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.

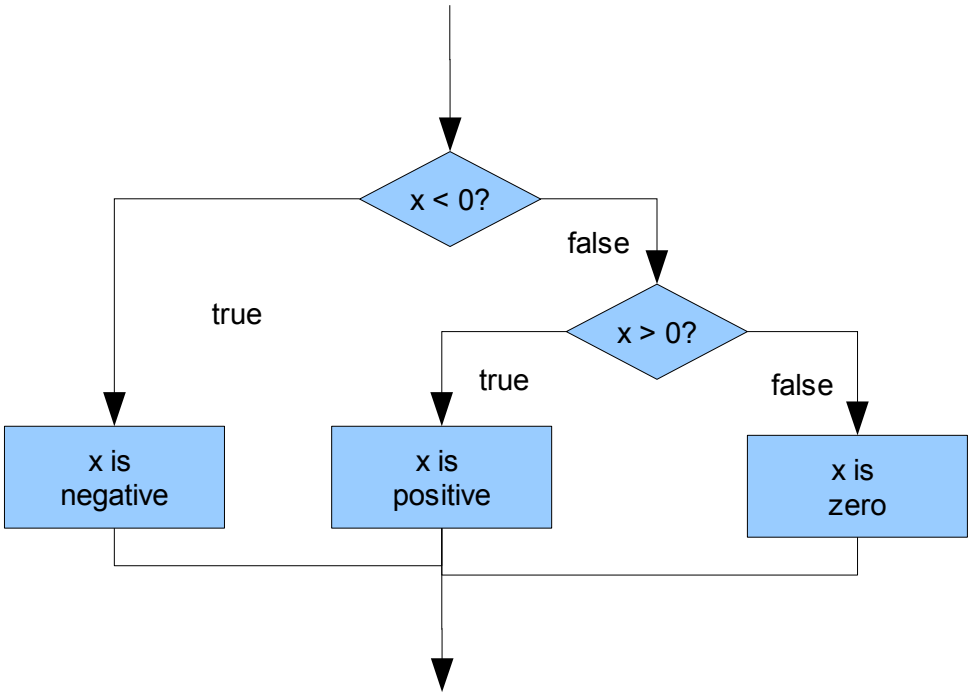
Example 1 – Determine whether a value is negative, positive, or zero.

```
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.



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;
      }
}
```

Decisions in Java – Nested IF Statements

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.");
}
```

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**.

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.

5. 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.
```
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
   (b) "Rejected" is printed

3.
```
if (a < b)
{
      // the smallest must be a or c
      if (a < c)
      {
            System.out.println(a + " is the smallest value");
      }
      else
      {
            System.out.println(c + " is the smallest value");
      }
}
else
{
      // the smallest must be b or c
      if (b < c)
      {
            System.out.println(b + " is the smallest value");
      }
      else
      {
            System.out.println(c + " is the smallest value");
      }
}
```

4.
```
if (item < 0)
{
      negativeSum = negativeSum + item;
}
else if (item > 0)
{
      positiveSum = positiveSum + item;
}
else
{
      zeroCount = zeroCount + 1;
}
```