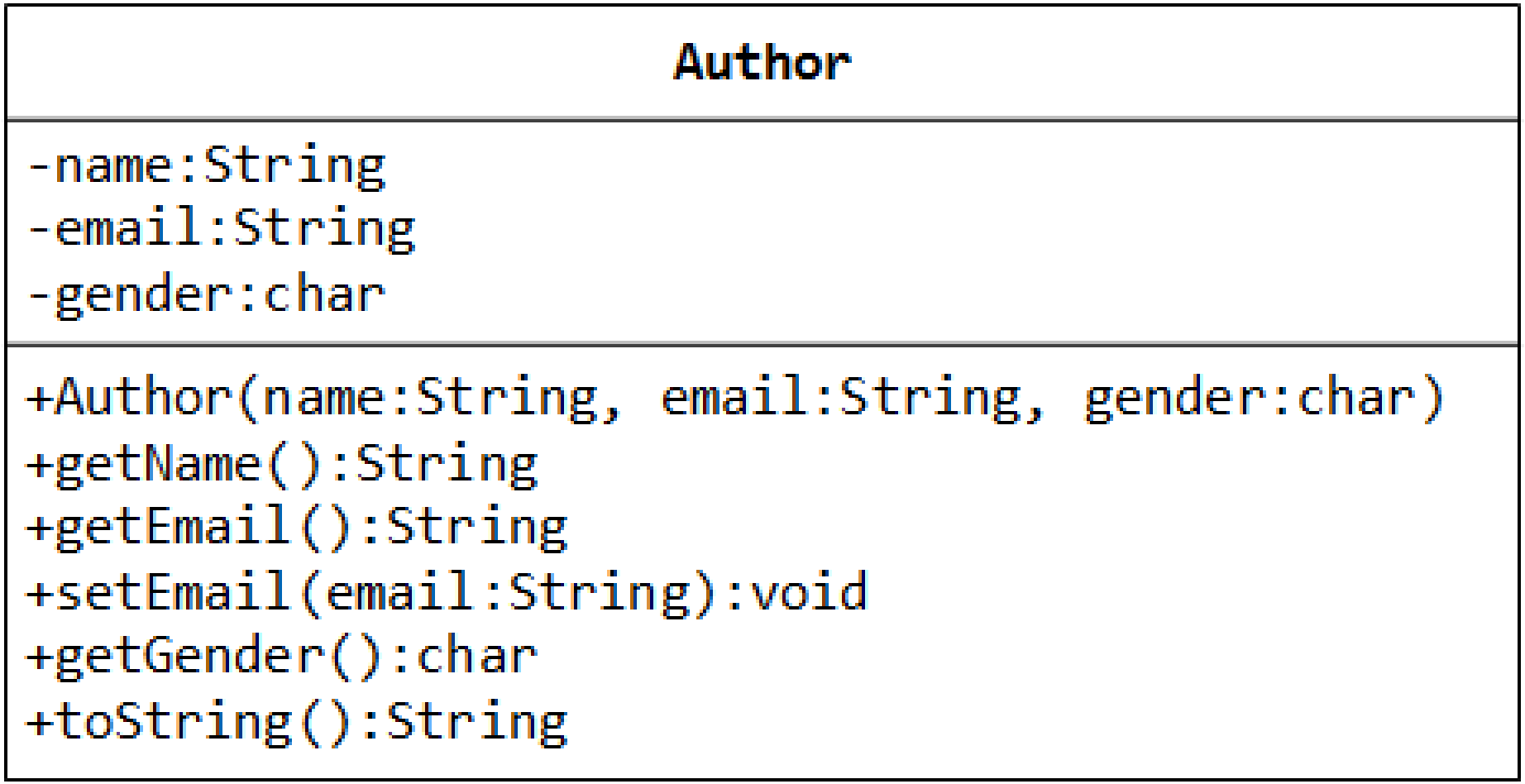
OOP Examples (II)

