

## 4.7 Student Class: Nested if...else Statements

The example of [Figs. 4.4–4.5](#) demonstrates a nested if...else statement that determines a student's letter grade based on the student's average in a course.

### Class Student

Class `Student` ([Fig. 4.4](#)) has features similar to those of class `Account` (discussed in [Chapter 3](#)). Class `Student` stores a student's name and average and provides methods for manipulating these values.

```
1 // Fig. 4.4: Student.java
2 // Student class that stores a student name and av
3 public class Student {
4     private String name;
5     private double average;
6
7     // constructor initializes instance variables
8     public Student(String name, double average) {
9         this.name = name;
10
11         // validate that average is > 0.0 and <= 100.
12         // keep instance variable average's default v
13         if (average > 0.0) {
14             if (average <= 100.0) {
```

```

15         this.average = average; // assign to insta
           16         }
           17     }
           18 }
           19
20     // sets the Student's name
21     public void setName(String name) {
           22         this.name = name;
           23     }
           24
25     // retrieves the Student's name
26     public String getName() {
           27         return name;
           28     }
           29
30     // sets the Student's average
31     public void setAverage(double studentAverage) {
32         // validate that average is > 0.0 and <= 100.
33         // keep instance variable average's current v
           34         if (average > 0.0) {
35             if (average <= 100.0) {
36                 this.average = average; // assign to in
           37             }
           38         }
           39     }
           40
41     // retrieves the Student's average
42     public double getAverage() {
           43         return average;
           44     }
           45
46     // determines and returns the Student's letter g
47     public String getLetterGrade() {
48         String letterGrade = ""; // initialized to em
           49
           50         if (average >= 90.0) {
           51             letterGrade = "A";
           52         }
           53         else if (average >= 80.0) {
           54             letterGrade = "B";

```

```
55         }
56     else if (average >= 70.0) {
57         letterGrade = "C";
58     }
59     else if (average >= 60.0) {
60         letterGrade = "D";
61     }
62     else {
63         letterGrade = "F";
64     }
65
66     return letterGrade;
67 }
68 }
```

Fig. 4.4

Student class that stores a student name and average.

The class contains:

- instance variable `name` of type `String` (line 4) to store a `Student`'s name
- instance variable `average` of type `double` (line 5) to store a `Student`'s average in a course
- a constructor (lines 8–18) that initializes the `name` and `average`—in [Section 5.9](#), you'll learn how to express lines 13–14 and 34–35 more concisely with logical operators that can test multiple conditions
- methods `setName` and `getName` (lines 21–28) to *set* and *get* the `Student`'s name
- methods `setAverage` and `getAverage` (lines 31–44) to *set* and *get* the `Student`'s average

- method `getLetterGrade` (lines 47–67), which uses *nested if...else statements* to determine the Student's *letter grade* based on the Student's average

The constructor and method `setAverage` each use *nested if statements* (lines 13–17 and 34–38) to *validate* the value used to set the `average`—these statements ensure that the value is greater than `0.0` *and* less than or equal to `100.0`; otherwise, `average`'s value is left *unchanged*. Each `if` statement contains a *simple* condition. If the condition in line 13 is *true*, only then will the condition in line 14 be tested, and *only* if the conditions in both line 13 *and* line 14 are *true* will the statement in line 15 execute.



## Software Engineering Observation 4.1

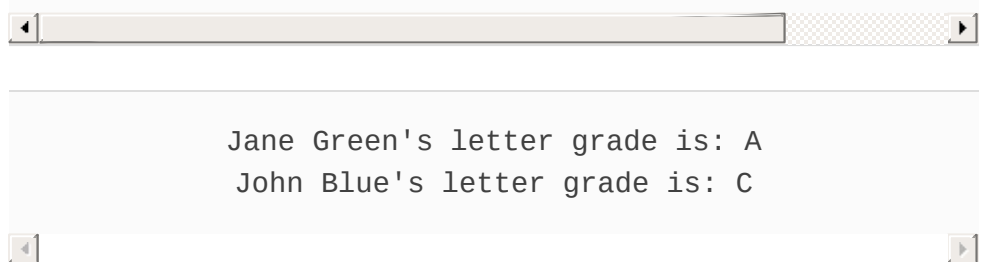
Recall from Chapter 3 that you should not call methods from constructors (we'll explain why in Chapter 10, *Object-Oriented Programming: Polymorphism and Interfaces*). For this reason, there is duplicated validation code in lines 13–17 and 34–38 of Fig. 4.4 and in subsequent examples.

## Class StudentTest

To demonstrate the nested `if...else` statements in class Student's `getLetterGrade` method, class

StudentTest's main method (Fig. 4.5) creates two Student objects (lines 5–6). Next, lines 8–11 display each Student's name and letter grade by calling the objects' getName and getLetterGrade methods, respectively.

```
1 // Fig. 4.5: StudentTest.java
2 // Create and test Student objects.
3 public class StudentTest {
4     public static void main(String[] args) {
5         Student account1 = new Student("Jane Green",
6         Student account2 = new Student("John Blue", 7
7
8         System.out.printf("%s's letter grade is: %s\n",
9         account1.getName(), account1.getLetterGrade()
10        System.out.printf("%s's letter grade is: %s\n",
11        account2.getName(), account2.getLetterGrade(
12        }
13    }
```



```
Jane Green's letter grade is: A
John Blue's letter grade is: C
```

Fig. 4.5

Create and test Student objects.