Introduction to Numeric Data Types

by Sophia

WHAT'S COVERED

In this lesson, you will learn about values, variables, and the basic data types used in Java. Specifically, this lesson covers:

1. Numeric Values and Variables
2. Boolean Data Type

1. Numeric Values and Variables

So far, we've discussed examples of variables and values. However, we haven't explored what variables and values are. A **value** is one of the basic things a program works with. It can be as simple as a letter or a number. You have used them in a few of the previous examples. In these examples, you hardcoded certain values into the program. You used values like the number 2, which is also an integer. You also used the letters "Hello World," which is actually a "string" of letters. Strings can be identified within the enclosed double quotation marks.

A variable is a human-readable tag that identifies a place in memory where a particular kind of data is stored. A variable in Java consists of an identifier. An **identifier** is a name for the variable. It is also a **type**, or kind of data that the variable can store. These variables are required so that programs can retain and update data as the program is executed. As a programmer, you will need to consider the various **data types** of variables as you work with them.

KEY CONCEPT

When providing an identifier, like 'a name', as a variable in Java, keep the following rules in mind:

- An identifier is a continuous sequence of characters without spaces. There can never be a space in a variable name.
- An identifier can consist of upper- and lowercase letters, digits, or underscores '_'.
- The first character must be a letter or an underscore.
- Identifiers are always case sensitive, so amount, `Amount`, and `AMOUNT` are different identifiers in Java.

It is important to give variables meaningful, brief, and descriptive names. Using a letter or two, such as `s` or `st`, is not nearly as meaningful as naming a variable `subtotal` or `salesTax`. Depending on the situation, it might even be better to use variable names like `stateTaxRate`, `stateTaxAmt`, `countyTaxRate`, and `countyTaxAmt`.

Appropriate variable names make the programmer's intentions clear. It also helps others who have a need to read and to understand the code. Variable names should be 4 - 15 letters in length, although it could take up to 20 letters to avoid confusion at times. Capital letters are used to help make multi-word identifiers more readable. For the time being in this tutorial, assume that a variable name normally begins with a lowercase letter. (This style of capitalization of subsequent words in a variable name is called "camel casing," but there will be more about that later.)

HINT

Several types of variables will be covered in this and the following tutorials. However, to keep things simple for now, you should consider types that are familiar to you. This should include:

- `int` is an integer (whole number) value, such 0, -1, 123456, etc.
- `String` (note the capital S) is for strings of characters: `"Hello world!"`, `"Please enter your name: "`, etc.

To use a variable in a program, users must declare the variable by indicating in the code the type and name of the variable:

```
int itemCount;
String customerName;
```

These declarations reserve space in computer memory for the type of data indicated, that is accessible using the identifier, by the variable name. It is important to consider that `itemCount` and `customerName` do not have values in these declarations. They will need to be given values before they can be used. Giving them values is accomplished by using a single equal sign (=), also known as the assignment **operator**.

A short and simple program using the variables shown above might look like the following:

```
public class OrderManagement {
  public static void main(String[] args) {
    // Declare variables by providing type & identifier
    int itemCount;
    String customerName;
    // Assign appropriate values using =
    itemCount = 3;
    customerName = "Jones, Anne";
    // Display values - Don't worry about the details just yet
```

```
    System.out.println("The customer's name is " + customerName + ".");
    System.out.println("The order contains " + itemCount +
      " item(s).");
  }
}
```

This code includes a couple of small features of the language that haven't been covered yet, but it should still make sense. For instance, the `+` operator is used to join text and the values of variables.

TRY IT

Directions: Try typing the code above into a file named OrderManagement.java in the IDE.

Run the program,

It should produce results like this:

```
The customer's name is Jones, Anne.
The order contains 3 item(s).
```

REFLECT

This code declares the variables and assigns them values in 2 separate steps. The declaration and value assignment can be done in the same line of code. Providing an initial value for a variable when it is declared is called **initialization**.

