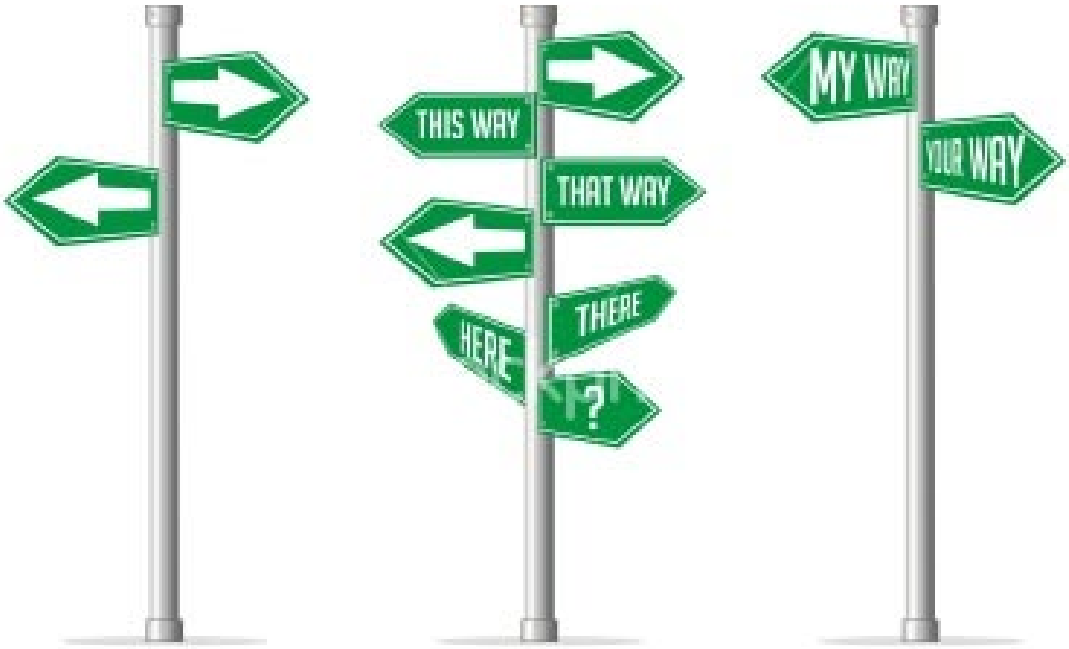
Chapter 2

Decision Making

What is in This Chapter ?

In this chapter we discuss how to write code that makes decisions using the **if** statement and the **switch** statement.



2.1 Using the IF Statement

It is often necessary to be able to make a decision in a program and to act accordingly. For example, assume that we would like to examine a student's final mark in this course and print out an appropriate message. For example:

> **If** the student's grade is less than 50 **then** print out "Too bad, try again next term.", **otherwise** print out "Congratulations!".

You will notice that this sentence is logically made up of 3 parts: (1) the part that checks whether or not the student passed (known as the "***condition statement***") (2) the part that specifies what to do if they do not pass (known as the "***if body***"), and (3) the part that specifies what to do if the student does pass (known as the "***else body***").

We can use the **if** statement in JAVA to make a similar decision like this:

```
if (grade < 50)
    System.out.println("Too bad, try again next term.");
else
    System.out.println("Congratulations!");
```

Notice that the condition statement is between parentheses. Depending on the value of the **grade** variable, the code will print out only one of the two statements, but not both. Do you know why the following code works the same way ?

```
if (grade >= 50)
    System.out.println("Congratulations!");
else
    System.out.println("Too bad, try again next term.");
```

In some cases, when making a decision, we do not need to "do something" in both cases. For example, what if we just wanted to print "Congratulations!" for students who passed, and do nothing in the case that they failed ? In this case, we could leave off the **else** part:

```
if (grade >= 50)
    System.out.println("Congratulations!");
```

In the examples above, we had one line of code to evaluate for each branch of the **if** condition. That is, we only had one statement to print each time. In general, when using an **if** statement, you are allowed to have multiple lines of JAVA code evaluated for each case. When you have more than one line, you need to insert some braces **{ }** after the **if** and **else** as follows …

```
if (grade >= 50) {
    System.out.print("Congratulations! ");
    System.out.print(grade);
    System.out.println(" is a passing grade.");
}
else {
    System.out.print(grade);
    System.out.println(" is quite low. Oh well, there's always next term.");
}
```

