Exercise 7 Answers

Conditional constructs

1.

2. The problem is that `warnIfNegative` always prints out a warning message, even when the value is positive. The reason for the problem is the stray semicolon after the conditional expression of the `if` statement. Java strangely considers this semicolon to be a statement in itself, therefore closing off the `if` statement. The next line is then executed unconditionally.

3. The solution is to remove this stray semicolon.

4. The problem is that `resetIfNegative` always resets the value of `anIntProp`, even if `anIntProp` is positive. The problem is that the `if` statement doesn't have any braces around the statements inside it. In the absence of braces, Java considers the first statement: `System.out.println(...)` to be the closing part of the `if` statement. The next statement `anIntProp = 0` is then executed unconditionally.

5. The solution is to put braces around the two statements mentioned above, so that Java knows these statements are supposed to be part of the `if` statement.

6. ```
   public boolean isInRange(int value, int upperBound, int lowerBound) {
       if (lowerBound <= value && value <= upperBound)
           return true;
       else
           return false;
   }
   ```

7. Add some statements like this to `callSomeMethods`:

   ```
   System.out.println(isInRange(20,10,30));
   System.out.println(isInRange(40,10,30));
   ```

   The first statement prints out `true`, the second prints out `false`.

8. ```
   public boolean isInRangeIfLess(int value, int upperBound, int lowerBound) {
       return (lowerBound <= value && value <= upperBound);
   }
   ```

9. ```
   public boolean isInRange(int value, int upperBound, int lowerBound) {
       if (upperBound <= lowerBound) {
           System.out.println("Warning: upper bound less than or equal to lower bound");
       }
       if (lowerBound <= value && value <= upperBound)
           return true;
       else
           return false;
   }
   ```
   ```