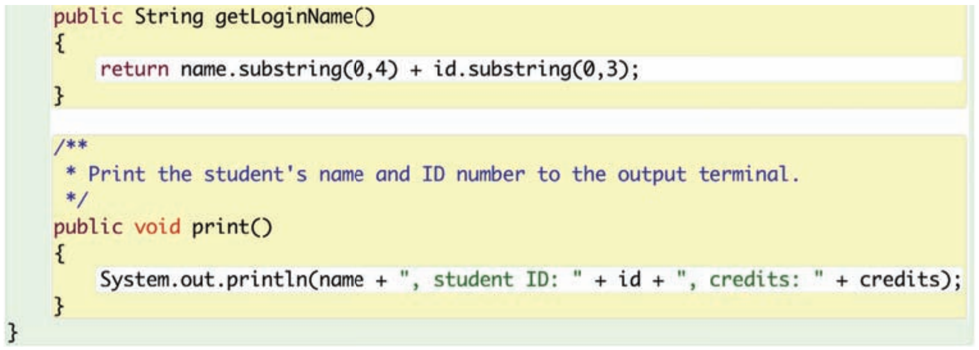
Code 2.9 continued
The **Student** class



```
    public String getLoginName()
    {
        return name.substring(0,4) + id.substring(0,3);
    }

    /**
     * Print the student's name and ID number to the output terminal.
     */
    public void print()
    {
        System.out.println(name + ", student ID: " + id + ", credits: " + credits);
    }
}
```

In this small example, the pieces of information we wish to store for a student are their name, their student ID, and the number of course credits they have obtained so far. All of this information is persistent during their time as a student, even if some of it changes during that time (the number of credits). We want to store this information in fields, therefore, to represent each student's state.

The class contains three fields: **`name`**, **`id`**, and **`credits`**. Each of these is initialized in the single constructor. The initial values of the first two are set from parameter values passed into the constructor. Each of the fields has an associated **`get`** accessor method, but only **`name`** and **`credits`** have associated mutator methods. This means that the value of an **`id`** field remains fixed once the object has been constructed. If a field's value cannot be changed once initialized, we say that it is *immutable*. Sometimes we make the complete state of an object immutable once it has been constructed; the **`String`** class is an important example of this.

2.21 Calling methods

The **`getLoginName`** method illustrates a new feature that is worth exploring:

```
public String getLoginName()
{
    return name.substring(0,4) +
           id.substring(0,3);
}
```

We are seeing two things in action here:

- Calling a method on another object, where the method returns a result.
- Using the value returned as a result as part of an expression.

Both **`name`** and **`id`** are **`String`** objects, and the **`String`** class has a method, **`substring`**, with the following header:

```
/**
 * Return a new string containing the characters from
 * beginIndex to (endIndex-1) from this string.
 */
public String substring(int beginIndex, int endIndex)
```

An index value of zero represents the first character of a string, so **getLoginName** takes the first four characters of the **name** string and the first three characters of the **id** string, then concatenates them together to form a new string. This new string is returned as the method's result. For instance, if **name** is the string **"Leonardo da Vinci"** and **id** is the string **"468366"**, then the string **"Leon468"** would be returned by this method.

We will learn more about method calling between objects in Chapter 3.

Exercise 2.74 Draw a picture of the form shown in Figure 2.3, representing the initial state of a **Student** object following its construction, with the following actual parameter values:

```
new Student("Benjamin Jonson", "738321")
```

Exercise 2.75 What would be returned by **getLoginName** for a student with **name "Henry Moore"** and **id "557214"**?

Exercise 2.76 Create a **Student** with name **"djb"** and **id "859012"**. What happens when **getLoginName** is called on this student? Why do you think this is?

Exercise 2.77 The **String** class defines a **length** accessor method with the following header:

```
/**
 * Return the number of characters in this string.
 */
public int length()
```

so the following is an example of its use with the **String** variable **fullName**:

```
fullName.length()
```

Add conditional statements to the constructor of **Student** to print an error message if either the length of the **fullName** parameter is less than four characters, or the length of the **studentId** parameter is less than three characters. However, the constructor should still use those parameters to set the **name** and **id** fields, even if the error message is printed. *Hint:* Use if-statements of the following form (that is, having no **else** part) to print the error messages.

```
if(perform a test on one of the parameters) {
    Print an error message if the test gave a true result
}
```

See Appendix D for further details of the different types of if-statements, if necessary.

Exercise 2.78 *Challenge exercise* Modify the **getLoginName** method of **Student** so that it always generates a login name, even if either the **name** or the **id** field is not strictly long enough. For strings shorter than the required length, use the whole string.