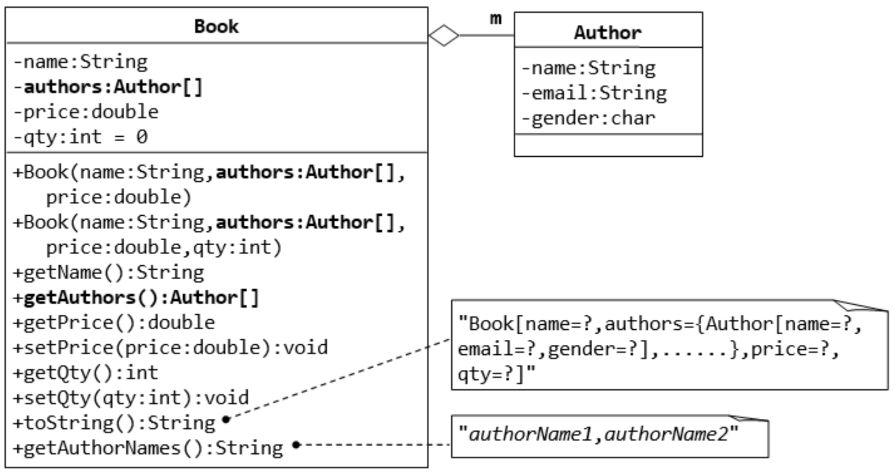
- public methods getName(), getAuthor(), getPrice(), setPrice(), getQty(), setQty().
- A toString() that returns "Book[name=?,Author[name=?,email=?,gender=?],price=?,qty=?". You should reuse Author's toString().

Write the Book class (which uses the Author class written earlier). Also write a test class 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.

Be aware that the 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 out the following:

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.


Book
-name:String
-authors:Author[]
-price:double
-qty:int = 0
+Book(name:String,authors:Author[], price:double)
+Book(name:String,authors:Author[], price:double,qty:int)
+getName():String
+getAuthors():Author[]
+getPrice():double
+setPrice(price:double):void
+getQty():int
+setQty(qty:int):void
+toString():String
+getAuthorNames():String
m
Author
-name:String
-email:String
-gender:char
"Book[name=?,authors={Author[name=?, email=?,gender=?],......},price=?, qty=?]"
"authorName1,authorName2"

In the current book class 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:

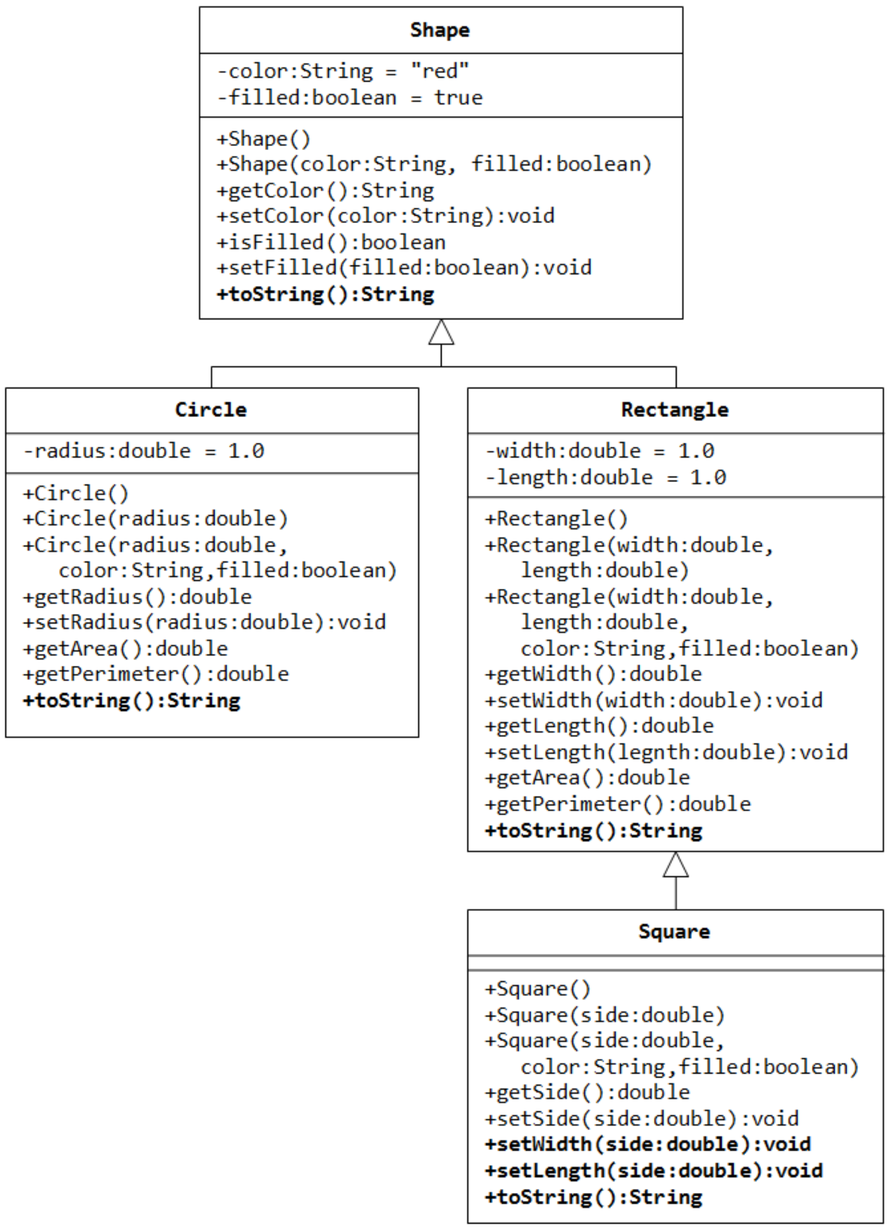
- The constructors take an array of Author (i.e., Author[]), instead of an Author instance. In this design, once a Book instance is constructor, you cannot add or remove author.

