Lesson 10 Assignment

Part 1: Identifying Types of Data Input

For this lesson's assignment, I'd like you to use the API and some of our formatting options. Look at the API documentation for the Scanner class, and identify all the different hasNext methods, like hasNextInt(), hasNextFloat(), hasNextBoolean(), and so on.

Use the hasNext methods to identify as many types of data input from the keyboard as possible. For example, if the user enters 0, that would be a byte type. If the user enters 333, that's too large for a byte, so it would be a short type. If the user enters 1.5 or 2.0e25 (that's exponential form for 2.0×10^{25}), then the type would be float.

See if you can write a program that will identify bytes, shorts, ints, longs, floats, and booleans. If the input isn't a type you can identify before reading it, then read it as a string.

Once you've identified it and read it, use the printf() method to display the type and the value. Format the output so that the types are left-justified (aligned with the left margin) in one column and the values are in another column, with numbers right-justified and Strings and booleans left-justified.

Part 2: Capture Huge Integers

Want a slightly bigger challenge? Try to capture integers that are too large for the long type in BigInteger objects. You can read about that class in the API. It allows you to store *very* big integers, even ones that have hundreds of digits. But other than that, using it is very similar to using the int or long types. There are even hasNextBigInteger() and nextBigInteger() methods in the Scanner class, and the printf() method treats them just like ints and longs.

Part 3: Distinguishing floats and doubles

If you're really looking for a challenge, try to distinguish between floats and doubles. This is difficult because the hasNextFloat() method interprets all floating-point numbers as floats, and hasNextDouble() reads them all as doubles. If the number's too big, both methods set its value to "infinity".

However, you can divide floating-point numbers into floats, doubles, and larger numbers using a different class named BigDecimal. You can do another API search to see how it works. If you decide to use it, here's the steps you'll need to take.

1. Read all floating-point numbers as BigDecimal objects. Fortunately Scanner contains hasNextBigDecimal() and nextBigDecimal() methods too.

2. Create four BigDecimal objects containing the minimum and maximum values for floats and doubles. Those values are in the constants Float.MIN_VALUE, Float.MAX_VALUE, Double.MIN_VALUE, and Double.MAX_VALUE. You can use those names as arguments for the BigDecimal constructor.

3. Compare the input value to those four values. If the input is between Float.MIN_VALUE and Float.MAX_VALUE, then the input is a float type. If it's not between those values, but it's between Double.MIN_VALUE and Double.MAX-VALUE, then it's a double. And if it's not between those values, it has to be a BigDecimal.

If you have questions, please let me know in the Discussion Area. If you'd like to look at my solution for all three parts, it's behind this link.

See my answer