It is often a good idea to use the braces anyway, even if you have only one line of code because it may prevent you from making some mistakes. For example, the following code is not the same as above:

```
if (grade >= 50)
    System.out.print("Congratulations! ");
    System.out.print(grade);
    System.out.println(" is a passing grade.");
else
    System.out.print(grade);
    System.out.println(" is quite low. Oh well, there's always next term.");
```

The code above will not compile. Since the brackets are missing, JAVA interprets the code as if there is only one line in the **if** body as follows:

```
if (grade >= 50)
    System.out.print("Congratulations! ");
System.out.print(grade);
System.out.println(" is a passing grade.");
else
    System.out.print(grade);
System.out.println(" is quite low. Oh well, there's always next term.");
```

It then sees the **else** as being out of place … and will give a compile error saying: **'else' without 'if'**. An even worse scenario is when JAVA does not notice the error at all. Consider the following:

```
if (grade >= 50)
    System.out.print("Congratulations! ");
    System.out.print(grade);
    System.out.println(" is a passing grade.");
System.out.println("All Done.");
```

In the above code, a **grade** of 75 will output the following:

```
Congratulations! 75 is a passing grade.
All Done.
```

… and a grade of 25 will output this:

```
25 is a passing grade.
All Done.
```

Clearly this is wrong. Also, be careful not to place a semi-colon **;** after the **if** statement parentheses:

```
if (grade >= 50);
    System.out.println("Congrats! " + grade + " is a passing grade.");
```

In the above code, a **grade** of 25 will output the following:

```
Congrats! 25 is a passing grade.
```

Why ? Because the semi-colon **;** at the end of the first line tells JAVA that there is no body for the **if** statement. Thus, the **System.out.println(…)** line is outside the **if** statement altogether and is therefore always evaluated.

Notice that we used **<** and **>=** in our examples above. These are called ***logical operators*** because they take two values, compare them, and then produce a logical **boolean** result of either **true** or **false**. They are often used to compare numbers. Here is the list of logical operators that we can use:

- **<** less than
- **<=** less than or equal to
- **==** equal to
- **!=** not equal to
- **>=** greater than or equal to
- **>** greater than

When writing an **if** statement, you must make sure that whatever is placed between the parentheses **( )** is a JAVA expression that results in a **boolean** (i.e., the answer is either **true** or **false**).

Example:

Bob and Steve went on a vacation together. During the trip **Bob** paid for all the food and for the hotel. **Steve** paid for the gas and for the entertainment. Write a program to compute the amount of money that **Bob** owes **Steve** (or Steve owes Bob) after the trip, assuming that they decided to split the expenses evenly.

First, we get all the expense information as follows:

```
float       food, hotel, gas, entertainment;
Scanner     keyboard = new Scanner(System.in);

// Get the user input
System.out.println("Enter food expense total: ");
food = keyboard.nextFloat();

System.out.println("Enter hotel bill total: ");
hotel = keyboard.nextFloat();

System.out.println("Enter gasoline expense total: ");
gas = keyboard.nextFloat();

System.out.println("Enter entertainment expense total: ");
entertainment = keyboard.nextFloat();
```

Now we need to compute the amount that each "should have" spent in order to be fair:

```
each = (food + hotel + gas + entertainment) / 2;
```

Then we need to determine who paid more and find the amount owed. We can do this by subtracting what Steve actually paid from what he should have paid as follows:

```
difference = each  - (gas + entertainment);
```

Now, if the difference is more than zero, then Steve owes Bob this difference. If it is less than zero, then Steve paid too much ... so Bob owes Steve. We need to use an **if** statement to check for this and print an appropriate message:

```
if (difference > 0)
     System.out.println("Steve owes Bob $" + difference);
else
     System.out.println("Bob owes Steve $" + difference);
```

However, although this code is correct, it simply displays the final answer and it is not easy to determine if the answer is correct. It may be nice to include some extra print statements that show the breakdown of the calculations as well as formatting the money values properly. Here is the completed program which provides a nice easy-to-read program output:

```
import java.util.Scanner;

public class TripExpenseProgram {
     public static void main(String[] args) {
         float          food, hotel, gas, entertainment, each, difference;
         Scanner        keyboard = new Scanner(System.in);

         // Get the user input
         System.out.println("Enter food expense total: ");
         food = keyboard.nextFloat();

         System.out.println("Enter hotel bill total: ");
         hotel = keyboard.nextFloat();

         System.out.println("Enter gasoline expense total: ");
         gas = keyboard.nextFloat();

         System.out.println("Enter entertainment expense total: ");
         entertainment = keyboard.nextFloat();

         // Calculate the difference
         each = (food + hotel + gas + entertainment) / 2;
         difference = each  - (gas + entertainment);

         // These four lines are not necessary, but they give a breakdown
         // that will help verify whether or not the result is correct.
         System.out.println("\nThe total expenses are " +
                            String.format("$%,1.2f", each * 2));
         System.out.println("Each person should pay " +
                            String.format("$%,1.2f", each));

         System.out.println("Steve paid " +
                            String.format("$%,1.2f", (gas + entertainment)));
         System.out.println("Bob paid   " +
                            String.format("$%,1.2f", (food + hotel)));

         // Decide who owes who and then display the results
         if (difference > 0)
            System.out.print("\nTherefore, Steve owes Bob ");
         else
            System.out.print("\nTherefore, Bob owes Steve ");

         System.out.println(String.format("$%,1.2f", difference));

     }
}
```

