The switch Statement

THE SECOND BRANCHING STATEMENT in Java is the `switch` statement, which is introduced in this section. The `switch` statement is used far less often than the `if` statement, but it is sometimes useful for expressing a certain type of multiway branch.

The Basic switch Statement

A switch statement allows you to test the value of an expression and, depending on that value, to jump directly to some location within the switch statement. Only expressions of certain types can be used. The value of the expression can be one of the primitive integer types int, short, or byte. It can be the primitive char type. It can be *String*. Or it can be an enum type. In particular, note that the expression **cannot** be a double or float value.

The positions within a switch statement to which it can jump are marked with case labels that take the form: "case **constant**:". The **constant** here is a literal of the same type as the expression in the `switch`. A case label marks the position the computer jumps to when the expression evaluates to the given **constant** value. You can also use the label "default:" in a `switch` statement; this provides a default jump point that is used when the value of the expression is not listed in any case label.

A `switch` statement, as it is most often used, has the form:

```
switch (expression) {
   case constant-1:
      statements-1
      break;
   case constant-2:
      statements-2
      break;
      .
      .   // (more cases)
      .
   case constant-N:
      statements-N
      break;
   default:  // optional default case
      statements-(N+1)
} // end of switch statement
```

This has exactly the same effect as the following multiway `if` statement, but the `switch` statement can be more efficient because the computer can evaluate one expression and jump directly to the correct case, whereas in the `if` statement, the computer must evaluate up to `N` expressions before it knows which set of statements to execute:

```
if (expression == constant-1) { // but use .equals for String!!
    statements-1
}
else if (expression == constant-2) {
    statements-2
}
    .
    .
    .
else if (expression == constant-N) {
    statements-N
}
else {
    statements-(N+1)
}
```

The `break` statements in the `switch` are not actually required by the syntax of the `switch` statement. The effect of a `break` is to make the computer jump past the end of the switch statement, skipping over all the remaining cases. If you leave out the break statement, the computer will just forge ahead after completing one case and will execute the statements associated with the next case label. This is rarely what you want, but it is legal. (inside a subroutine, the `break` statement is sometimes replaced by a `return` statement, which terminates the subroutine as well as the switch.)

Note that you can leave out one of the groups of statements entirely (including the `break`). You then have two case labels in a row, containing two different constants. This just means that the computer will jump to the same place and perform the same action for each of the two constants.

Here is an example of a switch statement. This is not a useful example, but it should be easy for you to follow. Note, by the way, that the constants in the case labels don't have to be in any particular order, but they must all be different:

```
switch ( N ) {   // (Assume N is an integer variable.)
   case 1:
      System.out.println("The number is 1.");
      break;
   case 2:
   case 4:
   case 8:
      System.out.println("The number is 2, 4, or 8.");
      System.out.println("(That's a power of 2!)");
```

```
         break;
      case 3:
      case 6:
      case 9:
         System.out.println("The number is 3, 6, or 9.");
         System.out.println("(That's a multiple of 3!)");
         break;
      case 5:
         System.out.println("The number is 5.");
         break;
      default:
         System.out.println("The number is 7 or is outside the range 1 to
   9.");
   }
```

Menus and switch Statements

One application of `switch` statements is in processing menus. A menu is a list of options. The user selects one of the options. The computer has to respond to each possible choice in a different way. If the options are numbered 1, 2, ..., then the number of the chosen option can be used in a `switch` statement to select the proper response.

In a command-line program, the menu can be presented as a numbered list of options, and the user can choose an option by typing in its number. Here is an example that could be used in a variation of the `LengthConverter` example from the last reading:

```
int optionNumber;   // Option number from menu, selected by user.
double measurement; // A numerical measurement, input by the user.
                    //    The unit of measurement depends on which
                    //    option the user has selected.
double inches;      // The same measurement, converted into inches.

/* Display menu and get user's selected option number. */

System.out.println("What unit of measurement does your input use?");
System.out.println();
System.out.println("         1.  inches");
System.out.println("         2.  feet");
System.out.println("         3.  yards");
System.out.println("         4.  miles");
System.out.println();
System.out.println("Enter the number of your choice: ");
optionNumber = TextIO.getlnInt();

/* Read user's measurement and convert to inches. */

switch ( optionNumber ) {
   case 1:
       System.out.println("Enter the number of inches: ");
```

```
            measurement = TextIO.getlnDouble();
            inches = measurement;
            break;
        case 2:
            System.out.println("Enter the number of feet: ");
            measurement = TextIO.getlnDouble();
            inches = measurement * 12;
            break;
        case 3:
            System.out.println("Enter the number of yards: ");
            measurement = TextIO.getlnDouble();
            inches = measurement * 36;
            break;
        case 4:
            System.out.println("Enter the number of miles: ");
            measurement = TextIO.getlnDouble();
            inches = measurement * 12 * 5280;
            break;
        default:
            System.out.println("Error!  Illegal option number!  I quit!");
            System.exit(1);
     } // end switch

     /* Now go on to convert inches to feet, yards, and miles... */
```

This example could instead be written using a *String* in the `switch` statement:

```
     String units;       // Unit of measurement, entered by user.
     double measurement; // A numerical measurement, input by the user.
     double inches;      // The same measurement, converted into inches.

     /* Read the user's unit of measurement. */

     System.out.println("What unit of measurement does your input use?");
     System.out.print("Legal responses: inches, feet, yards, or miles : ");
     units = TextIO.getln().toLowerCase();

     /* Read user's measurement and convert to inches. */

     System.out.print("Enter the number of " + units + ":  ");
     measurement = TextIO.getlnDouble();

     switch ( units ) {
        case "inches":
            inches = measurement;
            break;
        case "feet":
            inches = measurement * 12;
            break;
        case "yards":
            inches = measurement * 36;
            break;
        case "miles":
            inches = measurement * 12 * 5280;
            break;
        default:
```

```
       System.out.println("Wait a minute!  Illegal unit of measure!  I
quit!");
       System.exit(1);
} // end switch
```

A New switch Statement Syntax

A new version of the `switch` statement has been added to the Java language in Java 14. The new version uses `->` in place of a colon after a `case`, and the code in a case is a single statement, possibly a block statement consisting of several statements enclosed in braces. No `break` statement is required, although one can be used to end a case early. This avoids the common error of having control accidently fall through from one case to the next because of an omitted `break`. Furthermore, instead of allowing just one value per case label, a case can take several values separated by commas. Using the new syntax, the first example in this section can be written as follows:

```
switch ( N ) {   // (Assume N is an integer variable.)
   case 1 -> System.out.println("The number is 1.");
   case 2, 4, 8 -> {
      System.out.println("The number is 2, 4, or 8.");
      System.out.println("(That's a power of 2!)");
   }
   case 3, 6, 9 -> {
      System.out.println("The number is 3, 6, or 9.");
      System.out.println("(That's a multiple of 3!)");
   }
   case 5 -> System.out.println("The number is 5.");
   default ->
      System.out.println("The number is 7 or is outside the range 1 to
9.");
}
```

This seems to me to be a big improvement. But the original `switch` syntax is still available.