Object-Oriented Programming. Exercise 2.3

Unit 2. Inheritance and interfaces

In this exercise we will practice on how *inheritance*, *polymorphism* and *dynamic binding* allow reusing and redefining the behavior of previously defined classes, and how they provide adequate support for the *substitution principle*.

You will also learn about the benefits and needs for defining methods with protected visibility so that derived classes are allowed to manipulate and redefine the behavior of superclasses. You will also learn how a constructor of the derived class can invoke a constructor of its superclass, and how a method of the subclass can invoke the method of its superclass that is being redefined.

Finally, you will also learn how *interfaces* allow you to specify an operational interface that can be provided by various class instantiations.

Specifically, the `BookOnSale` class redefines the behavior of the `Book` class to model books that offer a discount on their final price. To do this, it redefines the `getFinalPrice` method of the superclass.

The `SalesBookStore` class redefines the behavior of the `BookStore` class in such a way that it allows creating both *normal* books and books *on sale*, and both types of books can be stored in the same data structure, thanks to the polymorphism and the *substitution principle*, since when the `getFinalPrice` method of a given book is invoked, the method corresponding to the *dynamic type* of the object will be properly invoked, thanks to dynamic binding.

Moreover, the `FlexSalesBookStore` class also redefines the behavior of the `BookStore` class, but in this case the selection of the books *on sale* is done in a more flexible way, so that this selection mechanism is left *open* so that any mechanism that implements the operational interface specified through the `FlexDiscount` interface can be used. Note that any class, current or designed in the future, that implements such interface could be used to select the books on discount.

Note 1: To facilitate the implementation of the classes in this exercise, you may find convenient to change the visibility of some features in the `BookStore` class from `private` to `protected`. For instance, it would make sense to change the visibility of the `INITIAL_CAPACITY` constant and some auxiliary methods such as `addBook`. Instance variables have getters and setters, so there is no need to change their visibility.

Note 2: You can add as many auxiliary methods as you need as long as they are defined as **private**.

Exercise 1. (project prBookStore)

In this exercise we will define subclasses of classes `Book` and `BookStore` (developed in *Exercise 2.2*) so that we can have *authors on sale* so that their books have a discount in the book shop.

The figure below shows the UML class diagram for this package.

Class `BookOnSale`

The class `BookOnSale` (in package `prBookStore`) inherits from class `Book`, and therefore contains information on a given book, but, in addition, it allows the specification of a given discount percentage, which will be applied to the **base price** when calculating the final price of the book.

- `BookOnSale(String, String, double, double)  // @Constructor`

  It receives as parameters, in the following order, the name of the author, the title, the base price, and the discount percentage of the book.

- `getDiscount(): double`

  Returns the discount percentage of the book.

