Variables and Data Types

Chapter 2

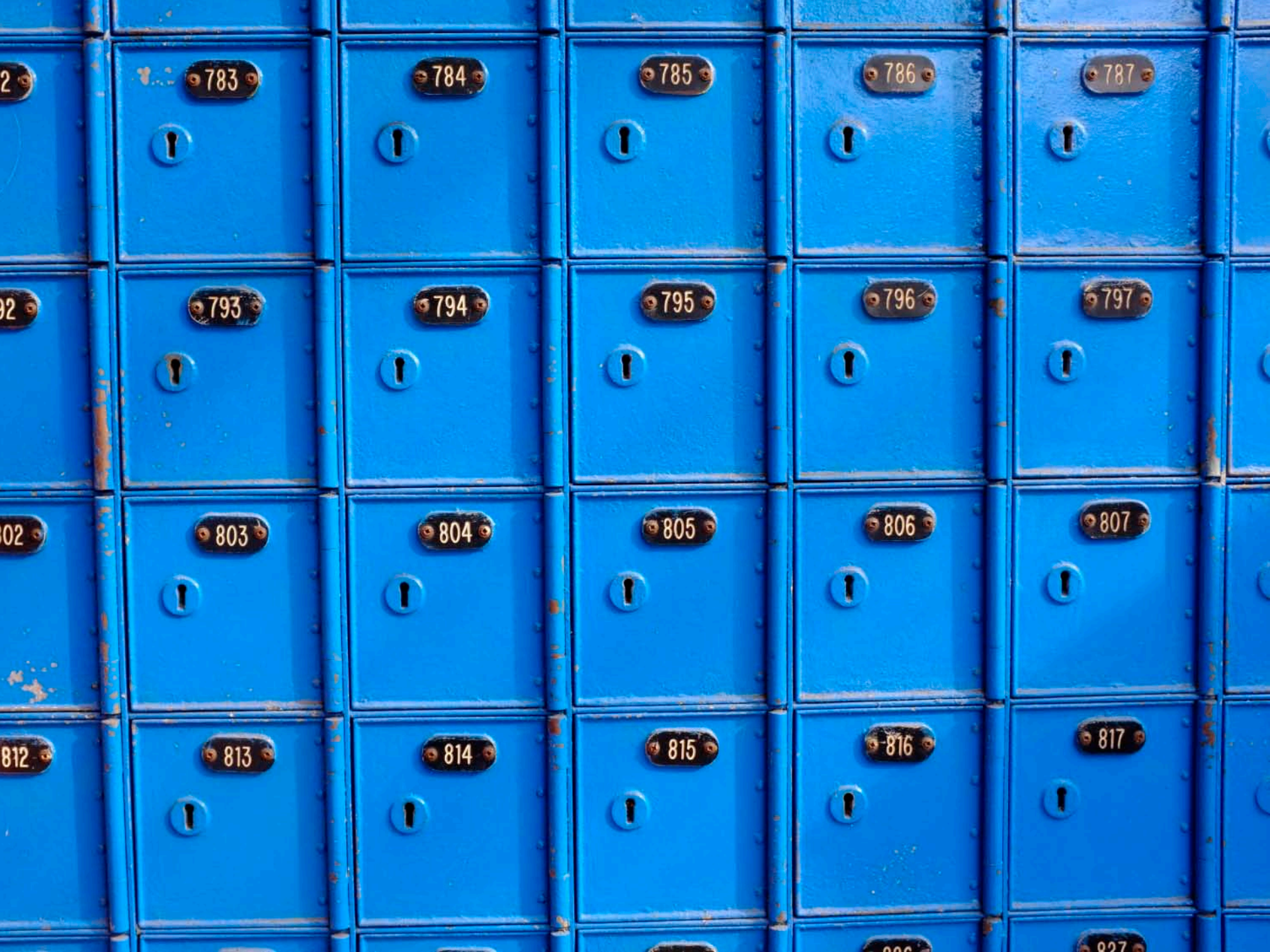
Chapter Goals

- To be able to declare and initialize variables and constants
- To write assignment statements
- To understand the properties and limitations of integers and floating-point numbers
- To create programs that read and process inputs, and display the results

Declaring Variables

Variables

- A variable is a name for a value stored in memory.
- Think of a variable as a mailbox...


