Mr G's Java Jive

#5: More Variables

With this handout you'll learn how to use other types of variables.

What We Know So Far

We know how to create a program that asks for the user's name (or any other piece of **text**), but we're not able to handle numbers yet. It's time to add that capability to our **Converse** program.

Data Types

Previously, we've used two **data types** and both of them were **objects**. The **Scanner** type is an object that looks for input and the **String** type is an object that holds a string of text. We actually started out with some of the fancier types because they were easier to deal with at first and you could see results faster. Now it's time to look at some of the more basic types. They're called **primitives**.

Primitive Types

There are actually quite a few primitive types, but for our purposes right now, we only need to know about four: **int**, **double**, **char**, and **boolean**.

An **int** variable holds an **integer** value. That's any positive or negative **whole** number. That was simple enough.

A **double** variable holds a **decimal** value. Why is it called **double** instead of **dec**? Because it holds a decimal value that's twice as precise than its cousin **float** (for **floating point** number).

A **char** variable holds a **single character**. Like the letter **Y**, or the digit **8**, or the punctuation mark **<**. **19** is not a char value because it's two characters. It would have to be either an **int** or a **String** (think about that last one for a minute).

Finally, a **boolean** variable holds a value that's either **true** or **false**. That's all it can hold. You'll see how useful that will be later on.

By the way, pay careful attention to the fact that the names of the **primitive** types are all **lowercase**. This is very important.

Creating New Variables

We're going to modify the **Converse** program by having it ask the user what year they were born in, what year it is now, and then have it figure out roughly how old they are. We'll need three new variables for this: **born**, **now**, and **age**. Take a look at the example below to see how to add them to **Converse**.

```
public static void main()
   {//start main
      //variables
        Scanner myScanner = new Scanner(System.in);
        String name;
        int born, now, age;

      //get input
```

You're just **declared** three new variables, and since they were all **int** variables, you were able to do them all on the same line, separated by **commas**. As usual, since this is a **statement**, the whole line ends with a **semicolon**.

Getting More Input

Now that we have the variables we need to hold the information, we need to get it from the user. The example below shows the lines that we'll add to the **input** section.

```
//get input
  System.out.print("Hi! What's your name? ");
  name=myScanner.nextLine();
  System.out.print("Nice to meet you "+name);
  System.out.print(". What year were you born in? ");
  born = myScanner.nextInt();
  System.out.print("Cool. And what year is it now? ");
  now = myScanner.nextInt();

//give response
```

One of the first things you should notice here is that I used two **print** statements when I really could've just used one. I did this because the code for doing it with just one **print** statement would be way too long for the editing window.

More About the Scanner

Up until now, you only used the **Scanner** object to scan for **text** input from the keyboard by using **myScanner.nextInt()**. But the **Scanner** class has quite a few methods for gathering information. Here's a list of a few of them.

Scanner Method	What it Gathers
nextLine()	an entire line of text, spaces included
next();	one word of text, ending at the first space
nextInt()	an integer
nextDouble()	a real number

You may have noticed that I didn't mention **nextChar()**. That's because there isn't one. The **Scanner** class has an odd little way of dealing with getting a single character that's a bit inconstant with the way the rest of the commands work. I'll give you a better, and more consistent, way to do this later on. Right now it's time to figure out how old our user is.

A New Section for Calculations

So far we have a section for **variables**, a section for **input**, and a section for **output**. Well since we're going to be doing some calculations now, we'll need one for them. Check out the example below to see what code you need to add to your program.

```
  now = myScanner.nextInt();

//calculations
  age = now-born;

//give response
```

OK, this all seems very basic to me, but there are still people I need to explain this to over and over again: **You can't do any calculations until you have values for your variables.** What does this mean? It means that if you try to do calculations before you get the values from the user, the program either won't work at all or it will give you garbage results. Make sense? Good. Let's move on.

Displaying Our Results

Now it's time to say how old we think the user is. Check out the **bold** additions to the **give response** section in the example below.

```
  //give response
    System.out.println("Nice to meet you, "+name+".");
    System.out.println("You'll be "+age+" this year.");

}//end main
```

Remember that fancy word **concatenation** that we used a few handouts ago? Well, that's what just happened here when we added the value of our **age** variable to the string of text we sent to the screen using our **println** statement.

Did you notice that we said what age the user **will be** and not what age they **are**? That's because depending on when their birthday is within the year, we could be wrong about their actual age. Saying hold old the user **will be** this year takes care of that.

Now run the program and see what happens.

Oops!

And that doesn't stand for **object-oriented programming** either. It mean we made a mistake. If you look at the output when you run this program, you'll see that we say that it's nice to meet the user **twice**. We originally did it in the **give response** section. Then we we asked for the user's birthdate, we did it again. We only need one of these, so strike (that is, **remove**) the old one from the **give response** section. If you're still confused about what I mean, check out the example below and remove that line that has the **strikethrough**.

```
//give response
     ~~System.out.println("Nice to meet you, "+name+".");~~
     System.out.println("You'll be "+age+" this year.");

}//end main
```

Now things should work just fine!

What's Up Next?

Now that you've had some success with this, we'll take a look at what happens when users are stupid, in handout #6.

This page intentionally left **almost** blank.