Here is what the output will look like ... notice how nicely it reads. Try to do this with your programs in the course so that correct results can be verified just by reading the program's output:

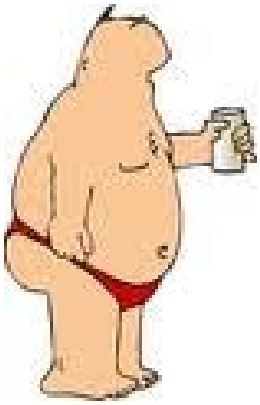
```
Enter food expense total:
536.92
Enter hotel bill total:
1282.13
Enter gasoline expense total:
78.45
Enter entertainment expense total:
389.21

The total expenses are $2,286.71
Each person should pay $1,143.35
Steve paid $467.66
Bob paid   $1,819.05

Therefore, Steve owes Bob $675.70
```

Example:

Assume that you want to take a vote among **5** friends to find out whether or not they agree to some issue (e.g., like not wearing speedos at a pool party). Each person votes yes or no. Write a program that determines the majority response (either yes or no).



To begin the program, consider first getting all 5 votes and storing them. Then use the votes to determine the majority.

```
char       v1, v2, v3, v4, v5;
Scanner   keyboard = new Scanner(System.in);

// Get the user input
System.out.println("Enter vote 1 (y or n): ");
v1 = keyboard.next().charAt(0);

System.out.println("Enter vote 2 (y or n): ");
v2 = keyboard.next().charAt(0);

System.out.println("Enter vote 3 (y or n): ");
v3 = keyboard.next().charAt(0);

System.out.println("Enter vote 4 (y or n): ");
v4 = keyboard.next().charAt(0);

System.out.println("Enter vote 5 (y or n): ");
v5 = keyboard.next().charAt(0);
```

Now we need to count up the yes votes and the no votes. We can use one variable to count the yes votes and another to count the no votes and then compare them to make our decision:

```
int   yesCount = 0;
int   noCount = 0;

if (v1 == 'y')
      yesCount++;
else
      noCount++;
if (v2 == 'y')
      yesCount++;
else
      noCount++;
if (v3 == 'y')
      yesCount++;
else
      noCount++;
if (v4 == 'y')
      yesCount++;
else
      noCount++;
if (v5 == 'y')
      yesCount++;
else
      noCount++;
if (yesCount > noCount)
      System.out.println("The majority voted YES");
else
      System.out.println("The majority voted NO");
```

Can you think of another way of doing this by using only one counter and only one variable for user input ?

Well, we can start with a count of zero and then increase it when we get a YES vote and decrease when we get a NO vote. Notice what would happen:

count starts at 0		**count** starts at 0	
vote 1 = yes	**count** increases to 1	vote 1 = no	**count** decreases to -1
vote 2 = yes	**count** increases to 2	vote 2 = no	**count** decreases to -2
vote 3 = no	**count** decreases to 1	vote 3 = no	**count** decreases to -3
vote 4 = yes	**count** increases to 2	vote 4 = yes	**count** increases to -2
vote 5 = no	**count** decreases to 1	vote 5 = yes	**count** decreases to -1

