ITP20003 Java Programming

Basic Operations
(Chapter 2)

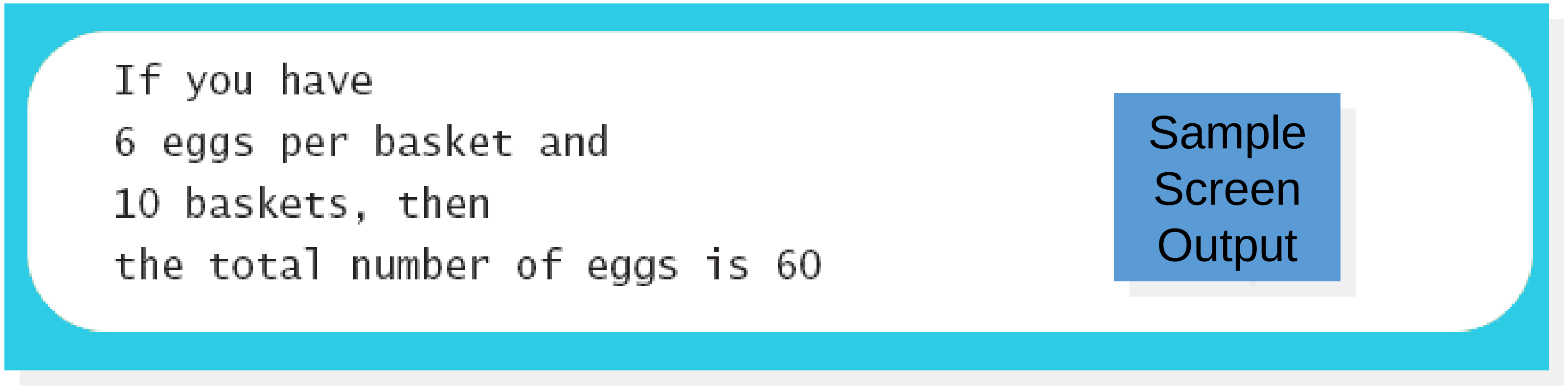
This slide is primary taken from the instructor's resource of Java: Introduction to Problem Solving and Programming, 7th ed. by Savitch and then edited partly by Shin Hong

Variables

- *Variables* store data such as numbers and letters.
  - Think of them as places to store data.
  - They are implemented as memory locations.
- The data stored by a variable is called its *value.*
  - The value is stored in the memory location.
- Its value can be changed.

Variables

- View sample program listing 2.1
  - **`Class EggBasket`**



```
If you have
6 eggs per basket and
10 baskets, then
the total number of eggs is 60
```

Example

```java
public class EggBasket
{
    public static void main (String [] args)
    {
      int numberOfBaskets, eggsPerBasket, totalEggs;
      numberOfBaskets = 10;
      eggsPerBasket = 6;
      totalEggs = numberOfBaskets * eggsPerBasket;
      System.out.println ("If you have");
      System.out.println (eggsPerBasket + " eggs per basket and");
      System.out.println (numberOfBaskets + " baskets, then");
      System.out.println ("the total number of eggs is " +
                           totalEggs);
    }
}
```

Variables and Values

- Variables

```
numberOfBaskets
eggsPerBasket
totalEggs
```

- Assigning values

```
eggsPerBasket = 6;
eggsPerBasket = eggsPerBasket - 2;
```

Naming and Declaring Variables

- Choose names that are helpful such as `count` or `speed`, but not `c` or `s`.
- When you *declare* a variable, you provide its name and type.

```
int numberOfBaskets,eggsPerBasket;
```

- A variable's *type* determines what kinds of values it can hold (`int`, `double`, `char`, etc.).
- A variable must be declared before it is used.

Syntax and Examples

- Syntax

  ```
  type variable_1, variable_2, …;
  ```

  (`variable_1` is a generic variable called a *syntactic variable*)

- Examples

  ```
  int styleChoice, numberOfChecks;
  double balance, interestRate;
  char jointOrIndividual;
  ```

Data Types

- A *class type* is used for a class of objects and has both data and methods.
  - `"Java is fun"` is a value of class type `String`
- A *primitive type* is used for simple, non-decomposable values such as an individual number or individual character.
  - `int`, `double`, and `char` are primitive types.

