

TM105: Introduction to Computer Programming

Meeting 2 (Chapter 2)

Elementary Programming

Motivations

- In the preceding chapter, you learned how to create, compile, and run a Java program.
- Starting from this chapter, you will learn how to solve practical problems programmatically.
- Through these problems, you will learn Java primitive data types and related subjects, such as variables, constants, data types, operators, expressions, and input and output.



Objectives

- To write Java programs to perform simple computations.
- To obtain input from the console using the **Scanner** class.
- To use identifiers to name variables, constants, methods, and classes.
- To name classes, methods, variables, and constants by following their naming conventions.
- To explore Java numeric primitive data types: **byte**, **short**, **int**, **long**, **float**, and **double** .
- To perform operations using operators +, **-**, *, /, and **%** .
- To write and evaluate numeric expressions.
- To cast the value of one type to another type.
- To read a variable value from the keyboard.



Variables

- **Integers**
  - Whole numbers; like 7, 0 and –22
- Programs remember numbers and other data in the computer's memory by using variables.
- **Variable**
  - A location in the computer's memory where a value can be stored.
  - Must be declared with a name and a type before they can be used.
  - The **name** can be any **valid identifier**.
  - The **type** specifies what kind of information is stored at that location in memory.



```java
// A small program to add 2 integers
public class Addition1 {
   public static void main(String[] args) {
      // declaration
      int number1; // first number to add
      int number2; // second number to add
      int sum; // sum of number1 and number2
      // initialization
      number1 = 45;
      number2 = 72;
      // add numbers, then store total in sum
      sum = number1 + number2;
      System.out.println("Sum is " + sum); // display sum
   } // end main method
} // end class Addition1
```

```
-----output-----
Sum is 117
```



Variable declaration

- Variable declaration statement
  - `int number1;`
  - Specifies a name (`number1`) and a type (`int`) of a variable that is used in this program.
  - Several variables of the same type may be declared in one declaration with the variable names separated by commas; e.g.:
    - `int number1, number2, sum;`
- Assignment statement

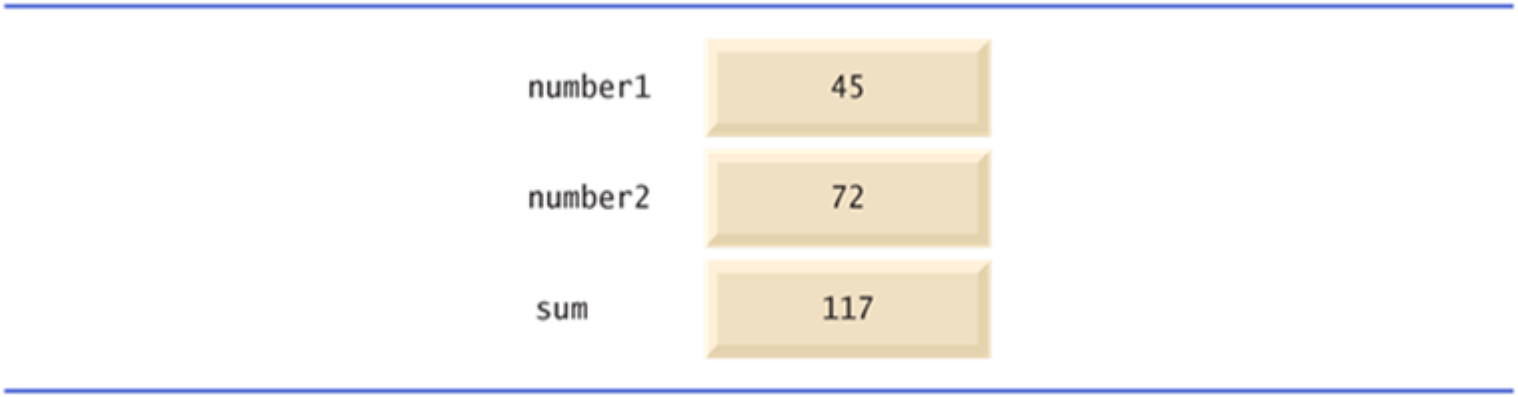
a variable → `number1 = 45;` ← an expression (or a literal)

  - The assignment operator (=) indicates that the integer literal (`45`) to the right will be assigned to (stored in the memory location of) the variable (`number1`) to the left.
  - When a value assigned to a variable for the first time, we say that the variable has been initialized (i.e., prepared for use in the program).
  - It is not allowed to use a (local) variable before initializing it.