- `getFinalPrice(): double  // @Redefinition`

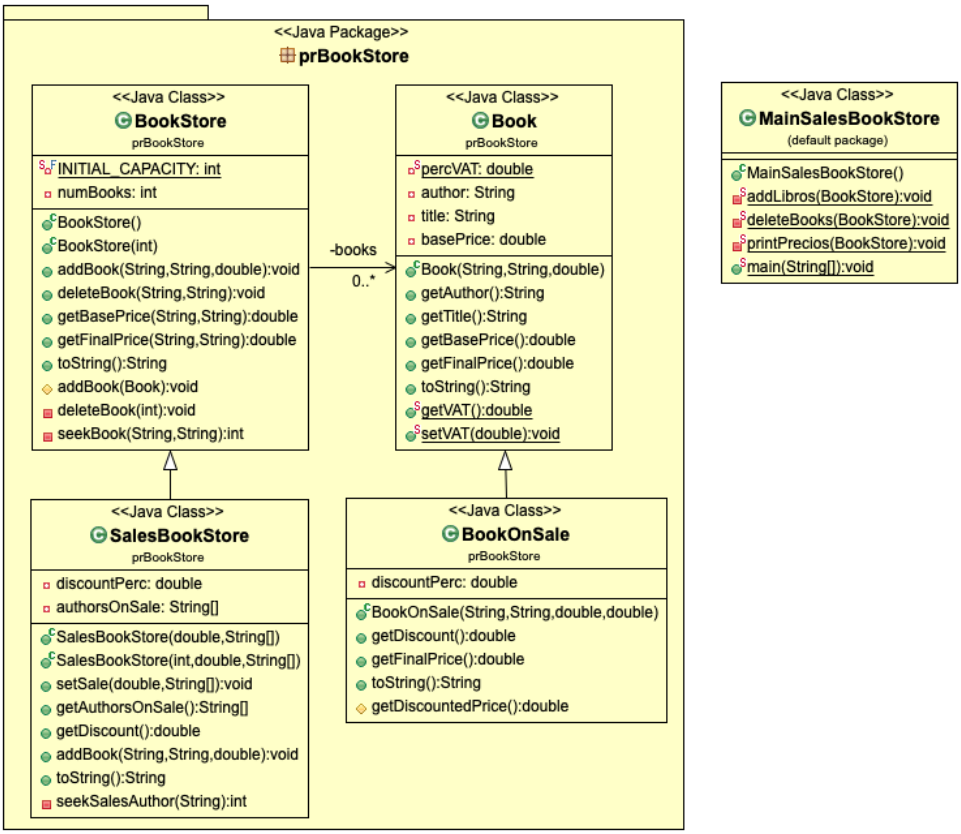

Figura 1: UML class diagram

Returns the final price of the book, applying the discount to the base price, and including VAT, according to the following equations.

$$PX = BP - BP \times Discount \div 100$$

$$BF = PX + PX \times VAT \div 100$$

- `toString(): String` *// @Redefinition*

  Returns the textual representation of the object, according to the format of the following example:

  `(Isaac Asimov; La Fundación; 7.30; 20.0%; 5.84; 10.0%; 6.424)`

Class `SalesBookStore`

The `SalesBookStore` class (in the `prBookStore` package) extends the `BookStore` class, allowing you to create and store books with discount. For this, it also contains an array with the names of the authors on sale, as well as the percentage of the offer to be applied to the books of these authors.

Note 3: Comparisons of both authors' names and book titles should be made without distinguishing between upper and lower case letters.

Note 4: Use private and/or protected auxiliary methods to simplify and modularize complex methods.

- `SalesBookStore(double, String[])` *// @Constructor*

  Builds an empty `SalesBookStore` object (no books) with the array to store books created with a default initial capacity. Furthermore, the discount percentage is received as the first parameter, and the array with the names of the authors on offer is received as the second parameter.

- `SalesBookStore(int, double, String[])` *// @Constructor*

Builds an empty `SalesBookStore` object (no books) with the array to store books created with the size received as first argument. Furthermore, the discount percentage is received as second argument, and the array with the names of the authors on offer is received as the third parameter.

- `getDiscount(): double`

  Returns the discount percentage.

- `addBook(String, String, double): void  // @Redefinition`

  If the name of the author (received as first parameter) is one of the authors in the list of authors on sale, then creates a `BookOnSale` object with the name of the author, the title, and the base price received as parameters, and the discount of the book store associated to such authors. Otherwise, it creates a `Book` object with the author name, title, and base price received as parameters. In both cases, the `addBook` method is invoked to add it to the library.

  Invokes: `addBook` (changed to *protected* in the `BookStore` class).

- `toString(): String  // @Redefinition`

  Returns the string representation of the object, according to the format of the following example (without line breaks):

```
20.0%[George Orwell, Isaac Asimov]
[(George Orwell; 1984; 6.20; 20.0%; 4.96; 10.0%; 5.456),
 (Philip K. Dick; ¿Sueñan los androides con ovejas eléctricas?; 3.50; 10.0%; 3.85),
 (Isaac Asimov; Fundación e Imperio; 9.40; 20.0%; 7.52; 10.0%; 8.272),
 (Ray Bradbury; Fahrenheit 451; 7.40; 10.0%; 8.14),
 (Aldous Huxley; Un Mundo Feliz; 6.50; 10.0%; 7.15),
 (Isaac Asimov; La Fundación; 7.30; 20.0%; 5.84; 10.0%; 6.424),
 (William Gibson; Neuromante; 8.30; 10.0%; 9.13),
 (Isaac Asimov; Segunda Fundación; 8.10; 20.0%; 6.48; 10.0%; 7.128),
 (Isaac Newton; Arithmetica Universalis; 10.50; 10.0%; 11.55)]
```