Primitive Types

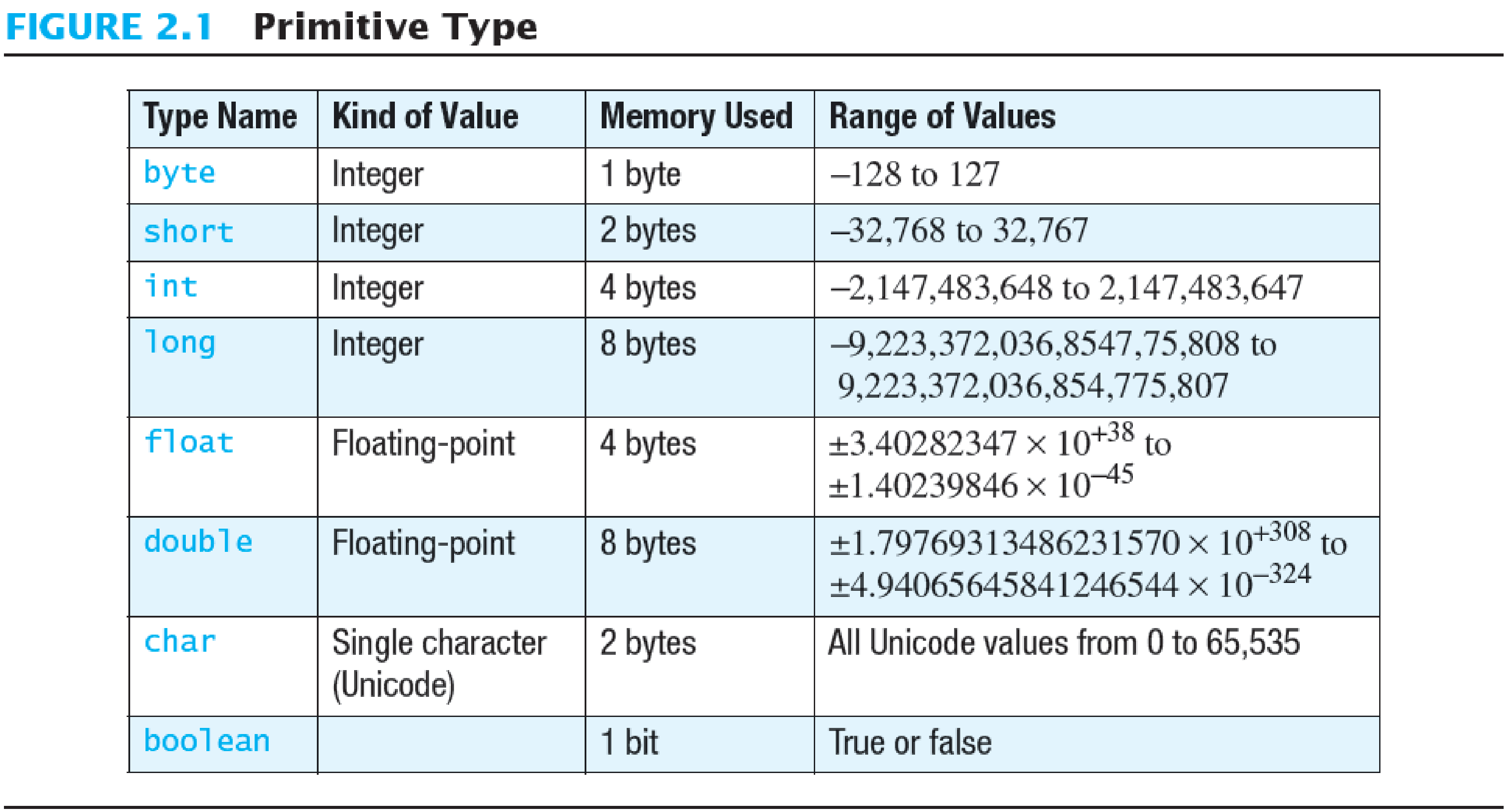


FIGURE 2.1 Primitive Type

Type Name	Kind of Value	Memory Used	Range of Values
`byte`	Integer	1 byte	–128 to 127
`short`	Integer	2 bytes	–32,768 to 32,767
`int`	Integer	4 bytes	–2,147,483,648 to 2,147,483,647
`long`	Integer	8 bytes	–9,223,372,036,8547,75,808 to 9,223,372,036,854,775,807
`float`	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$
`double`	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
`char`	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
`boolean`		1 bit	True or false

Java Identifiers

- An *identifier* is a name, such as the name of a variable
- Identifiers may contain only
  - Letters
  - Digits (0 through 9)
  - The underscore character (_)
  - And the dollar sign symbol ($) which has a special meaning
- The first character <u>cannot</u> be a digit.

Java Identifiers

- Identifiers may not contain any spaces, dots (`.`), asterisks (`*`), or other characters:

  `7-11   oracle.com   util.*` (not allowed)

- Identifiers can be arbitrarily long.
- Since Java is *case sensitive*, **`stuff`**, **`Stuff`**, and **`STUFF`** are different identifiers.

Keywords or Reserved Words

- Words such as `if` are called *keywords* or *reserved words* and have special, predefined meanings.
  - Cannot be used as identifiers.
  - See Appendix 1 for a complete list of Java keywords.
- Example keywords: `int, public, class`

Naming Conventions

- Class types begin with an uppercase letter (e.g. `String`).
- Primitive types begin with a lowercase letter (e.g. `int`).
- Variables of both class and primitive types begin with a lowercase letters (e.g. `myName`, `myBalance`)
- Multiword names are "punctuated" using uppercase letters.