Once we have all the votes, we can check if the **count** is positive. In that case, the majority noted YES. If the **count** is negative, the majority votes NO. If the **count** is zero (cannot happen when number of voters is odd though), then there is a TIE. Here is the resulting code that uses only one count variable:

```
import java.util.Scanner;

public class MajorityProgram {
     public static void main(String[] args) {
          char     vote;
          int      count = 0;

          Scanner  keyboard = new Scanner(System.in);

          // Get the user input and apply to the total
          System.out.println("Enter vote 1 (y or n): ");
          vote = keyboard.next().charAt(0);
          if (vote == 'y')
             count++;
          else
             count--;
          System.out.println("Enter vote 2 (y or n): ");
          vote = keyboard.next().charAt(0);
          if (vote == 'y')
             count++;
          else
             count--;
          System.out.println("Enter vote 3 (y or n): ");
          vote = keyboard.next().charAt(0);
          if (vote == 'y')
             count++;
          else
             count--;
          System.out.println("Enter vote 4 (y or n): ");
          vote = keyboard.next().charAt(0);
          if (vote == 'y')
             count++;
          else
             count--;
          System.out.println("Enter vote 5 (y or n): ");
          vote = keyboard.next().charAt(0);
          if (vote == 'y')
             count++;
          else
             count--;
          if (count > 0)
             System.out.println("The majority voted YES");
          else
             System.out.println("The majority voted NO");

     }
}
```

We will see later that we can shorten this code a lot more ...

Example:

Let us write a program that displays the following menu.

```
Luigi's Pizza
-------------------------------------------
                     S(SML)  M(MED)  L(LRG)
1. Cheese             5.00    7.50   10.00
2. Pepperoni          5.75    8.63   11.50
3. Combination        6.50    9.75   13.00
4. Vegetarian         7.25   10.88   14.50
5. Meat Lovers        8.00   12.00   16.00
```

The program should then prompt (i.e., *get input from*) the user for the type of pizza he/she wants to order (i.e., 1 to 5) and then the size of the pizza 'S', 'M' or 'L'. Then the program should compute and display the cost of the pizza with 13% HST added.

To begin, we need to define a class to represent the program and display the menu:

```
public class LuigisPizzaProgram {
      public static void main(String[] args) {
            System.out.println("Luigi's Pizza                                ");
            System.out.println("-------------------------------------------");
            System.out.println("                     S(SML)  M(MED)  L(LRG)");
            System.out.println("1. Cheese             5.00    7.50   10.00 ");
            System.out.println("2. Pepperoni          5.75    8.63   11.50 ");
            System.out.println("3. Combination        6.50    9.75   13.00 ");
            System.out.println("4. Vegetarian         7.25   10.88   14.50 ");
            System.out.println("5. Meat Lovers        8.00   12.00   16.00 ");
      }
}
```

We can then get the user input and store it into variables:

```
Scanner keyboard = new Scanner(System.in);

System.out.println("What kind of pizza do you want (1-5) ?");
int kind = keyboard.nextInt();

System.out.println("What size of pizza do you want (S, M, L) ?");
char size = keyboard.next().charAt(0);
```

Now that we have the **kind** and **size**, we can compute the total cost. Notice that the cost of a small pizza increases by $0.75 as the kind of pizza increases. Also, you may notice that the cost of a medium is 1.5 x the cost of a small and the cost of a large is 2 x a small. So we can compute the cost of any pizza based on its kind and size by using a single mathematical formula. Can you figure out the formula ?

A small pizza would cost: **smallCost** = $4.25 + (**kind** x $0.75)

A medium pizza would cost: **mediumCost** =**smallCost** * 1.5
A large pizza would cost: **largeCost** =**smallCost** * 2.

Can you write the code now ?

```
float cost = 4.25f + (kind * 0.75f);
if (size == 'M')
      cost *= 1.5f;
else if (size == 'L')
      cost *= 2;
```

And of course, we can then compute and display the cost before and after taxes. Here is the completed program:

```
import java.util.Scanner;

public class LuigisPizzaProgram {
      public static void main(String[] args) {
            System.out.println("Luigi's Pizza                                          ");
            System.out.println("---------------------------------------------");
            System.out.println("                      S(SML)  M(MED)  L(LRG)");
            System.out.println("1. Cheese              5.00    7.50   10.00 ");
            System.out.println("2. Pepperoni           5.75    8.63   11.50 ");
            System.out.println("3. Combination         6.50    9.75   13.00 ");
            System.out.println("4. Vegetarian          7.25   10.88   14.50 ");
            System.out.println("5. Meat Lovers         8.00   12.00   16.00 ");

            Scanner keyboard = new Scanner(System.in);

            System.out.println("What kind of pizza do you want (1-5) ?");
            int kind = keyboard.nextInt();

            System.out.println("What size of pizza do you want (S, M, L) ?");
            char size = keyboard.next().charAt(0);

            float cost = 4.25f + (kind * 0.75f);
            if (size == 'M')
                  cost *= 1.5f;
            else if (size == 'L')
                  cost *= 2;

            System.out.println("The cost of the pizza is: " +
                               String.format("$%,1.2f", cost));
            System.out.println("The price with tax is: " +
                               String.format("$%,1.2f", cost * 1.13));
      }
}
```

Example:

Consider writing a program that will be placed at a kiosk in front of a bank to allow customers to determine whether or not they qualify for the bank's new "Entrepreneur Startup Loan".