The application `MainSalesBookStore`

Implement an application (in the anonymous package) to test the previous classes. In it, create a `SalesBookStore` object with a 20.0 % discount for the authors *George Orwell* and *Isaac Asimov*. Then, add the following books to the bookstore:

```
("george orwell", "1984", 8.20)
("Philip K. Dick", "¿Sueñan los androides con ovejas eléctricas?", 3.50)
("Isaac Asimov", "Fundación e Imperio", 9.40)
("Ray Bradbury", "Fahrenheit 451", 7.40)
("Aldous Huxley", "Un Mundo Feliz", 6.50)
("Isaac Asimov", "La Fundación", 7.30)
("William Gibson", "Neuromante", 8.30)
("Isaac Asimov", "Segunda Fundación", 8.10)
("Isaac Newton", "arithmetica universalis", 7.50)
("George Orwell", "1984", 6.20)
("Isaac Newton", "Arithmetica Universalis", 10.50)
```

The output of the program should be the following one (without new lines):

```
20.0%[George Orwell, Isaac Asimov]
[(George Orwell; 1984; 6.20; 20.0%; 4.96; 10.0%; 5.456),
 (Philip K. Dick; ¿Sueñan los androides con ovejas eléctricas?; 3.50; 10.0%; 3.85),
 (Isaac Asimov; Fundación e Imperio; 9.40; 20.0%; 7.52; 10.0%; 8.272),
 (Ray Bradbury; Fahrenheit 451; 7.40; 10.0%; 8.14),
 (Aldous Huxley; Un Mundo Feliz; 6.50; 10.0%; 7.15),
 (Isaac Asimov; La Fundación; 7.30; 20.0%; 5.84; 10.0%; 6.424),
 (William Gibson; Neuromante; 8.30; 10.0%; 9.13),
 (Isaac Asimov; Segunda Fundación; 8.10; 20.0%; 6.48; 10.0%; 7.128),
 (Isaac Newton; Arithmetica Universalis; 10.50; 10.0%; 11.55)]
```

Then, the following books will be removed:

```
("George Orwell", "1984")
("Aldous Huxley", "Un Mundo Feliz")
("Isaac Newton", "Arithmetica Universalis")
("James Gosling", "The Java Language Specification")
```

The output of the program should be the following one (without new lines):

```
20.0%[George Orwell, Isaac Asimov]
[(William Gibson; Neuromante; 8.3; 10.0%; 9.13),
 (Philip K. Dick; ¿Sueñan los androides con ovejas eléctricas?; 3.5; 10.0%; 3.85),
 (Isaac Asimov; Fundación e Imperio; 9.4; 20.0%; 7.52; 10.0%; 8.272),
 (Ray Bradbury; Fahrenheit 451; 7.4; 10.0%; 8.14),
 (Isaac Asimov; Segunda Fundación; 8.1; 20.0%; 6.48; 10.0%; 7.128),
 (Isaac Asimov; La Fundación; 7.3; 20.0%; 5.84; 10.0%; 6.424)]
```

Finally, show the final price of the following books:

```
("George Orwell", "1984")
("Philip K. Dick", "¿Sueñan los androides con ovejas eléctricas?")
("isaac asimov", "fundación e imperio")
("Ray Bradbury", "Fahrenheit 451")
("Aldous Huxley", "Un Mundo Feliz")
("Isaac Asimov", "La Fundación")
("william gibson", "neuromante")
("Isaac Asimov", "Segunda Fundación")
("Isaac Newton", "Arithmetica Universalis")
```

The output of the program should be the following one (without new lines):

```
getFinalPrice(George Orwell, 1984): 0.0
getFinalPrice(Philip K. Dick, ¿Sueñan los androides con ovejas eléctricas?): 3.85
getFinalPrice(isaac asimov, fundación e imperio): 8.272
getFinalPrice(Ray Bradbury, Fahrenheit 451): 8.14
getFinalPrice(Aldous Huxley, Un Mundo Feliz): 0.0
getFinalPrice(Isaac Asimov, La Fundación): 6.424
getFinalPrice(william gibson, neuromante): 9.13
getFinalPrice(Isaac Asimov, Segunda Fundación): 7.128
getFinalPrice(Isaac Newton, Arithmetica Universalis): 0.0
```

Exercise 2. (project prBookStore)

In this exercise we will define subclasses of classes `Book` and `BookStore` (developed in *Exercise 2.2*) so that we can specify different discount mechanisms on the books in the book store.