Here is a version of the code using this approach:

```
public class OrderManagement {
  public static void main(String[] args) {
    // Declare variables & initialize them
    int itemCount = 3;
    String customerName = "Jones, Anne";
    // Display values - Don't worry about the details just yet
    System.out.println("The customer's name is " + customerName + ".");
    System.out.println("The order contains " + itemCount + " item(s).");
  }
}
```

REFLECT

You are probably already very familiar with the concept of deciding that you wanted a type of something, that does something you like, and then naming it. Perhaps you wanted pets. If you decided to get a dog (its type) and named him Spike (its identifier), and a cat (its type) and named him Sylvester (its identifier) then you have gone through the process of selecting variables for your pets – picking a type and naming each one. What

about values for what they do? Well, Spike might “play” for a while, then “bark” and Sylvester might “purr” and then “climb a tree.”

As you’re deciding on variables for your programs make sure you pick the right type for what you want it to do. You certainly wouldn’t want to pick the cat Sylvester if you need something to “bark” for protection. Similarly, if you need to keep track of how many bags of pet food you have, use an integer not a string. An integer is the best way to represent the whole number of bags that you have

Running this code in the IDE should produce the same results as the first version above. Determining whether a variable should be initialized, when it is declared, is a design decision that will be covered later. For now, be aware of these 2 approaches.

This tutorial focuses on numeric data types, so for the rest of this tutorial you will work with numeric data types. You will return to strings in the next tutorial.

When working with numeric data, it is important to distinguish between values that are whole numbers with no decimal (known as **integers**) and values that include—or may include—a decimal, known as **floating-point** values.

> **IN CONTEXT**
> Just as model airplanes are representations of real airplanes, Java’s numeric types are representations or models. They are representations or models of the data types and are abstract numbers that are dealt with in mathematics. When designing Java’s data types, various trade-offs have been made in order to come up with practical implementations.

Internally, the computer represents all values as series of 0's and 1's of different lengths. This way of representing values is called "binary" because each digit can be one of two things, either a 0 or a 1. Each of these binary digits is called a "bit." Java distinguishes between different data types for variables because different kinds of values take up different amounts of space in memory, and the way the data is represented in that space varies.

The amount of space, expressed as the number of bits used to store the data, determines the range of values that can be stored. So that the computer can keep track of where one value ends and the next begins, the different data types have specific lengths in bits. The number of bits used to store a given type of data is specified in the Java Standard, which provides Java's representation of values consistently across different platforms.

When dealing with types of integers, one of the bits is used to keep track of the number's sign, whether it is positive or negative. This means that for the byte data type, there are 7 bits available to represent the number. Since each bit can be in one of two states (0 or 1), the highest positive value that a byte can represent is 2 x 2 x 2 x 2 x 2 x 2 x 2. This is also considered the seventh power of 2, which is commonly written as 2^7.

For integer values, Java has the following types:

Type	Bits	Range of Values
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2,147,483,648 to 2,147,483,647
long	64	-2^{63} to $2^{63}-1$

While it's important to know that the `byte` and `short` types exist, `int` and `long` are the types of integers you are most likely to need for most programs.

For floating-point values, Java has 2 data types:

Type	Bits	Range of Values
float	32	-3.40292347E+38 to +3.40292347E+38
double	64	-1.79769313486231570E+308 to +1.79769313486231570E+308

The scientific notation used to specify the ranges of values that these types can store can be a bit hard to interpret, if you haven't worked with it previously. Scientific notation is used to express very large and very small values that have too many digits to be expressed concisely on the screen or page. The value to the left of the E is multiplied by 10 raised to the power to the right of the E. The key takeaway is that the `double` doesn't just store a wider range of values, but it can represent value with much greater accuracy than `float`. A `float` can store values with up to 8 significant digits (a significant digit is a digit on either side of the decimal point that contributes to the number's value). A `double`, on the other hand, can handle values with up to 17 significant digits.

