Decisions in Java – Nested IF Statements

Several Actions – The Nested `if` Statement

We have already explored using the if statement to choose a single action (vs no action), or to choose between two actions. It is also possible to choose between several actions (3 or more) using the if statement.

The most basic, and straightforward, way to accomplish this is using multiple if statements.

Example 1 – Determine whether a value is negative, positive, or zero.

```
if (x < 0)
{
      System.out.println(x + " is negative");
}
if (x > 0)
{
      System.out.println(x + " is positive");
}
if (x == 0)
{
      System.out.println(x + " is zero");
}
```

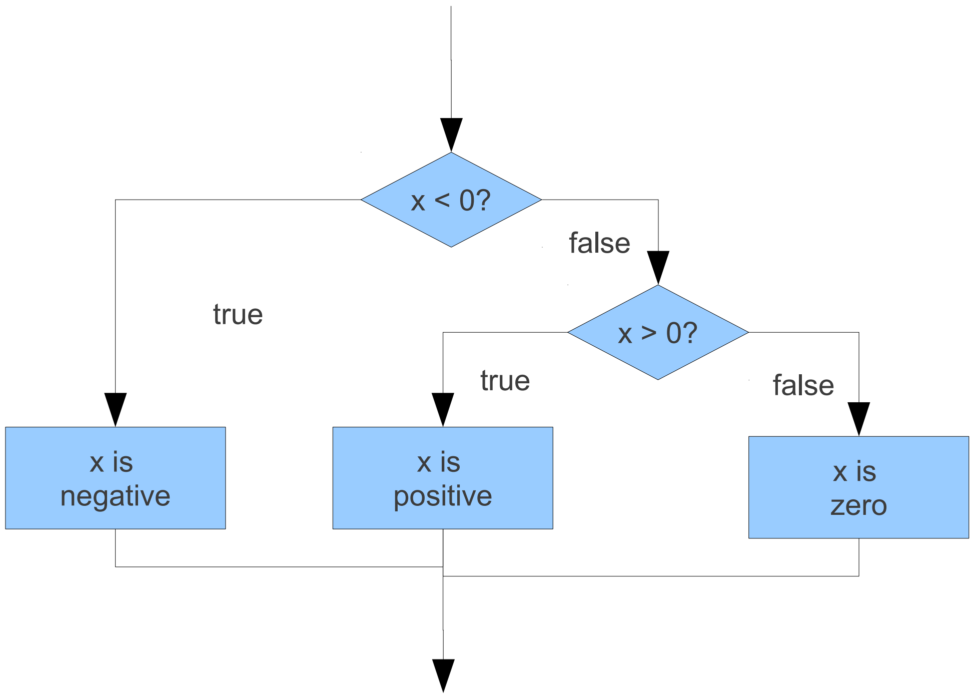
Although this code works, it isn't very efficient because it always performs three comparisons. A more efficient way of writing the same code involves putting one if statement inside another.

```
if (x < 0)
{
      System.out.println(x + " is negative");
}
else
{
      if (x > 0)
      {
            System.out.println(x + " is positive");
      }
      else
      {
            System.out.println(x + " is zero");
      }
}
```

This code takes some getting used to, but it is more efficient. When x is negative, only one comparison is needed (is x < 0?). If the first comparison is false, only one more is required to distinguish between positive and zero, so there will be a maximum of two comparisons. It seems trivial when the programming is this simple, but for programs running thousands of lines of code, millions of times over, it can make the difference between fast and slow performance.

Decisions in Java – Nested IF Statements

As a flowchart, the nested if statement has extra branches.



Example 2 – Given three values, x, y, and z, determine the largest value and assign this value to the variable **largest**.

```
if (x >= y)
{
      // y is eliminated, largest must be x or z
      if (x >= z)
      {
            largest = x;
      }
      else
      {
            largest = z;
      }
}
else
{
      // x is eliminated, largest must be y or z
      if (y >= z)
      {
            largest = y;
      }
      else
      {
            largest = z;
      }
}
```

In the examples shown so far, we have followed the indentation pattern consistent with our previous code. As nesting gets deeper and deeper, our code will continue to shift to the right, which threatens to make our code unreadable. To address this issue, Java offers a more condensed option which makes better use of space, and is also easy to read.