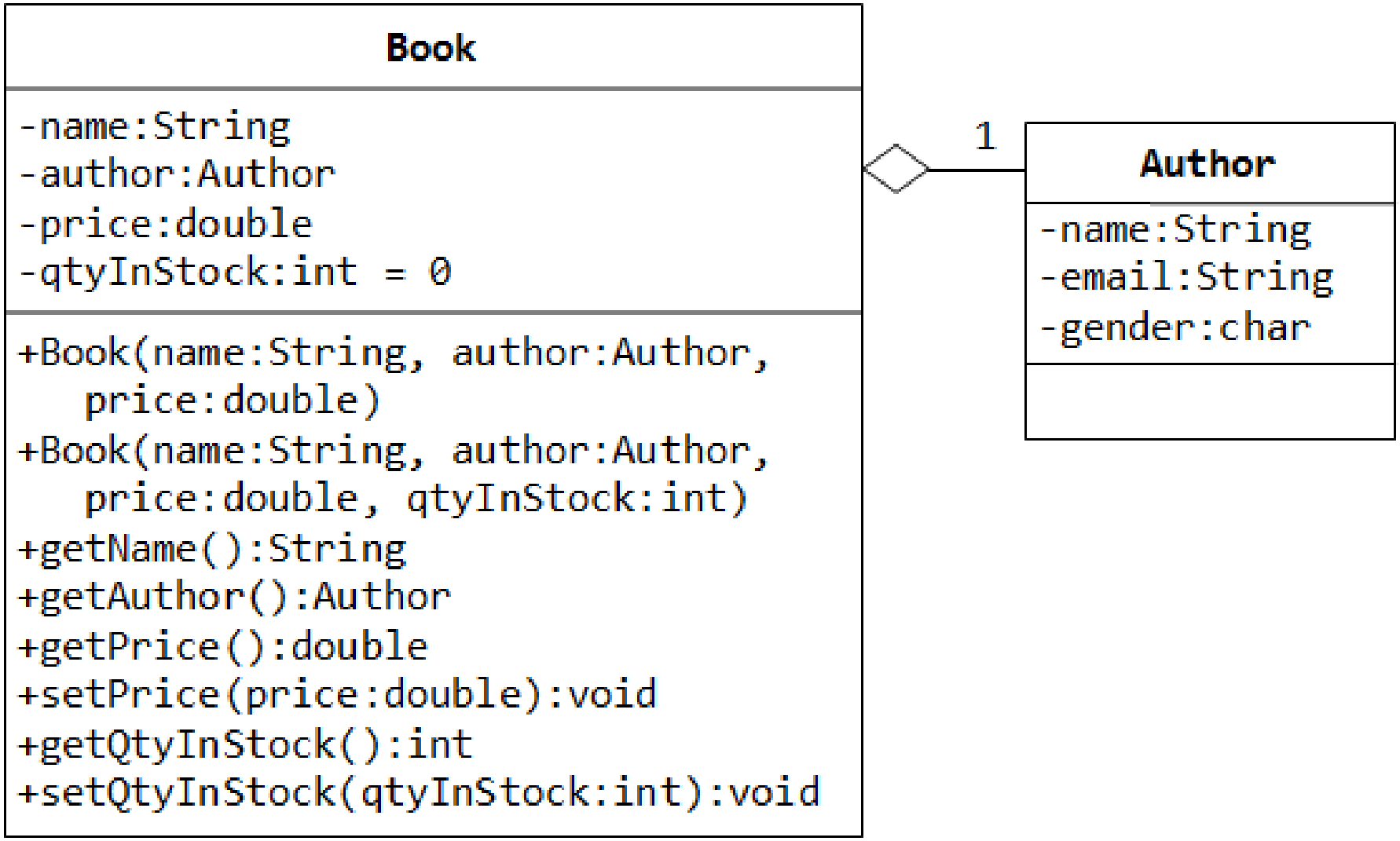
- **1**) A class called **Author** is designed as shown in the class diagram



- Content of the **Author** class:
- 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
- There is no default constructor for **Author**, as there are no defaults for name, email and gender
- Getters/setters: **getName()**, **getEmail()**, **setEmail()**, and **getGender()**
- There are no setters for **name** and **gender**, as these attributes cannot be changed
- The **toString()** method that returns "author-name (gender) at email", e.g., "*Author (m) at Author@upb.ro*"

