Chapter 2:
Data and Expressions
Variable Declaration

- In Java when you declare a variable, you must also declare the type of information it will hold

```java
float myFloat; // Declaration only
int x = 10; // Declaration with initial value
char aLetter = 'q';
String myString = “Hello World”;
```
Variable Assignment

• Assignment is done using the = sign. Assignment changes the value of a variable. Your program will always use the current value.

```java
int x; // Declares an integer variable x
x = 10; // Makes the current value of x equal to 10
System.out.println("The value of X is "+x); // Prints 10
x = 200; // Makes the current value of x equal to 200
System.out.println("The value of X is "+x); // Prints?
```
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

Anything that is not a primitive is a class in Java!
# Numeric Primitive Data

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>&lt; -9 \times 10^{18}</td>
<td>&gt; 9 \times 10^{18}</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 \times 10^{38} with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 \times 10^{308} with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>
boolean

boolean has two values: true or false

boolean status = false; // declare and assign
status = true; // Reassign the value

boolean variables are useful in decision statements

NOTE: booleans use the reserved words true and false (true and false are NOT strings, they are reserved words like “if” and “while” in the Java language.)
char variables hold one character

char myChar = ‘a’; // Declare and assign

myChar = ‘Z’; // assign new value

char anotherChar = ‘x’; // Create a new variable
Expressions

- An expression is a combination of one or more operators and operands
- Arithmetic expressions compute numeric results and make use of the arithmetic operators:

  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: /
  - Remainder: %

If either or both operands used by an arithmetic operator are floating point, then the result is a floating point
Division and Remainder

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)

\[
\begin{align*}
14 \div 3 &= 4 \\
14.0 \div 3 &= 4.33333 \\
8 \div 12 &= 0
\end{align*}
\]

The remainder operator (%) returns the remainder after dividing the second operand into the first. (Also know as : modulo operator)

\[
\begin{align*}
14 \mod 3 &= 2 \\
8 \mod 12 &= 8
\end{align*}
\]
The `System.out` object represents a destination (the monitor screen) to which we can send output.

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

- **object**
- **method name**
- **information provided to the method (parameters)**
The print Method

- The `System.out` object provides another service as well

- The `print` method is similar to the `println` method, except that it does not advance to the next line

- Therefore anything printed after a `print` statement will appear on the same line
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.
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
- int x = 7, y=12;
- int total = x+y; // Does addition
- String myString = “Dan”;  
- String name;
- name = myString + “ Fleck”; // Does String concatenation
Interactive Programs

- Programs generally need input on which to operate
  - The *Scanner* class provides convenient methods for reading input values of various types
  - A *Scanner* object can be set up to read input from various sources, including the user typing values on the keyboard
  - Keyboard input is represented by the *System.in* object
The following line creates a Scanner object that reads from the keyboard:

Scanner scan = new Scanner (System.in);

The new operator creates the Scanner object.

Once created, the Scanner object can be used to invoke various input methods, such as:

answer = scan.nextLine();
Reading Input

- The *Scanner* class is part of the *java.util* class library, and must be imported into a program to be used.

- The *nextLine* method reads all of the input until the end of the line is found.

- The details of object creation and class libraries are discussed further in Chapter 3.
Input Tokens

- Unless specified otherwise, white space is used to separate the elements (called tokens) of the input.

- White space includes space characters, tabs, new line characters.

- The next method of the Scanner class reads the next input token and returns it as a string.

- Methods such as nextInt and nextDouble read data of particular types.
Math class in Java

- The Math class provides useful functions (sin, cos, sqrt, abs, etc…)

- To use them we call the appropriate methods:

```java
x = Math.sqrt(24);
y = Math.sin(1.28); // Note: Angles MUST be in radians, not degrees!
double radAngle = Math.toRadians(90);
```
Now we know

- Math
- String
- Input/Output through the console

- So, we can write a calculator!
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"

• Every character string is an object in Java, defined by the String class. This means you can call methods on a String!

• Every string literal represents a String object
Escape Sequences

- What if we wanted to print a the quote character?
- The following line would confuse the compiler because it would interpret the second quote as the end of the string
  
  ```java
  System.out.println ("I said "Hello" to you.");
  ```

- An escape sequence is a series of characters that represents a special character
- An escape sequence begins with a backslash character (\)
  
  ```java
  System.out.println ("I said \"Hello\" to you.");
  ```
## Escape Sequences

Some Java escape sequences:

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>backspace</td>
</tr>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>\n</td>
<td>newline</td>
</tr>
<tr>
<td>\r</td>
<td>carriage return</td>
</tr>
<tr>
<td>&quot;</td>
<td>double quote</td>
</tr>
<tr>
<td>'</td>
<td>single quote</td>
</tr>
<tr>
<td>\</td>
<td>backslash</td>
</tr>
</tbody>
</table>

See [Roses.java](https://example.com/Roses.java)
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.
- The compiler will issue an error if you try to change the value of a constant.
- In Java, we use the `final` modifier to declare a constant.

```java
final int MIN_HEIGHT = 69;
```
Why use Constants

• Constants are useful for three important reasons
  
  First, they give meaning to otherwise unclear literal values
  ◦ For example, MAX_LOAD means more than the literal 250

  Second, they facilitate program maintenance
  ◦ If a constant is used in multiple places, its value need only be updated in one place

  Third, they formally establish that a value should not change, avoiding inadvertent errors by other programmers