If a value is assigned to a floating-point variable that contains more significant digits that the type can handle, the value will be rounded off. This means that if 12345.6789, a floating-point number with 9 significant digits, is assigned to a `float` variable, it will be rounded to 12345.679. This represents the closest value with 8 significant digits. Trying to assign the value 0.123456789 will result in rounding to 0.12345679, since the original value has 9 significant digits and needs to be rounded to 8. The 0 to the left of the decimal is not significant because it doesn't play a role in the actual value.

When entering numbers—not variables, but the actual numbers with decimals—Java assumes that any value with a decimal point is a `double`. If you want to indicate that a specific number should be treated as a float type, you need to put the lowercase letter at the end of the number without a space:

```
float width = 0.123456789f;
```

TRY IT

Directions: Here is a brief program to type into a file called FloatingPoint.java on the IDE. It demonstrates the difference in precision between the `float` and `double` data types in Java.

```
public class FloatingPoint {
  public static void main(String[] args) {
    float sampleFloatValue = 0.123456789f;
    double sampleDoubleValue = 0.123456789;
    System.out.println("Value as float:  " + sampleFloatValue);
    System.out.println("Value as double: " + sampleDoubleValue);
  }
}
```

Run the program.

The should look like this:

```
Value as float:  0.12345679
Value as double: 0.123456789
```

In addition to variables, which hold values that can change as the program runs, Java also allows for the storage of named values that are unchanging or constant. Like variables, constants have a data type and identifier (name). The keyword `final` is placed before the data type to indicate the declaration of a constant.

Storage of named values that are constant using the `final` keyword:

```
final int dozen = 12;
final double pi = 3.1415926;
```

KEY CONCEPT

A constant must be assigned a value when it is declared. Trying to assign a new value to a constant results in an error.

TERMS TO KNOW

Value
One of the basic units of data, like a number or string, that a program manipulates.

Identifier
An identifier is a name for the variable.

Data Type (or Type)
The type of data that a variable can store.

Initialization
The providing of an initial value of a variable when it is declared.

= operator
The = operator (equal sign) is an assignment operator that is used to assign values to variables.

int (integer)
A numeric data type consisting of an integer or a whole number. It can either be a positive or a negative number, but it does not have any decimals.

float
A numeric data type referring to a floating-point number. It is a number that can be positive or negative and uses decimal values.

---

2. Boolean Data Type

The **boolean** data type is a much more limited data type than the others. The `boolean` data type can only have one of two values: `true` or `false`. Both are all spelled with lowercase letters. Despite the limited range of values, `boolean` values are important in computer programming. It was important to mention them here in case you come across them or have a need for them. However, we will discuss them in greater detail in a future tutorial.

TERM TO KNOW

boolean
A boolean data type consisting of a value of either true or false.

SUMMARY

In this lesson, you learned about **numeric and boolean values and variables**. This included exploring how a value is one of the basic things a program works with, like a letter or number. You learned that variables hold values and can change values within a program. You also learned that variables need to be named appropriately based on some "rules" and that their names should be meaningful. You demonstrated the basics of writing variables in different case structures. Finally, you learned about Java's basic data types including `int`, `float`, `double`, `boolean`, and `string`.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf

It has also been adapted from "Python for Everybody" By Dr. Charles R. Severance. Source py4e.com/html3/

TERMS TO KNOW

= operator

The = operator (equal sign) is an assignment operator that is used to assign values to variables.

Data Type (or Type)

The type of data that a variable can store.

Identifier

An identifier is a name for the variable.

Initialization

The providing of an initial value of a variable when it is declared.

Value

One of the basic units of data, like a number or string, that a program manipulates.

boolean

A boolean data type consisting of a value of either true or false.

float

A numeric data type referring to a floating-point number. It is a number that can be positive or negative and uses decimal values.

int (integer)

A numeric data type consisting of an integer or a whole number. It can either be a positive or a negative number, but it does not have any decimals.