783
784
785
786
787
793
794
795
796
797
803
804
805
806
807
813
814
815
816
817

Declaring Variables

- Most computer programs hold temporary values in named storage locations
  - Programmers name them for easy access
- There are many different types (sizes) of storage to hold different things
- You 'declare' a variable by telling the compiler:
  - What type (size) of variable you need
  - What name you will use to refer to it

Declaring Variables

- A variable must be declared before it can be used.
- Variables are created using a declaration statement.
- Declaration statement takes the form:

```
<data type><name>
```

- An example:

```
int length
```

- Variables can only store one value at a time

Declaration Statements

- A declaration statement must include the data type and the identifier

int length

Data Type

Identifier

Example Declarations

```
int cansPerPack = 6;
```

Types introduced in this chapter are the number types `int` and `double`

and the `String` type

A variable declaration ends with a semicolon.

Supplying an initial value is optional, but it is usually a good idea.

Use a descriptive variable name.

Example

Applications typically contain many variables, as in RectangleArea:

```
/**
* Calculates and displays the area of a rectangle
*/

public class RectangleArea {

    public static void main String[] args) {
        int length = 10;  onger side of rectangle
        int width = 2;  shorter side of rectangle
        int area;  calculated area of rectangle

        area = length * width;
        System.out.println("Area of rectangle: " + area);

    }
}
```

Example Declarations

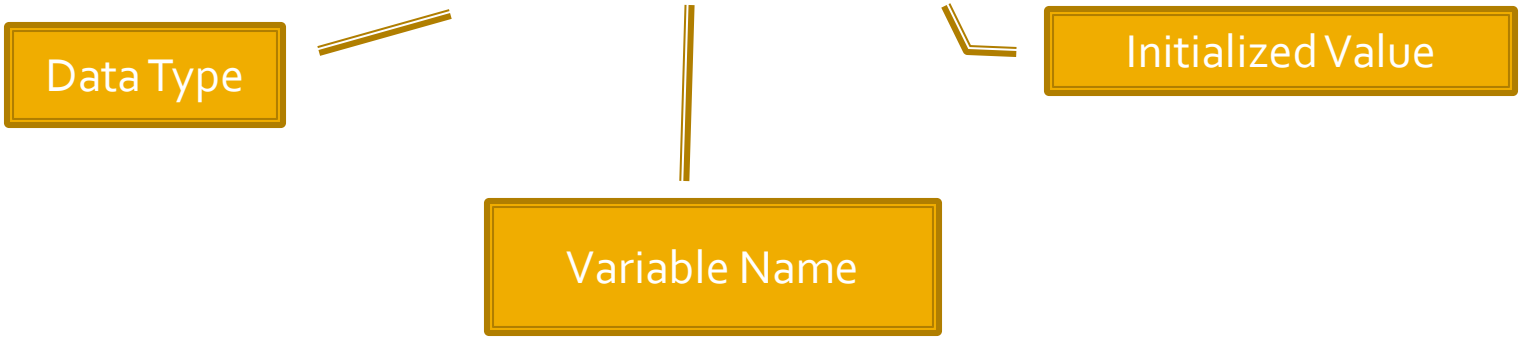
Table 1 Variable Declarations in Java

Variable Name	Comment
`int cans = 6;`	Declares an integer variable and initializes it with 6.
`int total = cans + bottles;`	The initial value need not be a constant. (Of course, `cans` and `bottles` must have been previously declared.)
`bottles = 1;`	**Error:** The type is missing. This statement is not a declaration but an assignment of a new value to an existing variable—see Section 2.2.
`int bottles = "10";`	**Error:** You cannot initialize a number with a string.
`int bottles;`	Declares an integer variable without initializing it. This can be a cause for errors—see Common Error 2.1 on page 37.
`int cans, bottles;`	Declares two integer variables in a single statement. In this book, we will declare each variable in a separate statement.

Variable Initialization

- When declaring a variable, you usually **initialize** it.
  - This means that you set the value that should be stored in the variable.

int cansPerPack = 6;



Variables – Code Conventions

- Applications generally use many variables
- Variable declarations should be grouped together at the beginning of a method or procedure.
- Put a blank line under the declarations

```
public class RectangleArea {

   public static void main(String[] args) {
       int length = 10;      //longer side of rectangle
       int width = 2;        //shorter side of rectangle
       int area;             //calculated area of rectangle

       area = length * width;
       System.out.println("Area of rectangle: " + area);

   }
}
```

Variable Naming Rules

- Names must start with a letter. Use only letters, digits or the underscore (_) in the rest of the name.
- Don't start with a numbers or use special characters ($,%,@)
- Separate words with 'camelHump' notation
- Variable names are case-sensitive
- Don't use reserved 'Java' keywords
- Variables start with lower case, class names with upper case (by convention, not a 'rule')

Variable Naming Rules

Table 3 Variable Names in Java

Variable Name	Comment
`canVolume1`	Variable names consist of letters, numbers, and the underscore character.
`x`	In mathematics, you use short variable names such as *x* or *y*. This is legal in Java, but not very common, because it can make programs harder to understand (see Programming Tip 2.1 on page 38).
⚠ `CanVolume`	**Caution:** Variable names are case sensitive. This variable name is different from `canVolume`, and it violates the convention that variable names should start with a lowercase letter.
⊘ `6pack`	**Error:** Variable names cannot start with a number.
⊘ `can volume`	**Error:** Variable names cannot contain spaces.
⊘ `double`	**Error:** You cannot use a reserved word as a variable name.
⊘ `ltr/fl.oz`	**Error:** You cannot use symbols such as / or .

