Solution

Fundamentals - Operators & Operator Precedence

Fill in your answers in the space provided.

1. The Mod Operator: %

1. What are the values of these expressions:
   a) 10 % 0 = Divide by zero error
   b) 10 % 2 = 0
   c) 10 % 3 = 1
   d) 10 % 6 = 4
   e) 10 % 20 = 10
   f) 10 % 25 = 10

2. Given an integer x, if (x % 2) equals 0, what can we say about x?
   x is even

3. Given an integer x, if (x % 2) equals 1, what can we say about x?
   x is odd

4. Given an integer x, can (x % 2) equal anything other than 0 or 1?
   Yes, it can equal -1. (E.g. -3 % 2 = -1)

5. Given integers x and y, if (x % y) equals 0, what can we say about the relationship between x and y?
   x is a multiple of y.

6. Given integers x and y, if x equals y, what is the value of (x % y)?
   0

7. Given integers x and y, if x is less than y, what is the value of (x % y)?
   x

8. What are two different real-world applications where a mod operator would be useful?
   1. A clock program that has to cycle around from 1 to 12 and then to 1 again.
2. Operator Precedence

Programming language developers strive to make the languages they create unambiguous so that no matter what machine a program is run on it will run the same way. Although this is hard to accomplish in every scenario, one area in which this can be accomplished is in evaluating expressions. The rules for evaluating expressions look a first glance like what you saw in high school algebra (multiplication and division have a higher precedence than addition and subtraction, and they all associate from left to right). Since most programming languages have many operators, the rules for the order of evaluation are summarized in an "operator precedence table".

Use as a reference the Operator Precedence Table on Java Resources link page. NOTE: Make sure you understand how to read the operator precedence table - higher to lower precedence and associativity by reading the Rules. Note: Programmers are encouraged to use parentheses for clarity.

For each expression listed below, add parentheses to show how Java will evaluate it.

```java
// #1
2 + 5 * 10    // Answer, since * has higher precedence than +: 2 + (5 * 10)

// #2
3 - 2 + 1    //Answer: (3 - 2) + 1

// #3
3 - 2 + 5 * 10 - 9    // Answer: ((3 - 2) + (5 * 10)) - 9

// #4
a < b && b < c || c < d // Answer: ((a < b) && (b < c)) || (c < d)

// #5
w = x = y = z;    // Answer: w = (x = (y = z));
```

3. Truth Table

Fill in the following "truth table", where T means true and F means false:

| a | b | a && b | a || b | !(a && b) | (!a) && b |
|---|---|--------|--------|----------|-----------|
| T | T | T      | T      | F        | F         |
| T | F | F      | T      | T        | F         |
| F | T | F      | T      | T        | T         |
| F | F | F      | F      | T        | F         |