The figure below shows the UML class diagram for this package.

Interface `FlexDiscount`

The `FlexDiscount` *interface* (in the `prBookStore` package) specifies the methods needed to calculate the discount percentage to be applied to a particular book.

- `getDiscount(Book): double`

  Returns the discount percentage that must be applied to a certain book received as a parameter. If no discount should be applied, it will return the value zero.

Class `DiscountPrice`

The `DiscountPrice` class (in the `prBookStore` package) implements the *interface* `FlexDiscount` and provides a method to calculate the discount percentage on the base price of the book, to be applied starting at a given threshold, both specified in the construction of the object.

- `DiscountPrice(double, double)`

  Builds an object with the discount percentage and the price threshold received as parameters, in that order.

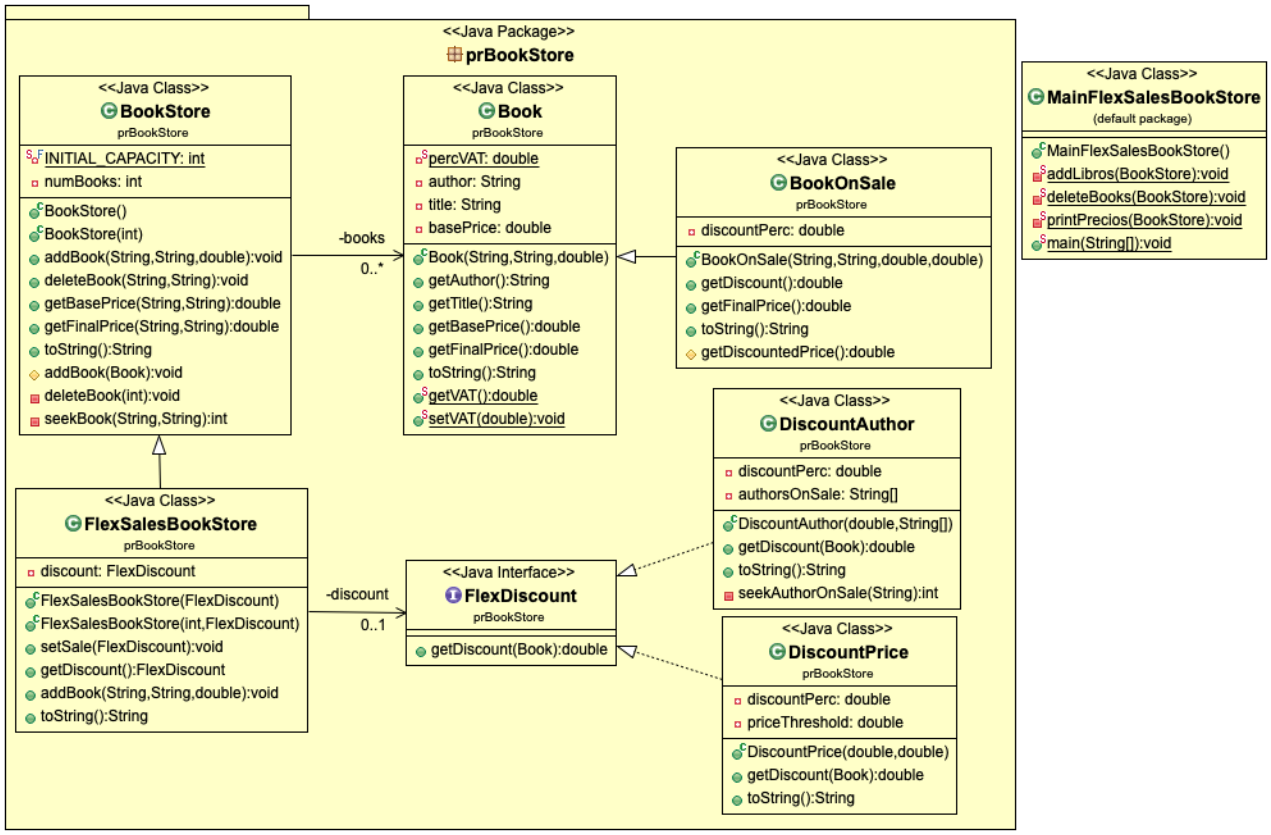

Figura 2: UML class diagram

- `getDiscount(Book): double` *// @Redefinition*

  Calculates the discount percentage to be applied to a certain book received as a parameter. A discount is applied if the base price of the book is greater than or equal to the threshold specified in the construction of the object; in case no discount should be applied, it will return the value zero.

