Decisions in Java –Boolean Values & Expressions

Boolean Values & Variables

In order to make decisions, Java uses the concept of `true` and `false`, which are *boolean values*. Just as is the case with other primitive data types, we can create boolean variables to hold these values.

```
boolean readyToProgram = true;
```

Boolean Expressions

A *boolean expression* is similar to a mathematical expression, except that the result is `true` or `false`, rather than a numeric value. To create a boolean expression, we use the relational operators to compare the values of various data types, such as integers, floats, characters, or strings, using a *relational expression*.

Relational Operator	Meaning	Example	Result
`==`	is equal to	`5 == 5`	`TRUE`
`!=`	is not equal to	`5 != 5`	`FALSE`
`<`	is less than	`3 < 7`	`TRUE`
`<=`	is less than or equal to	`4 <= 4`	`TRUE`
`>`	is greater than	`3 > 7`	`FALSE`
`>=`	is greater than or equal to	`7 >= 3`	`TRUE`

The following rules apply to the use of relational operators with different data types:

1. Values of any of the primitive numeric data types (e.g., int, float, and all their variations) can be used with any of the relational operators.
2. Boolean values can only be tested as “equal to” or “not equal to”.
3. Values of type char are ordered according to the Unicode encoding system. A character the occurs earlier in the system is “less than” a character that occurs later in the system. You can research full details of the Unicode system online.
   a) For alphabetic characters, this means that 'a' is less than 'z', and 'A' is less than 'Z', as expected.
   b) In the Unicode system, all uppercase letters occur earlier than all lowercase letters. Thus we get the relational ordering of:
   `'A' < 'B' < 'C' < ... < 'Z' < 'a' < 'b' < 'c' < ... <'z'`
   c) Representing numbers as characters, such as when you type on a keyboard, keeps the same ordering, so that `'0' < '1' < '2' < ... < '9'`.

Decisions in Java –Boolean Values & Expressions

The following program demonstrates the creation and output of boolean variables.

```
class BooleanOutput
{
      public static void main (String [] args)
      {
            boolean x = false;
            boolean y = true;
            System.out.print("x: ");
            System.out.println(x);
            System.out.print("y: ");
            System.out.println(y);
      }
}
```

Boolean Operators

It is possible to combine two or more *boolean values, variables, or expressions* into a more complicated boolean expression using the *boolean operators*. Unlike the *relational operators*, the boolean operators can only work on boolean values. You can use a relational operator to compare any data type, which forms a boolean expression. Multiple boolean expressions can be combined using boolean operators.

The boolean operators are summarized in the following table.

boolean value		**not p**	**p AND q**	**p OR q**
p	**q**	**!p**	**p && q**	**p || q**
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE

1. `!p` (not p) has the true/false value opposite to p.
2. `p && q` (p AND q) is true if and only if both p and q are both true.
3. `p || q` (p OR q) is true if p is true, q is true, or both p and q are true.

Decisions in Java –Boolean Values & Expressions

Exercises

1. State, with reasons, what this program will print (figure out your answer before trying to run the program!)

```
class BooleanVariables
{
     public static void main (String [] args)
     {
          boolean perhaps, maybe;
          perhaps = 4 < 5;
          maybe = (-17 % 4) == 1;
          System.out.println("perhaps: " + perhaps);
          System.out.println("maybe: " + maybe);
     }
}
```

2. Evaluate each expression, assuming the following declarations have been made.

```
boolean p = true, q = false, r = false, s = true;

a) !p                              f) s && !q
b) p || q                          g) p || !s
c) p && r                          h) !p && !q
d) !(q && s)                       i) s || (!q && r)
e) !q && s                         j) p == (q || r)
```

3. Rewrite the following program (from a previous exercise) using boolean expressions to combine the multiple IF statements into a single statement.

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

4. Using boolean expressions, 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. See the solution from the previous exercise (different note) if you need help with getting started.

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'. Use boolean logic to have your program handle uppercase at the same time ('Y' or 'N'). Also have your program handle an invalid response (anything other than 'y' or 'n').