Constant Variables

Constants

- It is good practice to declare values that do not change as 'constants'
  - Use the reserved word `final` before the type

```
final double BOTTLE_VOLUME = 1.75;
```

- Then use the constant name instead of the value:
  - They can be used by name just like variables

```
double volume = bottles * BOTTLE_VOLUME;
```

- Constants are usually declared near the beginning of a program or a class
  - If you edit the constant value and re-compile, and the rest of the code will use the new value!

You cannot assign a new value to a constant at run-time.

Data Types

Data Types

- Data type determines what kind of data the variable can store
- Different data types can store
  A. different kinds of data
  B. different sizes of data

int

- Positive or negative numbers

-2,147,483,648 to 2, 147,483,648

double

- A positive or negative number that may contain a decimal

-1.7e+308 to 1.7e+308

Table 2 Number Literals in Java

Number	Type	Comment
`6`	`int`	An integer has no fractional part.
`-6`	`int`	Integers can be negative.
`0`	`int`	Zero is an integer.
`0.5`	`double`	A number with a fractional part has type `double`.
`1.0`	`double`	An integer with a fractional part .0 has type `double`.
`1E6`	`double`	A number in exponential notation: 1×10^6 or 1000000. Numbers in exponential notation always have type `double`.
`2.96E-2`	`double`	Negative exponent: $2.96 \times 10^{-2} = 2.96 / 100 = 0.0296$
`100,000`		**Error:** Do not use a comma as a decimal separator.
`3 1/2`		**Error:** Do not use fractions; use decimal notation: 3.5

Assignment Statements

Assignment

- The details of the assignment statement
  - The value on the right is copied to the variable on the left

```
double total = 0;
   .
   .
total = bottleVolume;
   .
   .
total = total + canVolume;
```

This is an initialization of a new variable, NOT an assignment.

This is an assignment.

The name of a previously defined variable

The expression that replaces the previous value

The same name can occur on both sides. See Figure 1.

Adding Comments

Comments

- There are three forms of comments:
  - 1: // single line (or rest of line to right)

    ```
    double canVolume = 0.355;          //Liters is a 12-once can
    ```
  - 2: /*

    ```
    multi-line – all comment until matching
    */
    ```
  - 3: /**

    ```
    multi-line Javadoc comments
    */
    ```

Use comments to add explanations for humans who read your code. The compiler ignores comments.

- Use comments at the beginning of each program, and to clarify details of the code

```
1  /**
2     This program computes the volume (in liters) of a six-pack of soda cans.
3  */
4  public class Volume1
5  {
6     public static void main(String[] args)
7     {
8        int cansPerPack = 6;
9        double canVolume = 0.355; // Liters in a 12-ounce can
10
11       System.out.print("A six-pack of 12-ounce cans contains ");
12       System.out.print(cansPerPack * canVolume);
13       System.out.println(" liters.");
14    }
15 }
```

- Lines 1-3 are Javadoc comments for the class 'Volume1'
- Line 9 uses a single-line comment to clarify the unit of measurement

Obtaining Input

Obtaining Input

- You might need to ask for input (aka prompt for input) and then save what was entered.
- This is a three step process in Java
  1) Import the `Scanner` class from its 'package' `java.util`

     ```
     import java.util.Scanner;
     ```

  2) Setup an object of the `Scanner` class

     ```
     Scanner in = new Scanner(System.in);
     ```

  3) Use methods of the new `Scanner` object to get input

     ```
     int bottles = in.nextInt();
     double price = in.nextDouble();
     ```

Input Statement

- The `Scanner` class allows you to read keyboard input from the user
  - It is part of the Java API `util` package

Include this line so you can use the `Scanner` class.

Create a `Scanner` object to read keyboard input.

Display a prompt in the console window.

Define a variable to hold the input value.

```
import java.util.Scanner;
.
.
Scanner in = new Scanner(System.in);
.
.
System.out.print("Please enter the number of bottles: ");
int bottles = in.nextInt();
```

Don't use `println` here.

The program waits for user input, then places the input into the variable.