Overview
Hi class, this podcast will cover some of the basic programming concepts such as identifiers, variables, named constants, and data types. This podcast is not a substitute for reading your assigned resources.

Identifiers
Identifiers are descriptive names that are mapped to locations in the computer’s memory. Using a name for your memory location is easier than referring to the location via its numeric address. Identifiers are for naming variables, constants, methods, classes, and packages. In other words, the term identifier is a generic term. In this podcast, we will discuss some specific identifiers.

Computer languages have rules to follow when creating identifier names. Identifiers cannot contain spaces, but can contain letters, digits, underscores, and dollar signs. Although, according to convention, the dollar sign is used in machine generated code and therefore you should avoid using it in your identifier names. Identifiers cannot start with a digit, nor can they be the words true, false, or null. Many languages have a set of reserved words. These words are used by the language when it’s compiling and running a program so you should not use a reserved word as an identifier.

Although an identifier can just about be any length, do not make it overly long or too short. The name you give your identifier should be meaningful and convey what you are trying to do. Please get familiar with the WITC Software Development Standards document. There is a section on naming conventions which you are expected to follow.

Variables
Variables represent a place in memory that is capable of holding data. The easiest way to refer to the memory storage location is by a name. Think of a variable like an envelope. Your envelope can hold data, but the data must fit in the space. Also, it’s important to remember that the value of your variable can be changed while the program is running.

The name you give your variable should reflect what you are trying to do in your program. For instance, if you are trying to capture the number of songs someone wants to purchase, you could name the variable songs. However, that does not fully convey what you are trying to do. Naming the variable numberOfSongs would be more accurate. According to our WITC Software Development Standards, variables should be written using camelCase.

Named Constants
Named constants are similar to variables in that they represent a place in memory that is capable of holding data. However, unlike variables, the value of a named constant cannot be changed while the program is running. The name you give your named constant, or constant for short, should be
meaningful and convey what you are trying to do. Our WITC Software Development Standards
document states that constants should be in all capital letters. However, something like tax rate would
be hard to read if all in caps, therefore, we use an underscore between the two words TAX_RATE.

Data Types
Remember how I said that a variable is like an envelope and that your data had to fit into the space.
Well, the way you make your data fit is by creating identifiers of a certain type. Data types fall into two
basic categories, numeric and alphanumeric. Another data type we will use is Boolean.

Numeric Data Types
Each programming language has a range for their numeric data types. This range may differ from
language to language. Fortunately, most have a mechanism where we can compare a number to the
minimum or maximum in the range without having to know the exact value. A numeric literal is a
number you write in your program. So if you write the number 12, then 12 can be literally printed to the
screen. Writing the number instead of using a variable means the number is hard-coded and cannot be
changed while the program is running. It’s better to save the numeric value into a variable so you can
change it while the program is running. Remember if your number, such as tax rate, doesn’t need to be
changed, define it as a named constant with the appropriate numeric data type.

Something to remember is that not all numbers need to be represented with a numeric data type. My
general rule of thumb is, if you will be doing math with the variable, then make it the appropriate
numeric data type. However, numbers like your student id or zip code do not generally have math done
on them, so they can be represented as an alphanumeric data type.

Float or Double
Although technically float and double or not the same thing, they both will hold a number that contains
a decimal point. There are other data types which can hold decimal values, but for now, we’ll stick with
these two. Remember, I said, the data type you choose depends on what you will be doing. Well, if we
want to calculate how much the songs will cost, we would need a double or a float. In that case, you
would multiply the cost of the songs by the number of songs purchased. The resulting number is saved
into a variable of type float or double.

Integer
An integer is a numeric data type that holds whole numbers. Logically speaking, our variable
numberOfSongs would be of data type integer because people do not purchase part of a song. If you try
to save a decimal value into an integer variable, most programming languages will truncate the value. In
other words, it will disregard everything after the decimal point.

Alphanumeric Data Types
An alphanumeric data type is basically anything that can be typed from the keyboard, including letters,
numbers, and symbols. The alphanumeric data type falls into two basic categories, character and string.
When we refer to an alphanumeric literal, we mean one or more characters enclosed in quotes. Think of
it this way, whatever is in quotes can be literally printed to the screen. Using a sequence of quoted
characters instead of using a variable means the characters are hard-coded and cannot be changed
while the program is running. It’s better to save the quoted characters into a variable so you can change
them while the program is running. Remember if your sequence of characters does not need to be
changed, define them as a named constant with the appropriate alphanumeric data type.
**Character**

One printable letter, number, or symbol is considered a character. When you define a variable as data type character, you are making a contract stating you will only put one letter, number, or symbol in that variable. If you have a character literal, then it is typically enclosed in single quotes. An example of a literal character would be the letter ‘A’. An empty character would be the single quote marks with no spaces between them.

**String**

One or more printable letters, numbers, or symbols are considered a string. String literals are enclosed in a pair of double quotes. An example of a string literal would be my name, “Rene”. An empty string would be a pair of double quote marks with no spaces between them.

**Booleans**

The final data type we will cover is Booleans. A Boolean is also known as a logical data type. A Boolean data type has only two possibilities; either true or false. Not all languages have a built-in Boolean data type. So another way to represent Boolean values is with the number one and a zero. True is the number one and false is the zero. Fortunately, Java and C# use true and false.

When you use relational operators, such as less than, greater than, and the equality operator, behind the scene, the question is being asked, “Is this true”. Is it true that five is less than ten? Is it true that ten is greater than five? Is it true that five is equal to three? Or, is it true that the value of one variable is equal to the value in another variable? Remember though, when comparing strings in Java, you must use dot equals (.equals( )).

Remember, you are in control and order matters. The program is only going to do what you tell it to do so make a plan before you start because if you can say it, you can code it.

Thanks for listening and happy computing!