- Write the **Author** class
- Write a test program called **TestAuthor**, which includes the **main** method, to test the constructor and public methods
- Create an author and call toString() method to display the information
- *Author anAuthor = new Author(“Author", “Author@upb.ro", 'm');*
- Result: *Author (m) at Author@upb.ro*
- Change the email of an author and display the information
- *anAuthor.setEmail(“Author@fils.upb.ro")*
- Result: *Author (m) at Author@fils.upb.ro*

- **2**) A class called **Book** is designed as shown in the class diagram



- Content of the **Book** class:
- 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 methods **getName()**, **getAuthor()**, **getPrice()**, **setPrice()**, **getQtyInStock()**, **setQtyInStock()**
- The **toString()** method 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 **Book** class (which uses the **Author** class written earlier)
- Write a test program called **TestBook**, which includes the **main** method, 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**
- *Author anAuthor = new Author("Student","student@upb.ro",'m');*
- *Book aBook = new Book("Java for dummies", anAuthor, 19.95, 1000);*
- *Book anotherBook = new Book("C for dummies", new Author("Teacher","teacher@upb.ro",'m'), 29.95, 999);*

- 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 **anAuthor**, **anAuthor.name** refers to the name of the author
- There is no need (and not recommended) to call the variables bookName and authorName

- a) Create a book instance and call **toString()** method to display the information
- Result: *'Java for dummies' by Student (m) at student@upb.ro*
- Create another book instance and call **toString()** method to display the information
- Result: *'C for dummies' by Teacher (m) at teacher@upb.ro*

- b) Display the name and email of the author from a **Book** instance
- Result for **aBook** object
- *Student*
- *student@upb.ro*
- Result for **anotherBook** object
- *Teacher*
- *teacher@upb.ro*

- c) Introduce new methods called **getAuthorName()**, **getAuthorEmail()**, **getAuthorGender()** in the **Book** class to return the name, email and gender of the author of the book and test the methods for the two previous **Book** objects: **aBook** and **anotherBook**
- Result for **aBook** object
- *Student*
- *student@upb.ro*
- *m*
- Result for **anotherBook** object
- *Teacher*
- *teacher@upb.ro*
- *m*

- **3**) Create a class, called **Person**, that contains:
- Three private member variables, more exactly two variables of type **String**, representing the last name and the first name of a person, respectively, and one variable of type **int**, representing the age of a person
- One constructor, with three arguments, that initializes the last name, the first name and the age of a person
- The **displayPerson()** method, with no arguments, that displays the last name, the first name and the age of a person
- The **getLast()** method, with no arguments, that returns a **String** representing the last name of a person
- The last name of a person is the field used as the search key
- Suppose that all the persons have different last names

- The second class, called **PersonArray**, contains:
- Two private member variables, more exactly a reference to an array of persons and the number of data items in the array
- One constructor, with one argument, that allocates memory for the array of persons and initializes the current number of data items in the array to 0
- The **insert()** method, with three arguments (the last name, the first name, and the age of a person), creates a new object of type **Person**, using the constructor of the class **Person**, and inserts the object in the array

- Searching for a given person is done using the **find(String searchName)** method, with one argument representing the last name of the person that we search
- This method returns an object of type **Person** or **null** if the person that it is searched does not appear in the array
- The deletion of a given person is done using the method **delete(String searchName)**, with one argument representing the last name of the person that we want to delete from the array
- This method returns a **boolean** value, telling if the deletion was successfully or not
- The deletion is not successfully when the person that we want to delete does not appear in the array

- The **displayArray()** method, with no arguments, that displays the content of the array, by calling the **displayPerson()** method, for each element of the array
- In the third class, called **PersonMain**, which includes the **main** method, five elements are inserted in the array of persons
- The content of the array is displayed on the screen
- A person is searched and an appropriate message is displayed
- Then a person is deleted from the array and the new content of the array is displayed