Assume that this kind of loan is only given out to:

(1) someone who is currently employed AND who is a recent University graduate, or ..

(2) someone who is employed AND over 30 years of age AND has at least 10 years of full-time work experience.

The program should display information to the screen as well as ask the user various questions … and then determine if the person qualifies.

Here is a start to the program that displays some opening instructions:

```
import java.util.Scanner;

public class LoanQualificationProgram {
    public static void main(String[] args) {
        // Get a Scanner object for user input
        Scanner   keyboard = new Scanner(System.in);

        // Display some opening instructions
        System.out.println("Bank of Java");
        System.out.println("============");
        System.out.println("Follow the instructions below to " +
                           "determine whether or not you qualify " +
                           "for a Entrepreneur Startup Loan...\n");
    }
}
```

Now we need to start asking the user some questions. We can first ask whether or not he/she is employed. Likely we will ask for a character such as '**y**', '**Y**', '**n**' or '**N**'. We can then check the input and store the employment status as a **boolean**. Here is the new code to add:

```
char        charInput;
boolean     employed;

// Determine whether or not the user is employed
System.out.print("Are you currently employed (Y/N)?  ");
charInput = keyboard.nextLine().charAt(0);

if ((charInput == 'y') || (charInput == 'Y'))
    employed = true;
else
    employed = false;
```

Interestingly, we can shorten this code by realizing that the employed variable is a boolean variable and the condition of the IF statement is also a boolean. So we can replace this:

```
if ((charInput == 'y') || (charInput == 'Y'))
    employed = true;
else
    employed = false;
```

with this ... which does the same thing and is more concise:

```
employed = (charInput == 'y') || (charInput == 'Y');
```

We can similarly ask whether or not they have a recent University degree:

```
boolean     hasDegree;

// Determine if the user has a recent University degree
System.out.print("Did you graduate with a University degree " +
                 "in the past 6 months (Y/N)?  ");
charInput = keyboard.nextLine().charAt(0);

hasDegree = (charInput == 'y') || (charInput == 'Y');
```

In a similar manner, we can ask for the user's age and the number of years that they have worked at full time status:

```
int   age, yearsWorked;

// Determine the user's age
System.out.print("How old are you ?  ");
age = keyboard.nextInt();

// Determine the number of years worked at full time status
System.out.print("How many years have you been working " +
                 "at full time status ?  ");
yearsWorked = keyboard.nextInt();
System.out.println("\n");
```

Finally, we can determine whether or not they qualify:

```
char        charInput;
boolean     employed, hasDegree;
int         age, yearsWorked;

...



// Now determine whether or not the person qualifies for the loan
if (employed) {
```

```
    if (hasDegree)
        System.out.println("Congratulations! You qualify " +
                           "for the ESL loan.");
    else {
        if (age >= 30) {
            if (yearsWorked >= 10)
                System.out.println("Congratulations! You qualify " +
                                   "for the ESL loan.");
            else
                System.out.println("Sorry, you must have worked " +
                                   "at least 10 years at full " +
                                   "time status to qualify.");
        }
        else
            System.out.println("Sorry, you must be a recent " +
                               "University graduate or be at " +
                               "least 30 years of age.");
    }
}
else {
    System.out.println("Sorry, you must be currently " +
                       "employed to qualify.");
}
```

Here is the finished code:

```
import java.util.Scanner;

public class LoanQualificationProgram {
    public static void main(String[] args) {
        char        charInput;
        boolean     employed, hasDegree;
        int         age, yearsWorked;


        // Get a Scanner object for user input
        Scanner keyboard = new Scanner(System.in);

        // Display some opening instructions
        System.out.println("Bank of Java");
        System.out.println("============");
        System.out.println("Follow the instructions below to " +
                           "determine whether or not you qualify " +
                           "for a Entrepreneur Startup Loan...\n");

        // Determine whether or not the user is employed
        System.out.print("Are you currently employed (Y/N)?  ");
        charInput = keyboard.nextLine().charAt(0);
        employed = (charInput == 'y') || (charInput == 'Y');

        // Determine if the user has a recent University degree
        System.out.print("Did you graduate with a University degree " +
                         "in the past 6 months (Y/N)?  ");
        charInput = keyboard.nextLine().charAt(0);
        hasDegree = (charInput == 'y') || (charInput == 'Y');

        // Determine the user's age
        System.out.print("How old are you ?  ");
```

```
        age = keyboard.nextInt();

        // Determine the number of years worked at full time status
        System.out.print("How many years have you been working " +
                         "at full time status ?  ");
        yearsWorked = keyboard.nextInt();
        System.out.println("\n");

        // Now determine whether or not the person qualifies for the loan
        if (employed) {
            if (hasDegree)
                System.out.println("Congratulations! You qualify " +
                                   "for the ESL loan.");
            else {
                if (age >= 30) {
                    if (yearsWorked >= 10)
                        System.out.println("Congratulations! You qualify " +
                                           "for the ESL loan.");
                    else
                        System.out.println("Sorry, you must have worked " +
                                           "at least 10 years at full " +
                                           "time status to qualify.");
                }
                else
                    System.out.println("Sorry, you must be a recent " +
                                       "University graduate or be at " +
                                       "least 30 years of age.");
            }
        }
        else {
            System.out.println("Sorry, you must be currently " +
                               "employed to qualify.");
        }
    }
}
```