Where to Declare Variables

- Declare a variable
  - Just before it is used or
  - At the beginning of the section of your program that is enclosed in `{}`.

```
public static void main(String[] args)
{ /* declare variables here */
     . . .
}
```

Primitive Types

- Four integer types (`byte`, `short`, `int`, and `long`)
  - `int` is most common
- Two floating-point types (`float` and `double`)
  - `double` is more common
- One character type (`char`)
- One boolean type (`boolean`)

Examples of Primitive Values

- Integer types

  `0   -1   365   12000`

- Floating-point types

  `0.99   -22.8   3.14159 5.0`

- Character type

  `'a'   'A'   '#'   ' '`

- Boolean type

  `true   false`

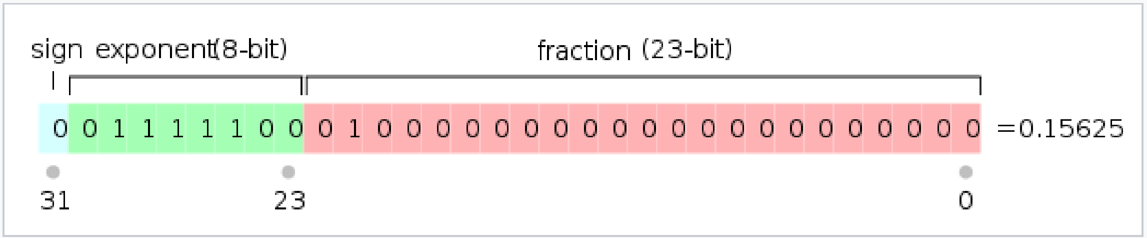
e Notation

- e notation is also called *scientific notation* or *floating-point notation.*
- Examples
  - `865000000.0` can be written as `8.65e8f` or `8.65e8d`
  - `0.000483` can be written as `4.83e-4f` or `4.83e-4d`
- The number in front of the `e` does not need to contain a decimal point.

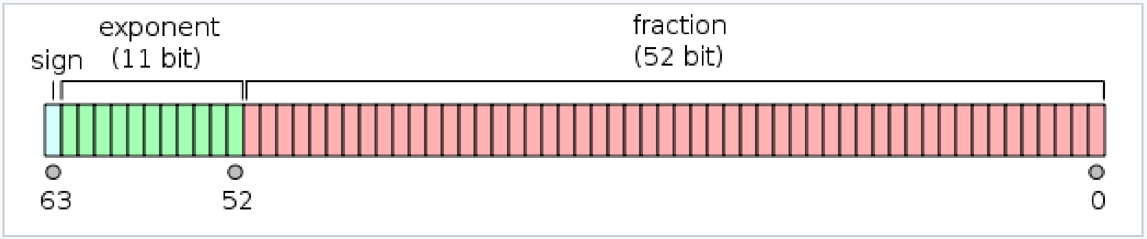
Floating Number Representation

- Ref. https://en.wikipedia.org/wiki/IEEE_754-1985

$$(-1)^{b_{31}} \times 2^{(b_{30}b_{29}\ldots b_{23})_2-127} \times (1.b_{22}b_{21}\ldots b_0)_2$$



The number 0.15625 represented as a single-precision IEEE 754-1985 floating-point number. See text for explanation.



The three fields in a 64bit IEEE 754 float

Imprecision in Floating-Point Numbers

- Floating-point numbers often are only approximations since they are stored with a finite number of bits.
- Hence `1.0/3.0` is slightly less than 1/3.
- `1.0/3.0 + 1.0/3.0 + 1.0/3.0`
  is less than 1.

Assignment Statements

- An assignment statement is used to assign a value to a variable.

  ```
  answer = 42;
  ```

- The "equal sign" is called the *assignment operator.*
- We say, "The variable named `answer` is assigned a value of 42," or more simply, "`answer` is assigned 42."

Assignment Statements

- Syntax

  `variable = expression`

  where **`expression`** can be another variable, a *literal* or *constant* (such as a number), or something more complicated which combines variables and literals using *operators*
  (such as + and -)

Assignment Examples

```
amount = 3.99;
firstInitial = 'W';
score = numberOfCards + handicap;
eggsPerBasket = eggsPerBasket - 2;
```

Initializing Variables

- A variable that has been declared, but no yet given a value is said to be *uninitialized.*
- Uninitialized class variables have the value **`null`**.
- Uninitialized primitive variables may have a default value.
- It's good practice not to rely on a default value.

Initializing Variables

- To protect against an uninitialized variable (and to keep the compiler happy), assign a value at the time the variable is declared.
- Examples:

```
int count = 0;
char grade = 'A';
```

Initializing Variables

- syntax

```
type variable_1 = expression_1,
variable_2 = expression_2, …;
```

Assignment Evaluation

- The expression on the right-hand side of the assignment operator (=) is evaluated first.
- The result is used to set the value of the variable on the left-hand side of the assignment operator.

```
score = numberOfCards +  handicap;
eggsPerBasket = eggsPerBasket - 2;
```