Variable declaration

- Variables
  - Every variable has a name, a type, a size (in bytes) and a value.
  - When a new value is placed into a variable, the



The (+) operator

- The operator (+) as an addition operator:

```
sum = number1 + number2;
```

  - `number1` and `number2` are called operands.
  - The operator (+) is called a binary operator because it operates on two operands.
- The operator (+) as a concatenation operator:
  - `"Sum is " + sum`
- If any one of the two operands is a string, the result will be a string (the first operand concatenated with the second operand).
  - Examples:
    - `"Welcome to " + "Java!"`
    - `"Sum is " + sum`
    - `sum + " is the result."`



Primitive data types

- Java is a strongly typed language: requires all variables to have a type.
- The following table summarizes the eight **primitive types** in Java:

Primitive Data Types				
Category	**Description**	**JAVA**	**Size**	**Values**
numbers	whole numbers (without decimal points)	`byte`	1 byte (8 bits)	-128 to 127 (i.e. -2^7 to 2^7-1)
		`short`	2 bytes	-32768 to 32767
		`int`	4 bytes	-2147483648 to 2147483647
		`long`	8 bytes	-2^{63} to $2^{63}-1$
	real numbers	`float`	4 bytes	e.g. 17.345f
		`double`	8 bytes	e.g. 12452.212 (more accurate)
characters	single characters	`char`	2 bytes	e.g. 'a', '1' and '?'
booleans	boolean values	`boolean`	It is specific to JVM on each platform.	true or false

Primitive data types

- All primitive data types are written in small letters (they are keywords).
- Characters are represented using Unicode, so they are stored in 16 bits, for example:
  - '4' stored in memory as $(0000\ 0000\ 0011\ 0100)_2 = (52)_{10}$
  - '5': 53
  - 'A': 65
  - 'a': 97
  - 'c': 99
- Characters could be non-printable ones such as: tab, enter, etc.



Primitive data types

- Examples:
  - `int i1 = 35;` ← int literal
  - `int i2 = (7 + 2);` ← int expression
  - `long l = 9876543210L;` ← long literal
  - `double d = 35.4;` ← double literal
  - `float f = 35.4f;` ← float literal
  - `char c1 = 'a';` ← char literal
  - `char c2 = ' '; // space character`
  - `char c3 = '\n'; // new line character`
  - `boolean b1 = true;` ← boolean literal
  - `boolean b2 = (10 <= 7);` ← boolean expression



Primitive data types

- **Characters** are delimited (enclosed) using single quotes where **strings** are delimited using double quotes.
- Compare the following:
  - 5 + 4 = 9 //Numbers
  - "5" + "4" = "54 “ //Strings
  - '5' + '4' = 105 //Characters



Identifiers

- An identifier is a sequence of characters that consist of letters, digits, underscores (_), and dollar signs ($).
- An identifier must start with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit.
- An identifier cannot be a reserved word.
  - See Appendix A, "Java Keywords," for a list of reserved words.
- An identifier cannot be `true`, `false`, or `null`.
- An identifier can be of any length.



Naming Conventions

- Choose meaningful and descriptive names.
- **Variables and method names:**
  - Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name. For example, the variables radius and area, and the method computeArea.
- **Class names:**
  - Capitalize the first letter of each word in the name. For example, the class name ComputeArea.



Another program

Computing the Area of a Circle: This program computes the area of the circle.

ComputeArea

Run

Note: Clicking the green button displays the source code with interactive animation. You can also run the code in a browser. Internet connection is needed for this button.

Note: Clicking the blue button runs the code from Windows. If you cannot run the buttons, see

IMPORTANT NOTE: If you cannot run the buttons, see liveexample.pearsoncmg.com/slide/javaslidenote.doc.