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

You are required to:

1. Write the code for the updated Book class (lets call it MultiBook). You shall re-use the `Author` class written earlier.
2. Write a test class (called `TestMultiBook`) to test the MultiBook class.

Assignment 2 - Inheritance


Shape
-color:String = "red"
-filled:boolean = true
+Shape()
+Shape(color:String, filled:boolean)
+getColor():String
+setColor(color:String):void
+isFilled():boolean
+setFilled(filled:boolean):void
+toString():String
Circle
-radius:double = 1.0
+Circle()
+Circle(radius:double)
+Circle(radius:double, color:String,filled:boolean)
+getRadius():double
+setRadius(radius:double):void
+getArea():double
+getPerimeter():double
+toString():String
Rectangle
-width:double = 1.0
-length:double = 1.0
+Rectangle()
+Rectangle(width:double, length:double)
+Rectangle(width:double, length:double, color:String,filled:boolean)
+getWidth():double
+setWidth(width:double):void
+getLength():double
+setLength(legnth:double):void
+getArea():double
+getPerimeter():double
+toString():String
Square
+Square()
+Square(side:double)
+Square(side:double, color:String,filled:boolean)
+getSide():double
+setSide(side:double):void
+setWidth(side:double):void
+setLength(side:double):void
+toString():String

This example will built up on the circle example from the previous exercise (06). Write a superclass called Shape (as shown in the class diagram), which contains:

- Two instance variables `color` (`String`) and `filled` (`boolean`).
- Two constructors: a no-arg (no-argument) constructor that initializes the `color` to "green" and `filled` to `true`, and a constructor that initializes the `color` and `filled` to the given values.
- Getter and setter for all the instance variables. By convention, the getter for a boolean variable xxx is called `isXXX()` (instead of `getXxx()` for all the other types).
- A `toString()` method that returns "`A Shape with color of xxx and filled/Not filled`".

Write a test program to test all the methods defined in Shape.

Write two subclasses of Shape called `Circle` and `Rectangle`, as shown in the class diagram.

The Circle class contains:

- An instance variable radius (double).
- Three constructors as shown. The no-arg constructor initializes the radius to 1.0.
- Getter and setter for the instance variable radius.
- Methods getArea() and getPerimeter().
- Override the toString() method inherited, to return "A Circle with radius=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

The Rectangle class contains:

- Two instance variables width (double) and length (double).
- Three constructors as shown. The no-arg constructor initializes the width and length to 1.0.
- Getter and setter for all the instance variables.
- Methods getArea() and getPerimeter().
- Override the toString() method inherited, to return "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

- Provide the appropriate constructors (as shown in the class diagram). Hint:

```
public Square(double side) {
   super(side, side);  // Call superclass Rectangle(double, double)
}
```

- Override the toString() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.
- Do you need to override the getArea() and getPerimeter()? Try them out.
- Override the setLength() and setWidth() to change both the width and length, so as to maintain the square geometry.

Write a test program to test all the methods defined in Shape, circle, rectangle and square.