And here is some sample output:

```
Bank of Java
============
Follow the instructions below to determine whether or not you qualify for a
Entrepreneur Startup Loan...

Are you currently employed (Y/N)?  Y
Did you graduate with a University degree in the past 6 months (Y/N)?  N
How old are you ?  38
How many years have you been working at full time status ?  12


Congratulations! You qualify for the ESL loan.
```

Example:

Consider now an example in which we take the number grade of a student (i.e., from **0%** to **100%**) and output a letter grade (from **F** to **A+**). How would we do this ? Well … we would need to understand which letter grade corresponds to which number grades as follows:

A = 80% - 100%	D = 50% - 59%
B = 70% - 79%	F = 0% - 49%
C = 60% - 69%	

So, given a grade, how can we output the appropriate letter ? We could use the **if** statement:

```
if (grade >= 80)
    System.out.println("A");
else {
    if (grade >= 70)
        System.out.println("B");
    else {
        if (grade >= 60)
            System.out.println("C");
        else {
            if (grade >= 50)
                System.out.println("D");
            else
                System.out.println("F");
        }
    }
}
```

You may notice now that we have an **if** statement inside of an **else** statement's body. This is known as ***nested if*** statements. Notice how the code is indented carefully so that when reading the code we can see what is inside each **if**/**else** statement's body.

In fact, an **if**/**else** statement is actually considered a single JAVA statement. So, we do not need the braces here. In fact, we can even place the succeeding **if** statements up on the same line as the **else** statements and align everything up on the left. The following is how we usually write such code:

```
if (grade >= 80)
    System.out.println("A");
else if (grade >=70)
    System.out.println("B");
else if (grade >= 60)
    System.out.println("C");
else if (grade >= 50)
    System.out.println("D");
else
    System.out.println("F");
```

Example:

Consider another example in which we are given an integer representing a **month** and we would like to store (in a variable called **days**) the number of days in that month (we will assume that it is not a leap year). Here is the table of information that we need to know:

Month	**Jan**	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**
Days	31	28	31	30	31	30	31	31	30	31	30	31

Here is how we can do this with **if** statements:

```
int month, days;

month =  ... // assume that we got this from the user

if (month == 1)
    days = 31;
else if (month == 2)
    days = 28;
else if (month == 3)
    days = 31;
else if (month == 4)
    days = 30;
//... etc ...
```

However, you can see that the **if** statement will be 24 lines long! Since there are only 3 values for the months (i.e., 31, 30 and 28 (ignore leap year for now)), there should be a way to arrange it in a format like this …

```
int month, days;

month =  ...      // assume that we got this from the user
if (...)          // Jan, Mar, May, Jul, Aug, Oct, Dec
    days = 31;
else if (...)     // Apr, Jun, Sep, Nov
    days = 30;
else if (...)     // Feb only
    days = 28;
```

To do this, we need to have some way of asking whether or not the month is Jan **or** Mar **or** May **or** Jul etc… Well, it is a good thing then that JAVA supplies us with what is known as ***Boolean operators***. Here are three useful **boolean** operators:

- `&&` conditional **and**
- `||` conditional **or**
- `!` **not** (prefix)

The **&&** and **||** operators are used in-between two JAVA expressions that evaluate to booleans. Below is a table explaining the results of using these two boolean values **b1**and **b2** in various expressions:

b1	b2	if (b1 && b2)	if (b1 || b2)	if (!b1)	if (b1)
false	false	false	false	**true**	false
false	**true**	false	**true**	**true**	false
true	false	false	**true**	false	**true**
true	**true**	**true**	**true**	false	**true**

Notice that the **&&** results in **true** only when both booleans are **true**, and **false** otherwise. Conversely, the **||** results in **false** only when both booleans are **false**, and **true** otherwise. Also note that the **!** results in the opposite value of the boolean.