- `toString(): String` *// @Redefinition*

  Returns the textual representation of the object, according to the format of the following example:

  `20.0%(8)`

Class `DiscountAuthor`

The `DiscountAuthor` class (in the`prBookStore` package) implements the *interface* `FlexDiscount` and provides a method to calculate the discount percentage to be applied on the on-sale authors. The list of authors on which to apply a discount and such a discount are specified in the object construction.

Note 5: Comparisons of author names and book titles should be made without distinguishing between upper and lower case letters.

- `DiscountAuthor(double, String[])`

  Builds an object with the discount percentage and the array of authors on discount.

- `getDiscount(Book): double` *// @Redefinition*

  Calculates the discount percentage to be applied to the book received as parameter. If the author's name is found in the array of authors on sale specified in the construction of the object, the corresponding value is returned; otherwise, it will return the value zero.

- `toString(): String` *// @Redefinition*

Returns the textual representation of the object, according to the format of the following example:

```
20.0%[George Orwell, Isaac Asimov]
```

Class `FlexSalesBookStore`

The `FlexSalesBookStore` class (in the `prBookStore` package) extends the `BookStore` class, but allows you to create and store books on sale. To do this, it contains a reference to an object that implements the `FlexDiscount` interface.

Note 6: Comparisons of author names and book titles should be made without distinguishing between upper and lower case letters.

Note 7: Use as many private and protected methods as necessary to simplify and appropriately modularize the implementation of complex methods.

- `FlexSalesBookStore(FlexDiscount)`

  Builds an empty `FlexSalesBookStore` object (no books) with an array of books with a default initial capacity. It also stores the reference to the object to calculate book offers received as parameter.

- `FlexSalesBookStore(int, FlexDiscount)`

  Builds an empty `FlexSalesBookStore` object (no books) with an array of books with initial capacity the integer value received as argument. It also stores the reference to the object to calculate book offers received as second parameter.

- `addBook(String, String, double): void  // @Redefinition`

  Creates a new `Book` object with the author name, title, and base price received as parameters and adds it to the library (sing the `addBook` method). If the calculated discount percentage for this book is other than zero, then the created book is a `BookOnSale` object, with the calculated discount.

  > Invokes: `FlexDiscount.getDiscount` and `addBook` (changed to *protected* in the `BookStore` class).

- `toString(): String  // @Redefinition`

  Returns the textual representation of the object, according to the format of the following example (without the line breaks):[1]

  ```
  20.0%[George Orwell, Isaac Asimov]
  [(George Orwell; 1984; 6.20; 20.0%; 4.96; 10.0%; 5.456),
  (Philip K. Dick; ¿Sueñan los androides con ovejas eléctricas?; 3.50; 10.0%; 3.85),
  (Isaac Asimov; Fundación e Imperio; 9.40; 20.0%; 7.52; 10.0%; 8.272),
  (Ray Bradbury; Fahrenheit 451; 7.40; 10.0%; 8.14),
  (Aldous Huxley; Un Mundo Feliz; 6.50; 10.0%; 7.15),
  (Isaac Asimov; La Fundación; 7.30; 20.0%; 5.84; 10.0%; 6.424),
  (William Gibson; Neuromante; 8.30; 10.0%; 9.13),
  (Isaac Asimov; Segunda Fundación; 8.10; 20.0%; 6.48; 10.0%; 7.128),
  (Isaac Newton; Arithmetica Universalis; 10.50; 10.0%; 11.55)]
  ```

The first line shows information on the type of discount (in this case it is a 20 % discount on authors). The rest of the information shows the books in the bookstore.

