

2. The `Book` class is used to store information about a book. A partial `Book` class definition is shown.

```
public class Book
{
    /** The title of the book */
    private String title;

    /** The price of the book */
    private double price;

    /** Creates a new Book with given title and price */
    public Book(String bookTitle, double bookPrice)
    { /* implementation not shown */ }

    /** Returns the title of the book */
    public String getTitle()
    { return title; }

    /** Returns a string containing the title and price of the Book */
    public String getBookInfo()
    {
        return title + "-" + price;
    }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

You will write a class `Textbook`, which is a subclass of `Book`.

A `Textbook` has an edition number, which is a positive integer used to identify different versions of the book. The `getBookInfo` method, when called on a `Textbook`, returns a string that also includes the edition information, as shown in the example.

Information about the book title and price must be maintained in the `Book` class. Information about the edition must be maintained in the `Textbook` class.

The `Textbook` class contains an additional method, `canSubstituteFor`, which returns `true` if a `Textbook` is a valid substitute for another `Textbook` and returns `false` otherwise. The current `Textbook` is a valid substitute for the `Textbook` referenced by the parameter of the `canSubstituteFor` method if the two `Textbook` objects have the same title and if the edition of the current `Textbook` is greater than or equal to the edition of the parameter.

**GO ON TO THE NEXT PAGE.**

The following table contains a sample code execution sequence and the corresponding results. The code execution sequence appears in a class other than `Book` or `Textbook`.

Statement	Value Returned (blank if no value)	Class Specification
<code>Textbook bio2015 = new Textbook("Biology", 49.75, 2);</code>		bio2015 is a <code>Textbook</code> with a title of "Biology", a price of 49.75, and an edition of 2.
<code>Textbook bio2019 = new Textbook("Biology", 39.75, 3);</code>		bio2019 is a <code>Textbook</code> with a title of "Biology", a price of 39.75, and an edition of 3.
<code>bio2019.getEdition();</code>	3	The edition is returned.
<code>bio2019.getBookInfo();</code>	"Biology-39.75-3"	The formatted string containing the title, price, and edition of bio2019 is returned.
<code>bio2019. canSubstituteFor(bio2015);</code>	true	bio2019 is a valid substitute for bio2015, since their titles are the same and the edition of bio2019 is greater than or equal to the edition of bio2015.
<code>bio2015. canSubstituteFor(bio2019);</code>	false	bio2015 is not a valid substitute for bio2019, since the edition of bio2015 is less than the edition of bio2019.
<code>Textbook math = new Textbook("Calculus", 45.25, 1);</code>		math is a <code>Textbook</code> with a title of "Calculus", a price of 45.25, and an edition of 1.
<code>bio2015. canSubstituteFor(math);</code>	false	bio2015 is not a valid substitute for math, since the title of bio2015 is not the same as the title of math.

Write the complete `Textbook` class. Your implementation must meet all specifications and conform to the examples shown in the preceding table.

---

**Begin your response at the top of a new page in the Free Response booklet and fill in the appropriate circle indicating the question number. If there are multiple parts to this question, write the part letter with your response.**

**GO ON TO THE NEXT PAGE.**