We can actually combine multiple booleans operators together in various ways:

```
if ((boolean1 && boolean2 && boolean3) || (boolean1 && !boolean2)) …
```

So how do we use this in our month/days example ? Well, each boolean expression can be in a format something like this: **(month == 2)**. Therefore, our solution may look like this:

```
if ((month == 1) || (month == 3) || (month == 5) || (month == 7) ||
    (month == 8) || (month == 10) || (month == 12))
    days = 31;
else if ((month == 4) || (month == 6) || (month == 9) || (month == 11))
    days = 30;
else if (month == 2)
    days = 28;
```

So we just used a bunch of “or” operators. We can actually simplify the code by noticing something interesting. Since the first two **if** statements checked 11 of the 12 months, then we do not need to ask **if (month == 2)** in the last if statement because it is the only possible month remaining (assuming that the month was in the valid range of 1 to 12). Also, it does not matter which order we check the months in. So, the following code will also do the same thing, but is much shorter:

```
if (month == 2)
    days = 28;
else if ((month == 4) || (month == 6) || (month == 9) || (month == 11))
    days = 30;
else
    days = 31;
```

As one more example, how would we use an **if** statement that prints out the message “valid” when a given **number** is within a given range (e.g., from 1 to 10) and “invalid” otherwise ?

Here is a common mistake that many students make:

```
int number = ... ; //code left out intentionally

if (1 <= number <= 10)
    System.out.println("Valid");
else
    System.out.println("Invalid");
```

JAVA does not allow us to check ranges in this manner. Instead, we have to check the two sides of the range separately.

```
int number = ... ; //code left out intentionally

if ((number >= 1) && (number <= 10))
    System.out.println("Valid");
else
    System.out.println("Invalid");
```

Although **if** statements are quite easy to use, it is often the case that students do not **fully** understand how to use **boolean** logic. As a result, sometimes students end up writing overly complex and inefficient code ... sometimes even using an **if** statement when it is not even required!

To illustrate this, consider the following examples of "BAD" coding style. Try to determine why the code is inefficient and how to improve it. If it is your desire to be a good programmer, pay careful attention to these examples.

Example 1:

```
boolean   male = ...;

if (male == true) {
   System.out.println("male");
else
   System.out.println ("female");
```

Here, the **boolean** value of **male** is *already* **true** or **false**, we can make use of this fact:

```
boolean   male = ...;

if (male) {
   System.out.println ("male");
else
   System.out.println ("female");
```

Example 2:

```
boolean   adult = ...;

if (adult == false)
    discount = 3.00;
```

Here is a similar situation as above, but with a negated **boolean**. Below is better code.

```
boolean   adult = ...;

if (!adult) {
    discount = 3.00;
```

Example 3:

```
boolean   tired = ...;

if (tired)
    result = true;
else
    result = false;
```

Above, we are actually returning the identical **boolean** as **tired**. No **if** statement is needed:

```
boolean   tired = ...;

result = tired;
```

Example 4:

```
if ((age < 6) || (age > 65))
    discount = true;
else
    discount = false;
```

The discount is solely determined by the **age**. No **if** statement is needed:

```
discount = (age < 6) || (age > 65);
```

Example5:

```
if ((age < 6) || (age > 65))
    fullPrice = false;
else
    fullPrice = true;
```

Just like above, we do not need the **if** statement:

```
fullPrice = !((age < 6) || (age > 65));

// or …

fullPrice = (age >= 6) && (age <= 65);
```

Supplemental Information: The Selection Operator

In addition to the **if** statement, JAVA allows a ***Selection Operator*** to be used. Here is the general format:

<booleanCondition> ? <valueIfTrue> : <valueIfFalse>

The result of the expression is one of the two values supplied, depending on whether the condition is **true** or **false**. For example,

```
String result;

result = grade > 50 ? "pass" : "fail";
```

does the same thing as:

```
String result;

if (grade > 50)
    result = "pass";
else
    result = "fail";
```

So this saves the amount of code to write, but it does sacrifice a little bit of *readability* of the code.

2.2 The Switch Statement

In addition to the **if** statement, there is another construct in JAVA called the **switch** statement which is beneficial for simplifying code that contains *nested* **if** statements.