Trace a Program Execution

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

allocate memory for radius

radius no value



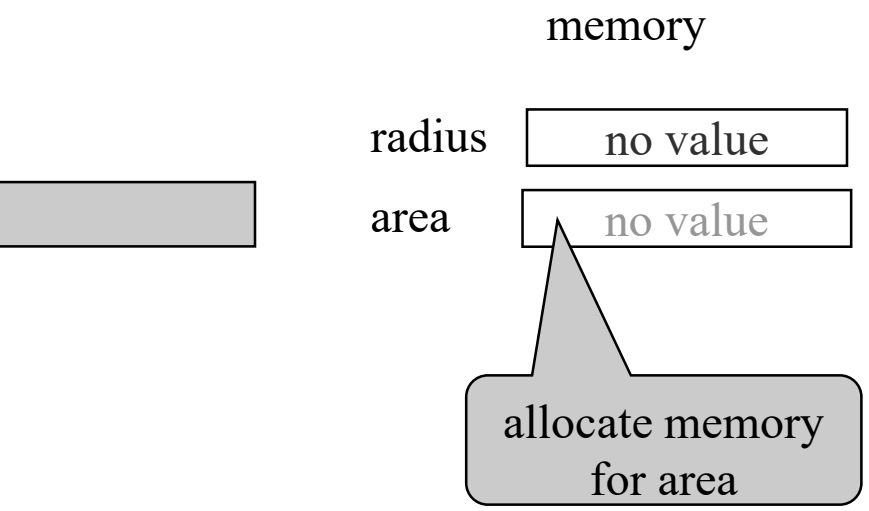
Trace a Program Execution

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```





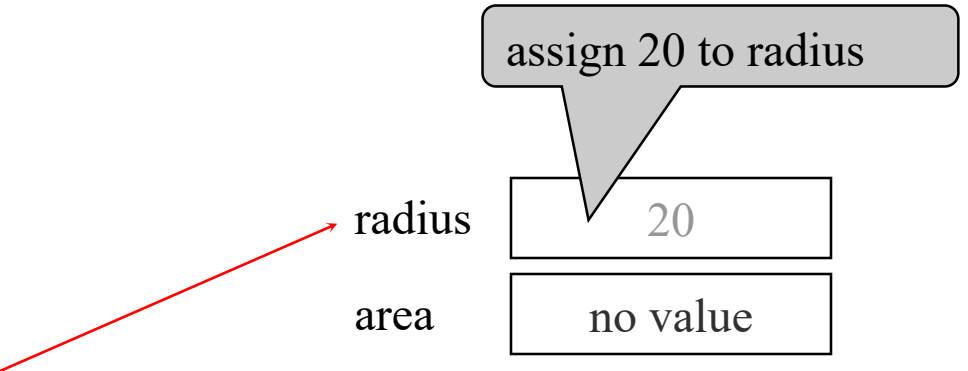
Trace a Program Execution

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```





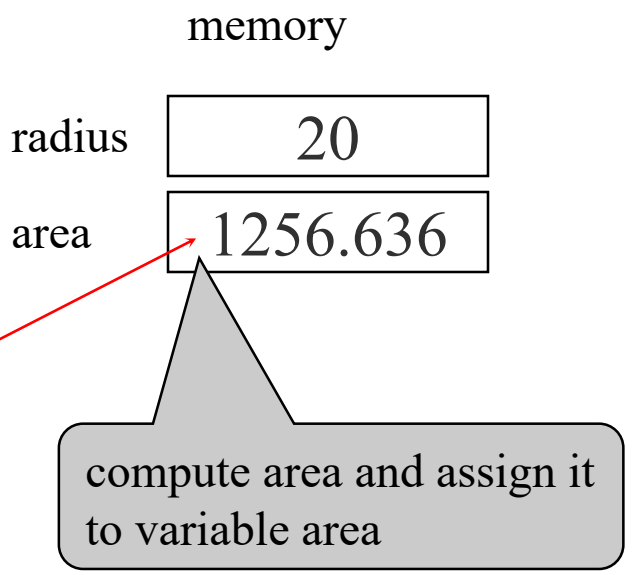
Trace a Program Execution

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```





Trace a Program Execution

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // Assign a radius
    radius = 20;

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

memory

radius	20
area	1256.636

print a message to the console



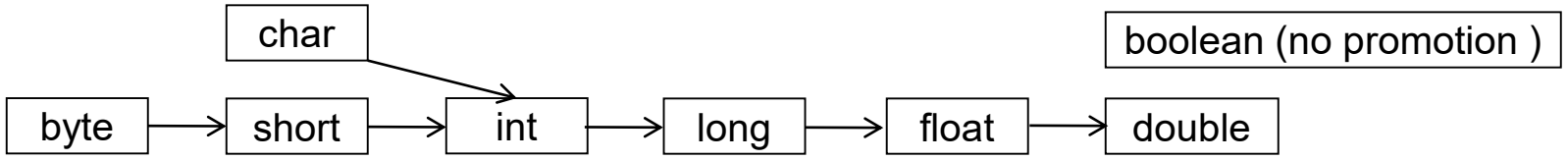
Promotion

- Promotion (or implicit conversion)

```
radius = 20; // no error
```

  - The `int` literals (or expressions) could be promoted to `double`. (`20` becomes `20.0`)

