Chapter 3: Switch-Case Control Structure

Switch-Case Control Structure

Sometimes when we have a lot of values to check, a series of nested if-else statements can be tedious. Java has a switch structure that allows us to check the value of a variable and branch any number of different ways based on that value, as long as the value is one of Java's integral types (char, byte, short, int, long, or boolean) or a String object.

Here's the general format of a switch statement, which involves both the switch and case keywords:

```
switch (<variable name>) {
    case <value1>:
        // code for this case
        break;
    case <value2>:
    case <value3>:
        // code for this case
        break;
      .
      .
      .
    case <valuen>:
        // code for this case
        break;
    default:
        // code for all other cases
}
```

The <*variable name*> in the first line, right after the switch keyword, is the variable whose value we want to test. The values <*value1*> through <*valuen*> following each case keyword are all the values we want to have different actions for. The code for each value follows the case statement containing that value, and you can list as many case values as you need. The final case, default, is optional. You'll use it if the variable doesn't contain any of the listed values.

Notice that the code for each value ends with a break statement. The break statement tells Java to exit the switch block. If the break statement is missing for a value, then after executing the code for that value, execution will *fall through*, or continue, to the next value's code. So the break statement is normally essential for each case.

However, note that the case for <*value2*> doesn't have a break statement. That can be convenient when two values require the same action. In this case, Java will execute the same code for <*value2*> and <*value3*>.

I think an example will help clarify things. Here's a simple Java program that converts a numeric month to that month's name:

```
import java.util.Scanner;
/**
 * SwitchTest1 is a simple example of a switch structure. It reads
 * a number from the keyboard, converts that number to the equivalent
 * month's name, and displays that name.
 *
 * @author Merrill Hall
 * @version 1.0
 */
public class SwitchTest1 {
    public static void main(String[] args) {
        int monthNumber = 0;
        String monthName = "";
        Scanner in = new Scanner(System.in);

        System.out.print("Enter a month's number: ");
        if (in.hasNextInt()) {
            monthNumber = in.nextInt();
            switch (monthNumber) {
                case 1:
                    monthName = "January";
                    break;
                case 2:
                    monthName = "February";
                    break;
                case 3:
                    monthName = "March";
                    break;
                case 4:
                    monthName = "April";
                    break;
                case 5:
                    monthName = "May";
                    break;
                case 6:
                    monthName = "June";
                    break;
                case 7:
                    monthName = "July";
                    break;
                case 8:
                    monthName = "August";
```

```
            break;
        case 9:
            monthName = "September";
            break;
        case 10:
            monthName = "October";
            break;
        case 11:
            monthName = "November";
            break;
        case 12:
            monthName = "December";
            break;
        default:
            monthName = "ERROR - That was not a valid month number!";
            break;
        }
        System.out.println("The month's name is: " + monthName + "\n");
    }
    else {
        System.out.println("That was not a number!  Please run the program again
with a number.\n");
    }
  }
}
```

Copy that code, and paste it into BlueJ. Run it through the debugger a few times, and take a look at its output.

Here's another example that shows how to combine values that have the same action. This example uses strings for the values, and it displays a message based on the day of the week:

```
import java.util.Scanner;
/**
 * SwitchTest2 is an example of a switch structure using strings. It reads
 * a day of the week from the keyboard and then displays a message based on the day.
 *
 * @author Merrill Hall
 * @version 1.0
 */
public class SwitchTest2 {
    public static void main(String[] args) {
        String day = "";
        Scanner in = new Scanner(System.in);

        System.out.print("Enter a day of the week: ");
        day = in.next().toUpperCase();
        switch (day) {
            case "SUNDAY":
            case "SATURDAY":
                System.out.println("It's the weekend! Hooray!");
                break;
            case "MONDAY":
                System.out.println("It's Monday! I'm sorry!");
                break;
            case "FRIDAY":
                System.out.println("Only one day to go!");
                break;
            case "WEDNESDAY":
                System.out.println("You're halfway there!");
                break;
            case "TUESDAY":
            case "THURSDAY":
                System.out.println("Nothing special about today.");
                break;
            default:
                System.out.println("Sorry - that wasn't a valid day of the week.");
                break;
        }
        System.out.println();
    }
}
```

Since any logic that works in a switch block can also be performed using nested if statements, the choice to use a switch instead of ifs is one of design and coding preference. For the most part, a programmer uses switch statements when he or she believes the logic would be clearer and easier to read using a switch rather than a set of ifs.

One advantage of if logic over switches is that an if statement can evaluate far more complex conditions. A switch statement is limited to checking the value of a single variable.

Now You Try

See if you can write a simple program to convert the numbers 1 through 7 to the appropriate name of a day of the week (or to change the names of the days to a number). Please feel free to share your questions with other students in the Discussion Area.

How did you do? Was it helpful to look at my example as you wrote your program?

Let's move on to a different topic in Chapter 4: enumeration.