Example:

Consider a simplified letter grade that is given for a course project (i.e., **A**, **B**, **C**, **D**, **F**). (i.e., for the purpose of brevity, we will assume that there is no **A+**, **A-**, **B+**, **B-** ... grades). Sometimes when a student receives a letter grade he/she would like to know what percentage range corresponds to that letter. For example, a **B** corresponds to a grade between 70% and 79%.

Consider code that uses **if** statements to compute the proper range as follows …

```
char        aLetter;

aLetter = ...;  // the code for obtaining the grade has been omitted

if (aLetter == 'A')
    System.out.println("80% - 100%");
else if (aLetter == 'B')
    System.out.println("70% - 79%");
else if (aLetter == 'C')
    System.out.println("60% - 69%");
else if (aLetter == 'D')
    System.out.println("50% - 59%");
else if (aLetter == 'F')
    System.out.println("0% - 49%");
else
    System.out.println("not defined");
```

Looking at the code, we can see that there are 5 **if** statements and it looks very messy. If we later decide to handle the A+, A-, B+, etc.. cases, then the code will look much longer and cluttered. There is a better way to write this code. We can use a **switch** statement. The **switch** statement is typically used in situations where we have a sequence of nested **if** statements in which only one of the **if** statements is to be executed.

The switch statement has the format shown here on the right:

```
switch (aPrimitiveExpression) {
    case val1:
        /* 1 or more lines of code*/;
        break;
    case val2:
        /* 1 or more lines of code*/;
        break;
    ...
    case valN:
        /* 1 or more lines of code*/;
        break;
    default:
        /* 1 or more lines of code*/;
        break;
}
```

In the above code, **aPrimitiveExpression** is either a variable or any JAVA code that results in one of the following types:

- a primitive variable of type **int**, **char**, **byte**, or **short**
- a **String** type
- an **Enum type** (discussed later)

The values of **val1**, **val2**, …, **valN** must all be primitive constant values of the same type as **aPrimitiveExpression**.

The **switch** statement works as follows:

1. It evaluates **aPrimitiveExpression** to obtain a value (the expression MUST result in a primitive data type, it **cannot** be an object).

2. It then checks the values **val1**, **val2**, …, **valN** in order from top to bottom until a value is found equal to the value of **aPrimitiveExpression**. If none match, then the **default** case is executed.

3. It then executes the statements corresponding to the **case** whose value matched.

4. If there is a **break** at the end of the lines of JAVA code for that **case**, then the **switch** statement quits. Otherwise it continues to evaluate all the successive **case** statements that follow ... until a **break** is found or until no more cases remain.

Here is how can we make use of the switch statement for solving the grade range problem ?

```
import java.util.Scanner;

public class GradingSwitchProgram {
     public static void main(String[] args) {
         char           letter;
         Scanner        keyboard = new Scanner(System.in);

         // Get the user input
         System.out.print("Enter the grade: ");
         letter = keyboard.next().charAt(0);

         switch(letter) {
            case 'A': System.out.println("80% - 100%"); break;
            case 'B': System.out.println("70% - 79%"); break;
            case 'C': System.out.println("60% - 69%"); break;
            case 'D': System.out.println("50% - 59%"); break;
            case 'F': System.out.println("0% - 49%"); break;
            default:  System.out.println("not defined");
         }
     }
}
```

We can clearly see that the code is simpler to read. However, this is not the only advantage of a **switch** statement.

Consider our previous example in which we were given an integer representing a month and we would like to know the number of days in that month:

```
if (month == 2)
    days = 28;
else if ((month == 4) || (month == 6) || (month == 9) || (month == 11))
    days = 30;
else
    days = 31;
```

Here is how we can use a **switch** statement …

```
switch(month) {
    case 2: days = 28; break;
    case 4:
    case 6:
    case 9:
    case 11: days = 30; break;
    default: days = 31;
}
```

Note that when the month is 4, 6, 9, or 11, then **days = 30**; is evaluated. The code is not necessarily much shorter, but it is simpler to read. This is the main advantage of a **switch** statement.

One thing that needs mentioning is that the value of the cases must be **literal** value. That is, they cannot be expressions nor ranges. Nor can we make use of the logical operators such as **and**'ing and **or**'ing.

So these two examples will not work:

```
switch (age) {
    case 1-12:  price = 5.00; break;  // Won't compile
    case 13-17: price = 8.00; break;  // Won't compile
    case 18-54: price = 10.00; break; // Won't compile
    default:    price = 6.00;
}
```

```
switch (age) {
    case < 12:  discount = 100; break; // Won't compile
    case > 65:  discount = 50;  break; // Won't compile
    default:    discount = 0;
}
```