Main class `MainFlexSalesBookStore`

Develop an application (in the anonymous package) to test the previous classes. In it, you will create a `FlexSalesBookStore` object with a `DiscountAuthor` object that specifies a 20.0 % discount for authors *George Orwell* and *Isaac Asimov*. Then, add the following books to the bookstore:

[1] Note that the first line corresponds to the result of the `toString()` method on the corresponding instance of `FlexDiscount`.

```
("george orwell", "1984", 8.20)
("Philip K. Dick", "¿Sueñan los androides con ovejas eléctricas?", 3.50)
("Isaac Asimov", "Fundación e Imperio", 9.40)
("Ray Bradbury", "Fahrenheit 451", 7.40)
("Aldous Huxley", "Un Mundo Feliz", 6.50)
("Isaac Asimov", "La Fundación", 7.30)
("William Gibson", "Neuromante", 8.30)
("Isaac Asimov", "Segunda Fundación", 8.10)
("Isaac Newton", "arithmetica universalis", 7.50)
("George Orwell", "1984", 6.20)
("Isaac Newton", "Arithmetica Universalis", 10.50)
```

After that, print in the terminal the library. The output should look as follows (without new lines):

```
20.0%[George Orwell, Isaac Asimov]
[(George Orwell; 1984; 6.20; 20.0%; 4.96; 10.0%; 5.456),
 (Philip K. Dick; ¿Sueñan los androides con ovejas eléctricas?; 3.50; 10.0%; 3.85),
 (Isaac Asimov; Fundación e Imperio; 9.40; 20.0%; 7.52; 10.0%; 8.272),
 (Ray Bradbury; Fahrenheit 451; 7.40; 10.0%; 8.14),
 (Aldous Huxley; Un Mundo Feliz; 6.50; 10.0%; 7.15),
 (Isaac Asimov; La Fundación; 7.30; 20.0%; 5.84; 10.0%; 6.424),
 (William Gibson; Neuromante; 8.30; 10.0%; 9.13),
 (Isaac Asimov; Segunda Fundación; 8.10; 20.0%; 6.48; 10.0%; 7.128),
 (Isaac Newton; Arithmetica Universalis; 10.50; 10.0%; 11.55)]
```

After that, delete the following books:

```
("George Orwell", "1984")
("Aldous Huxley", "Un Mundo Feliz")
("Isaac Newton", "Arithmetica Universalis")
("James Gosling", "The Java Language Specification")
```

After that, print in the terminal the library. The output should look as follows (without new lines):

```
20.0%[George Orwell, Isaac Asimov]
[(William Gibson; Neuromante; 8.3; 10.0%; 9.13),
 (Philip K. Dick; ¿Sueñan los androides con ovejas eléctricas?; 3.5; 10.0%; 3.85),
 (Isaac Asimov; Fundación e Imperio; 9.4; 20.0%; 7.52; 10.0%; 8.272),
 (Ray Bradbury; Fahrenheit 451; 7.4; 10.0%; 8.14),
 (Isaac Asimov; Segunda Fundación; 8.1; 20.0%; 6.48; 10.0%; 7.128),
 (Isaac Asimov; La Fundación; 7.3; 20.0%; 5.84; 10.0%; 6.424)]
```

Finally, show the final price of the following books:

```
("George Orwell", "1984")
("Philip K. Dick", "¿Sueñan los androides con ovejas eléctricas?")
("isaac asimov", "fundación e imperio")
("Ray Bradbury", "Fahrenheit 451")
("Aldous Huxley", "Un Mundo Feliz")
("Isaac Asimov", "La Fundación")
("william gibson", "neuromante")
("Isaac Asimov", "Segunda Fundación")
("Isaac Newton", "Arithmetica Universalis")
```

After that, print in the terminal the library. The output should look as follows (without new lines):

```
getFinalPrice(George Orwell, 1984): 0.0
getFinalPrice(Philip K. Dick, ¿Sueñan los androides con ovejas eléctricas?): 3.85
getFinalPrice(isaac asimov, fundación e imperio): 8.272
getFinalPrice(Ray Bradbury, Fahrenheit 451): 8.14
getFinalPrice(Aldous Huxley, Un Mundo Feliz): 0.0
getFinalPrice(Isaac Asimov, La Fundación): 6.424
getFinalPrice(william gibson, neuromante): 9.13
getFinalPrice(Isaac Asimov, Segunda Fundación): 7.128
getFinalPrice(Isaac Newton, Arithmetica Universalis): 0.0
```