- The primitive types could be arranged in the following order:



Such that it is allowed to assign a value of one type to a variable of any type to its right but **not the vice versa**. (To do that a casting (or explicit conversion) need to be used)

```
double d = 72; // no error
int num1 = 72.5; // compilation error
int num2 = 72.0; // compilation error
```



Numeric Operators

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Integer Division

- 5 / 2 yields an integer 2.
- 5.0 / 2 yields a double value 2.5
- 5 % 2 yields 1 (the remainder of the division)



Number Literals

- A literal is a constant value that appears directly in the program.
- For example, 34, 1,000,000, and 5.0 are literals in the following statements:
  - `int i = 34;`
  - `long x = 1000000;`
  - `double d = 5.0;`



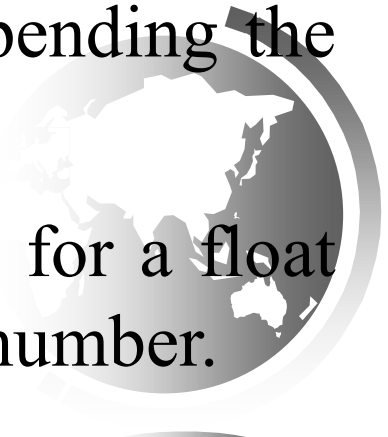
Integer Literals

- An integer literal can be assigned to an integer variable as long as it can fit into the variable.
- A compilation error would occur if the literal were too large for the variable to hold.
- For example, the statement byte b = 1000 would cause a compilation error, because 1000 cannot be stored in a variable of the byte type.
- An integer literal is assumed to be of the int type, whose value is between -2^{31} (-2147483648) to $2^{31}-1$ (2147483647).
- To denote an integer literal of the long type, append it with the letter L or l.
  - L is preferred because l (lowercase L) can easily be confused with 1 (the digit one).



Floating-Point Literals

- Floating-point literals are written with a decimal point.
- By default, a floating-point literal is treated as a double type value.
- For example, 5.0 is considered a double value, not a float value.
- You can make a number a float by appending the letter f or F, and make a number a double by appending the letter d or D.
- For example, you can use 100.2f or 100.2F for a float number, and 100.2d or 100.2D for a double number.



double vs. float

The double type values are more accurate than the float type values. For example,

```
System.out.println("1.0 / 3.0 is " + 1.0 / 3.0);
```

displays `1.0 / 3.0 is 0.3333333333333333`

16 digits

```
System.out.println("1.0F / 3.0F is " + 1.0F / 3.0F);
```

displays `1.0F / 3.0F is 0.33333334`

7 digits



Arithmetic Expressions

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

is written in Java as:

```
(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)
```



