

Variables, Constants, and Data Types

- Primitive Data Types
- Variables, Initialization, and Assignment
- Constants
- Characters
- Strings
- Reading for this class: L&L, 2.1-2.3, App C

Primitive Data

- There are eight primitive data types in Java
- Four of them represent integers:
 - `byte`, `short`, `int`, `long`
- Two of them represent floating point numbers:
 - `float`, `double`
- One of them represents characters:
 - `char`
- And one of them represents boolean values:
 - `boolean`

Numeric Primitive Data

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

<u>Type</u>	<u>Storage</u>	<u>Min Value</u>	<u>Max Value</u>
byte	8 bits	-128	127
short	16 bits	-32,768	32,767
int	32 bits	-2,147,483,648	2,147,483,647
long	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
float	32 bits	+/- 3.4×10^{38} with 7 significant digits	
double	64 bits	+/- 1.7×10^{308} with 15 significant digits	

Boolean Primitive Data

- A `boolean` value represents a true or false condition
- The reserved words `true` and `false` are the only valid values for a `boolean` type

```
boolean done = false;
```


- A `boolean` variable can represent any two states such as a light bulb being on or off

```
boolean isOn = true;
```

Variables

- A *variable* is a name for a location in memory
- A variable must be *declared* by specifying the variable's name and the type of information that it will hold

data type variable name



```
int total;
```

- Multiple variables can be created in one declaration:

```
int count, temp, result;
```

Variable Initialization

- A variable can be given an initial value in the declaration with an equals sign

```
int sum = 0;  
int base = 32, max = 149;
```

- When a variable is referenced in a program, its current value is used
- See [PianoKeys.java](#) (page 66-67)

```
int keys = 88;  
System.out.println("A piano has " + keys + " keys.");
```

- Prints as:
A piano has 88 keys.

Assignment

- An *assignment statement* changes the value of a variable
- The equals sign is also the assignment operator

```
total = 55;
```



- The expression on the right is evaluated and the result is stored as the value of the variable on the left
- The value previously stored in `total` is overwritten
- You can only assign a value to a variable that is consistent with the variable's declared type
- See [Geometry.java](#) (page 68)

Constants

- A constant is an identifier that is similar to a variable except that it holds the same value during its entire existence
- As the name implies, it is constant, not variable
- In Java, we use the reserved word `final` in the declaration of a constant

```
final int MIN_HEIGHT = 69;
```

- Any subsequent assignment statement with `MIN_HEIGHT` on the left of the `=` operator will be flagged as an error

Constants

- Constants are useful for three important reasons
- First, they give meaning to otherwise unclear literal values
 - For example, NUM_STATES means more than the literal 50
- Second, they facilitate program maintenance
 - If a constant is used in multiple places and you need to change its value later, its value needs to be updated in only one place
- Third, they formally show that a value should not change, avoiding inadvertent errors by other programmers

Characters

- A `char` variable stores a single character
- Character literals are delimited by single quotes:

`'a'` `'X'` `'7'` `'$'` `','` `'\n'`

- Example declarations:

```
char topGrade = 'A';
```

```
char terminator = ';', separator = ' ';
```

Character Sets

- A *character set* is an ordered list of characters, with each character corresponding to a unique number
- A `char` variable in Java can store any character from the *Unicode character set*
- The Unicode character set uses sixteen bits per character, allowing for 65,536 unique characters
- It is an international character set, containing symbols and characters from many world languages

Characters

- The *ASCII character set* is older and smaller than Unicode, but is still quite popular (in C programs)
- The ASCII characters are a subset of the Unicode character set, including:

uppercase letters	A, B, C, ...
lowercase letters	a, b, c, ...
punctuation	period, semi-colon, ...
digits	0, 1, 2, ...
special symbols	&, , \, ...
control characters	carriage return, tab, ...

Character Strings

- A string of characters can be represented as a *string literal* by putting double quotes around the text:

- Examples:

```
"This is a string literal."
```

```
"123 Main Street"
```

```
"X"
```

- Note the distinction between a primitive character `'X'`, which holds only one character, and a `String` object, which can hold a sequence of one or more characters
- Every character string is an object in Java, defined by the `String` class

The println Method

- In the `Lincoln` program from Chapter 1, we invoked the `println` method to print a character string
- The `System.out` object represents a destination (the monitor screen) to which we can send output

```
System.out.println ("Whatever you are, be a good one.");
```



The print Method

- The `System.out` object provides another method
- The `print` method is similar to the `println` method, except that it does not start the next line
- Therefore any parameter passed in a call to the `print` method will appear on the same line
- See [Countdown.java](#) (page 59)
 - `System.out.print ("Three... ");`
 - `System.out.print ("Two... ");`
- Prints as:
Three... Two...


String Concatenation

- The *string concatenation operator* (+) is used to append one string to the end of another

```
"Peanut butter " + "and jelly"
```

- It can also be used to append a number to a string
- A string literal cannot be broken across two lines in a program so we must use concatenation
- See [Facts.java](#) (page 61)

```
System.out.println("We present the following facts for your "  
+ "extracurricular edification");
```

NOTE: 
No ; here

String Concatenation

- The + operator is also used for arithmetic addition
- The function that it performs depends on the type of the information on which it operates
- If both operands are strings, or if one is a string and one is a number, it performs string concatenation
- If both operands are numeric, it adds them
- The + operator is evaluated left to right, but parentheses can be used to force the order
- See [Addition.java](#) (page 62)
 - `System.out.println("24 and 45 concatenated: " + 24 + 45);`
- Prints as:
 - 24 and 45 concatenated: 2445

String Concatenation

- The + operator is evaluated left to right, but parentheses can be used to force the order
- See [Addition.java](#) (page 62)
System.out.println("24 and 45 added: " + (24 + 45));
Addition is Done first
- Prints as:
24 and 45 added: 69
Then concatenation is done

Escape Sequences

- What if we want to include the quote character itself?
- The following line would confuse the compiler because it would interpret the two pairs of quotes as two strings and the text between the strings as a syntax error:

```
System.out.println ("I said "Hello" to you.");
```

A String Syntax Error A String

- An *escape sequence* is a series of characters that represents a special character
- Escape sequences begin with a backslash character (\)

```
System.out.println ("I said \"Hello\" to you.");
```

A String

Escape Sequences

- Some Java Escape Sequences

<u>Escape Sequence</u>	<u>Meaning</u>
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\"</code>	double quote
<code>'</code>	single quote
<code>\\</code>	backslash

- See [Roses.java](#) (page 64)

```
System.out.println("Roses are red,\n\tViolets are blue,\n" +
```

- Prints as:

```
Roses are red,  
    Violets are blue,
```

Escape Sequences

- To put a specified Unicode character into a string using its code value, use the escape sequence: `\uhhhh` where `hhhh` are the hexadecimal digits for the Unicode value
- Example: Create a string with a temperature value and the degree symbol:

```
double temp = 98.6;
System.out.println(
    "Body temperature is " + temp + " \u00b0F.");
```

- Prints as:

```
Body temperature is 98.6 °F.
```

Methods of the String class

- String is a class and classes can have methods.
- Use the Sun website link to find definitions of the methods for each standard library class
- The classes are listed in alphabetical order
- The String class has methods that can be used to find out the characteristics of a String object such as its length:

```
System.out.println("Hello".length());
```
- Prints the number 5 (for 5 characters in length)