The constants KILOGRAMS_PER_POUND and METERS_PER_INCH are defined in lines 15–16. Using constants here makes programs easy to read.

3.10 Case Study: Computing Taxes

You can use nested if *statements to write a program for computing taxes.*

Key Point

The United States federal personal income tax is calculated based on filing status and taxable income. There are four filing statuses: single filers, married filing jointly or qualified widow(er), married filing separately, and head of household. The tax rates vary every year. Table 3.2 shows the rates for 2009. If you are, say, single with a taxable income of $10,000, the first $8,350 is taxed at 10% and the other $1,650 is taxed at 15%, so, your total tax is $1,082.50.

VideoNote
Use multi-way `if-else` statements

TABLE 3.2 2009 U.S. Federal Personal Tax Rates

Marginal Tax Rate	*Single*	*Married Filing Jointly or Qualifying Widow(er)*	*Married Filing Separately*	*Head of Household*
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351– $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 – $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

You are to write a program to compute personal income tax. Your program should prompt the user to enter the filing status and taxable income and compute the tax. Enter 0 for single filers, 1 for married filing jointly or qualified widow(er), 2 for married filing separately, and 3 for head of household.

Your program computes the tax for the taxable income based on the filing status. The filing status can be determined using if statements outlined as follows:

```
if (status == 0) {
  // Compute tax for single filers
}
else if (status == 1) {
  // Compute tax for married filing jointly or qualifying widow(er)
}
else if (status == 2) {
  // Compute tax for married filing separately
}
else if (status == 3) {
  // Compute tax for head of household
}
else {
  // Display wrong status
}
```

For each filing status there are six tax rates. Each rate is applied to a certain amount of taxable income. For example, of a taxable income of $400,000 for single filers, $8,350 is taxed at 10%, (33,950 – 8,350) at 15%, (82,250 – 33,950) at 25%, (171,550 – 82,250) at 28%, (372,950 – 171,550) at 33%, and (400,000 – 372,950) at 35%.

Listing 3.6 gives the solution for computing taxes for single filers. The complete solution is left as an exercise.

LISTING 3.6 ComputeTax.java

```java
import java.util.Scanner;

public class ComputeTax {
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter filing status
    System.out.print(
      "(0-single filer, 1-married jointly or qualifying widow(er), "
      + "\n2-married separately, 3-head of household)\n" +
      "Enter the filing status: ");
    int status = input.nextInt();

    // Prompt the user to enter taxable income
    System.out.print("Enter the taxable income: ");
    double income = input.nextDouble();

    // Compute tax
    double tax = 0;

    if (status == 0) { // Compute tax for single filers
      if (income <= 8350)
        tax = income * 0.10;
      else if (income <= 33950)
        tax = 8350 * 0.10 + (income - 8350) * 0.15;
      else if (income <= 82250)
        tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
          (income - 33950) * 0.25;
      else if (income <= 171550)
        tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
          (82250 - 33950) * 0.25 + (income - 82250) * 0.28;
      else if (income <= 372950)
        tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
          (82250 - 33950) * 0.25 + (171550 - 82250) * 0.28 +
          (income - 171550) * 0.33;
      else
        tax = 8350 * 0.10 + (33950 - 8350) * 0.15 +
          (82250 - 33950) * 0.25 + (171550 - 82250) * 0.28 +
          (372950 - 171550) * 0.33 + (income - 372950) * 0.35;
    }
    else if (status == 1) { // Left as exercise
      // Compute tax for married file jointly or qualifying widow(er)
    }
    else if (status == 2) { // Compute tax for married separately
      // Left as exercise
    }
    else if (status == 3) { // Compute tax for head of household
      // Left as exercise
    }
    else {
      System.out.println("Error: invalid status");
      System.exit(1);
    }

    // Display the result
    System.out.println("Tax is " + (int)(tax * 100) / 100.0);
  }
}
```

input status

input income

compute tax

exit program

display output

```
(0-single filer, 1-married jointly or qualifying widow(er),
2-married separately, 3-head of household)
Enter the filing status: 0 ↵Enter
Enter the taxable income: 400000 ↵Enter
Tax is 117683.5
```

line#	status	income	tax	output
13	0			
17		400000		
20			0	
38			117683.5	
57				Tax is 117683.5

The program receives the filing status and taxable income. The multi-way `if-else` statements (lines 22, 42, 45, 48, 51) check the filing status and compute the tax based on the filing status.

`System.exit(status)` (line 53) is defined in the `System` class. Invoking this method terminates the program. The status `0` indicates that the program is terminated normally. A nonzero status code indicates abnormal termination.

`System.exit(status)`

An initial value of `0` is assigned to `tax` (line 20). A compile error would occur if it had no initial value, because all of the other statements that assign values to `tax` are within the `if` statement. The compiler thinks that these statements may not be executed and therefore reports a compile error.

To test a program, you should provide the input that covers all cases. For this program, your input should cover all statuses (`0`, `1`, `2`, `3`). For each status, test the tax for each of the six brackets. So, there are a total of 24 cases.

test all cases

Tip

For all programs, you should write a small amount of code and test it before moving on to add more code. This is called *incremental development and testing*. This approach makes testing easier, because the errors are likely in the new code you just added.

incremental development and testing

3.17 Are the following two statements equivalent?

Check Point

MyProgrammingLab™

```
if (income <= 10000)
  tax = income * 0.1;
else if (income <= 20000)
  tax = 1000 +
    (income - 10000) * 0.15;
```

```
if (income <= 10000)
  tax = income * 0.1;
else if (income > 10000 &&
         income <= 20000)
  tax = 1000 +
    (income - 10000) * 0.15;
```

3.11 Logical Operators

The logical operators `!`, `&&`, `||`, *and* `^` *can be used to create a compound Boolean expression.*

Key Point

Sometimes, whether a statement is executed is determined by a combination of several conditions. You can use logical operators to combine these conditions to form a compound Boolean expression. *Logical operators*, also known as *Boolean operators*, operate on Boolean values