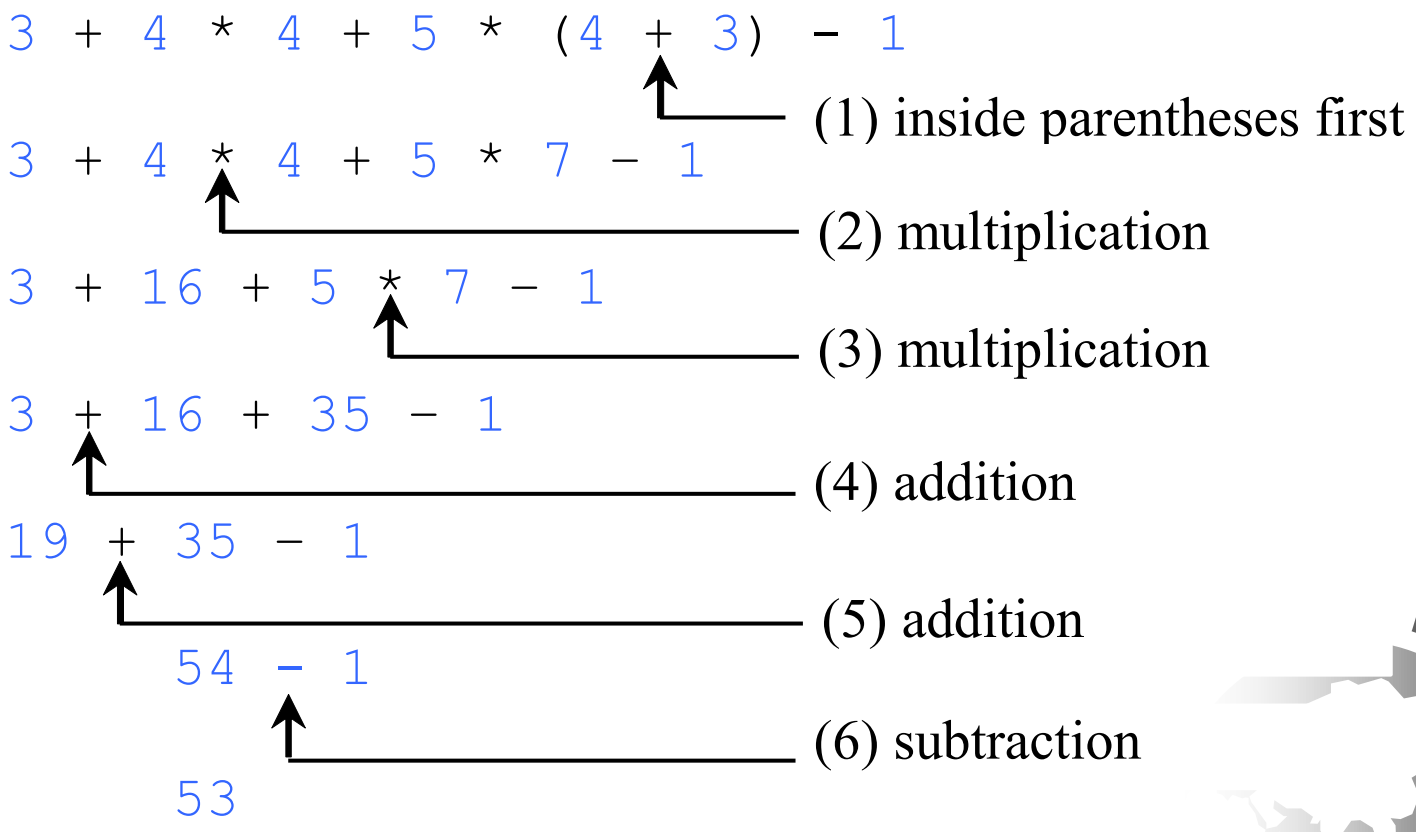
How to Evaluate an Expression

Operator(s)	Operation(s)	Order of evaluation (precedence)
*	Multiplication	Evaluated first. If there are several operators of this type, they're evaluated from left to right.
/	Division	
%	Remainder	
+	Addition	Evaluated next. If there are several operators of this type, they're evaluated from left to right.
-	Subtraction	
=	Assignment	Evaluated last.

Rules of operator precedence



How to Evaluate an Expression



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Reading Numbers from the Keyboard

```
Scanner input = new Scanner(System.in);
int value = input.nextInt();
```

Method	Description
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

```java
// A small program that reads 2 integers from the user using
// the class Scanner, computes and displays their sum and average
import java.util.Scanner;
public class SumAverage {
   public static void main(String[] args) {
      // create a Scanner to obtain input from the command window
      Scanner input = new Scanner(System.in);
      int number1, number2, sum;
      System.out.print("Enter first integer: "); // prompt
      number1 = input.nextInt(); // read 1st number from user
      System.out.print("Enter second integer: "); // prompt
      number2 = input.nextInt(); // read 2nd number from user
      sum = number1 + number2;
      double average = (double) sum / 2;
      String message = "Sum is";
      System.out.printf("%s %d\n", message, sum);
      System.out.printf("Average is %.2f\n", average);
   } // end main method
} // end class SumAverage
```

```
----output----
Enter first integer: 5
Enter second integer: 8
Sum is 13
Average is 6.50
```

Try to remove the rounding and see the result

Reading Numbers from the Keyboard

- import declaration
  - Helps the compiler locate a class that is used in this program.
  - Classes are grouped into packages, or Java class library, or the Java Application Programming Interface (Java API).
  - A Package is a named group of related classes.
  - All `import` declarations must appear before the first class declaration.
  - Placing an `import` declaration inside or after a class declaration is a syntax error.



Reading Numbers from the Keyboard

```
System.out.print("Enter first integer: ");
```

- Is an output statement that directs the user to take a specific action.
- `System` is a class:
  - Part of package `java.lang`.
  - Class `System` is not imported with an `import` declaration at the beginning of the program.
  - Be default, package `java.lang` is imported in every Java program.
    - i.e. don't require an `import` declaration.