Example 3 – This fragment of code prints a message depending upon the letter grade contained in a **char** variable called **grade**.

```
if (grade == 'A')
{
      System.out.println("Excellent!");
}
else if (grade == 'B')
{
      System.out.println("Good!");
}
else if (grade == 'C')
{
      System.out.println("Average.");
}
else if (grade == 'D')
{
      System.out.println("Poor.");
}
else if (grade == 'F')
{
      System.out.println("Need help!");
}
else
{
      System.out.println("Invalid grade.");
}
```

Decisions in Java – Nested IF Statements

Exercises

1. Simplify the following sequence by nesting so that the effect is the same, but fewer comparisons are required. You may want to incorporate this code into a full program for testing purposes.

```
if (temperature > maxTemp)
{
     System.out.println("Porridge is too hot.");
}
if (temperature < minTemp)
{
     System.out.println("Porridge is too cold.");
}
if (temperature >= minTemp && temperature <= maxTemp)
{
     System.out.println("Porridge is just right.");
}
```

2. Consider the following statement:

```
if (age < minAge)
{
     if (income > minIncome)
     {
          System.out.println("Accepted");
     }
     else
     {
     System.out.println("Rejected");
     }
}
```

   What will the statement print if

   (a) `age > minAge` and `income < minIncome`?
   (b) `age < minAge` and `income < minIncome`?

3. Using nested if statements, write a fragment of code that prints the **smallest value** contained in the variables **a**, **b**, and **c**. Create a program to test your code fragment.

4. Write a fragment of code that tests the value of the variable **item**. If the value is negative, it adds the value to **negativeSum**. If the value is positive, it adds the value to **positiveSum**. If the value is zero, increase the variable **zeroCount** by one. Create a program to test your code fragment.

5. A typical program will ask the user, "Are you sure you wish to continue?", and then prompt them for a 'y' or 'n' (char) response. Write a program that asks the user a suitable question and prompts them for a single character response, 'y' or 'n'. Also have your program handle an invalid response (anything other than 'y' or 'n'). Try to have your program handle upper or lower case 'Y' or 'N' as well.

6. Assume that the following declarations have been made:

   ```
   int year;
   boolean isLeapYear;
   ```

   Write a fragment that will assign **isLeapYear** to **true** if **year** represents a leap year and **false** otherwise.

   Note: The simplest definition of a leap year is any year that is divisible by four. For a challenge, you could research the full definition of a leap year and create a fragment to detect a proper leap year.

Solutions

1. Notice that the last condition has been completely removed and replaced with a simple "else" statement. The last "else" is the "catch all" action if all of the previous conditions fail. In this case, porridge that is neither too hot nor too cold must be just right.

```
if (temperature > maxTemp)
{
     System.out.println("Porridge is too hot.");
}
else if (temperature < minTemp)
{
     System.out.println("Porridge is too cold.");
}
else
{
     System.out.println("Porridge is just right.");
}
```

2. a) Nothing is printed. The first comparison will be false. Since there is no "else" attached to the first comparison, the program does nothing.

   b) The output is "Rejected". The first comparison is true, which means the second comparison must be tested. In the second comparison, the result is false, so the "else" action is performed.

3. This program is very similar to the example provided in the note (see above) for determining the greatest value. It is recommended to start with that code and modify it for this new purpose. Changes to the "greatest value" example are highlighted.

```
if (a <= b)
{
      // b is eliminated, smallest must be a or c
      if (a <= c)
      {
            smallest = a;
      }
      else
      {
            smallest = c;
      }
}
else
{
      // a is eliminated, smallest must be b or c
      if (b <= c)
      {
            smallest = b;
      }
      else
      {
            smallest = c;
      }
}
```

4. The code fragment is shown below. For testing, copy your fragment (or this fragment) into a program that already works. Don't forget to declare all of your variables. You will need to program some input for the variable "item" – recommend an integer data type.

```
if (item < 0)
{
      negativeSum = negativeSum + item;
}
else if (item > 0)
{
      positiveSum = positiveSum + item;
}
else
{
      zeroCount = zeroCount + 1;
}
```

5. The simplest program will choose between 'y' or 'n' or something else. A couple of extra lines allows us to also check for 'Y' or 'N', the uppercase responses. Later, we will learn to combine the upper and lower case responses in a cleaner, more efficient way.

```
System.out.println("Are you sure you wish to continue?");
char response = In.getChar();

if (response == 'y')
{
     System.out.println("Answer is YES");
}
else if (response == 'Y')
{
     System.out.println("Answer is YES");
}
else if (response == 'n')
{
     System.out.println("Answer is NO");
}
else if (response == 'N')
{
     System.out.println("Answer is NO");
}
else
{
     System.out.println("Invalid response!");
}
```

6. This is the simple version. Recall that the modulo operator (%) can be used to find the remainder from integer division. If a number is divisible by 4, then dividing by 4 should give a remainder of zero.

```
boolean leapYear;
System.out.println("Please enter a year");
int year = In.getInt();

if (year % 4 == 0)
{
     isLeapYear = true;
}
else
{
     isLeapYear = false;
}
```