Reading Numbers from the Keyboard

- `Scanner` method `nextInt()`

```
number1 = input.nextInt();
```

  - Obtains an integer from the user at the keyboard.
  - Program waits for the user to type the number and press the Enter key to submit the number to the program.



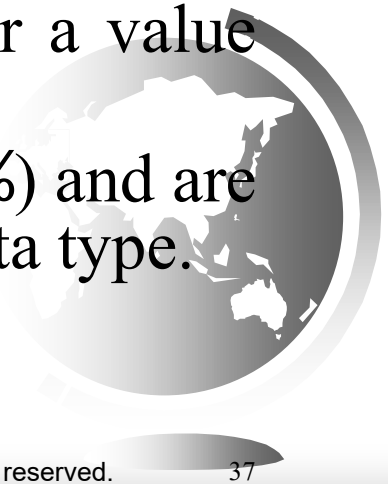
Reading Numbers from the Keyboard

- Type casting (or explicit conversion)
  - Java evaluates only arithmetic expressions in which the operands' types are identical.
  - To perform a floating-point calculation with integers, Java temporarily treat these values as floating-point numbers.
  - The unary cast operator (double) creates a temporary floating-point copy of its operand.
  - The value stored in the operand is **unchanged**.
  - Cast operators are available for any type.
  - Casting to the wrong type may cause compilation or runtime errors; E.g.:
    - `int num = (int) 72.8; // num = 72`
  - Cast operators associate from right to left.



Displaying Text with `printf`

- System.out.printf method
  - `f` means "formatted"
  - displays formatted data
- Multiple method arguments are placed in a comma-separated list.
- Method `printf`'s first argument is a format **string**.
  - May consist of fixed text and format specifiers.
  - Each format specifier is a placeholder for a value and specifies the type of data to output.
- Format specifiers begin with a percent sign (%) and are followed by a character that represents the data type.



Displaying Text with `printf`

- Formatting output

```
System.out.printf("%s %d\n", message, sum);
```

  - Format specifier `%s` is a placeholder for a string.
  - Format specifier `%d` is a placeholder for an integer value (`byte`, `short`, `int` and `long`). The letter `d` stands for "decimal integer".

```
System.out.printf("Average is %.2f\n", average);
```

  - Format specifier `%f` is a placeholder for a floating point value (`float` and `double`). Using `%.2f` rounds the number to exactly 2 decimal places.

  - Format specifier `%c` is a placeholder for a `char` value.
  - Format specifier `%b` is a placeholder for a `boolean` value.
  - Format specifier `%n` is a line separator but cannot be used with `System.out.print` or `println`.



```java
// To read different types of information from the user
import java.util.Scanner;
public class ReadingFromUser {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      String name, profession;
      double height;   char gender;
      System.out.print("Enter your name: ");
      name = input.nextLine();
      System.out.print("Enter your profession: ");
      profession = input.next();
      System.out.print("Enter your height: ");
      height = input.nextDouble();
      System.out.print("Enter your gender (M/F): ");
      gender = input.next().charAt(0);
      System.out.println("-------");
      System.out.printf("%s %s%n%s %s%n%s %.2f%n%s %c%n",
                        "Name:", name, "Profession:", profession,
                        "Height:", height, "Gender:", gender);
   } // end main method
} // end class
```

```
----output----
Enter your name: Adel Omar
Enter your profession: Engineer
Enter your height: 1.87
Enter your gender (M/F): M
-------
Name: Adel Omar
Profession: Engineer
Height: 1.87
Gender: M
```



Reading from the user

- methods `nextLine()` and `next()`
  - Obtain a string value from the user at the keyboard.
  - `nextLine()` reads characters entered by the user (including space characters) till the newline character is reached.
  - `next()` reads characters entered by the user till the space character is reached.
- method `nextDouble()`
  - Obtains a double value from the user at the keyboard.
- method `charAt()`
  - Is a `String` method.
  - `charAt` returns the character at a specific position in a string.
  - To read a single character from the user, use the method `next()` and then select the character at index 0 using `charAt(0)`.



Reading Input from the Console

Two more examples:

ComputeAreaWithConsoleInput	Run
ComputeAverage	Run



Problem: Converting Temperatures

Write a program that converts a Fahrenheit degree to Celsius using the formula:

$$celsius = (\frac{5}{9})(fahrenheit - 32)$$

Note: you have to write
celsius = (5.0 / 9) * (fahrenheit – 32)

FahrenheitToCelsius Run



Thanks!

