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.

Introducing Programming with an Example

Listing 2.1 Computing the Area of a Circle

This program computes the area of the circle.

animation

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);
  }
}
```

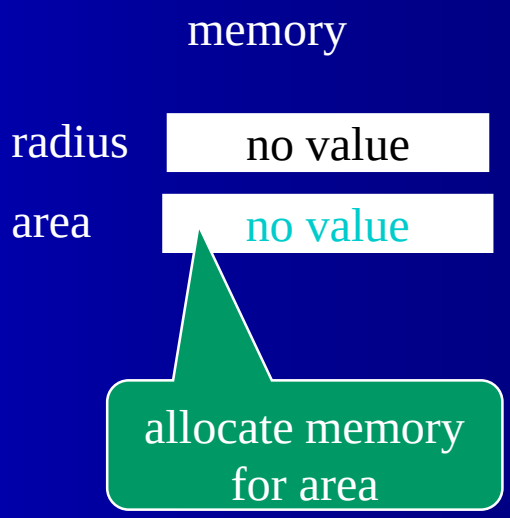
allocate memory for radius

radius | no value

animation

Trace a Program Execution

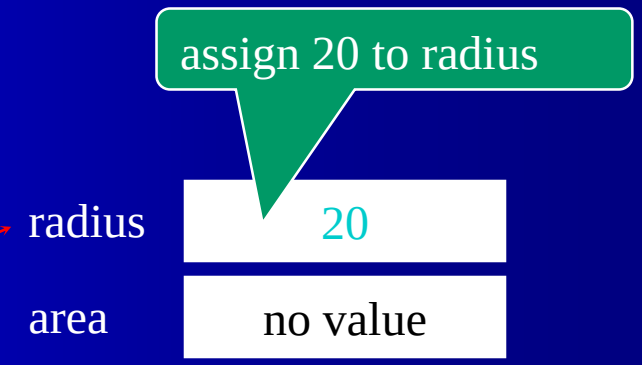
```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // 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);
  }
}
```



animation

Trace a Program Execution

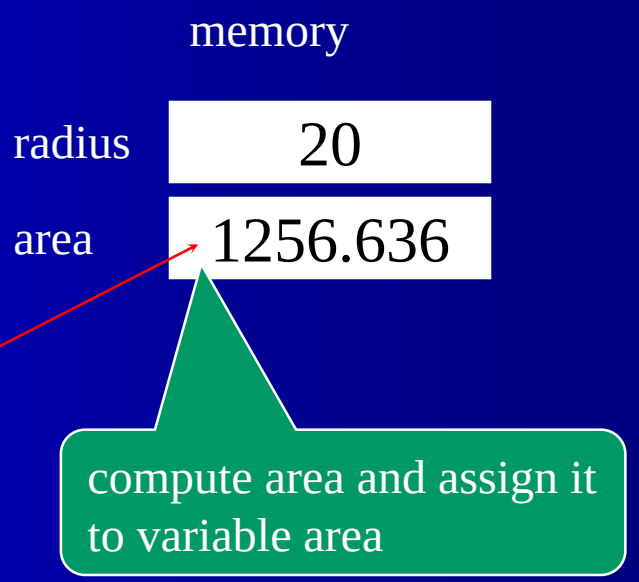
```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // 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);
  }
}
```



animation

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);
  }
}
```



animation

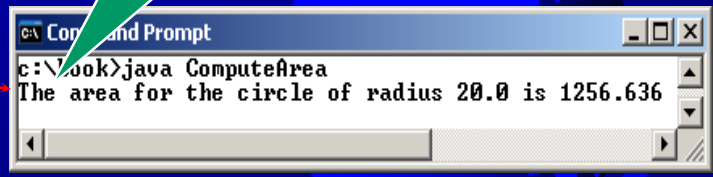
Trace a Program Execution

```
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



I made 2 versions

ComputeArea.java

ComputeAreaInWindow.java

```
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);
  }
}
```

```
import javax.swing.JOptionPane;

public class ComputeAreaInWindow {
  /** 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 in a window
    JOptionPane.showMessageDialog(null,"radius " + radius + " is " + area,"Circle Area with
",JOptionPane.INFORMATION_MESSAGE);

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

Now let's enter some data from the console

1. Create a Scanner object

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

2. Use the methods:

next(), nextByte(), nextShort(), nextInt(), nextLong(), nextFloat(), nextDouble(), or nextBoolean()

Two examples

ComputeAreaWithConsoleInput.java

ComputeAverage.java

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
  public static void main(String[] args) {
    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a radius
    System.out.print("Enter a number for radius: ");
    double radius = input.nextDouble();

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

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

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAverage {
  public static void main(String[] args) {
    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter three numbers
    System.out.print("Enter three numbers: ");
    double number1 = input.nextDouble();
    double number2 = input.nextDouble();
    double number3 = input.nextDouble();

    // Compute average
    double average = (number1 + number2 + number3) / 3;

    // Display result
    System.out.println("The average of " + number1 + " " + number2
      + " " + number3 + " is " + average);
  }
}
```

Identifiers

- is a sequence of characters that consist of:
  - letters
  - digits
  - underscores (_)
  - dollar signs ($).
- must **start** with a letter an underscore (_), or a dollar sign ($). It cannot start with a digit.
- cannot be a reserved words
- cannot be true, false, or null.
- can be of any length.

Declaring Variables

```
int x;            // Declare x to be an
                  // integer variable;
double radius;    // Declare radius to
                  // be a double variable;
char a;           // Declare a to be a
                  // character variable;
```

Assignment Statements

```
x = 1;                  // Assign 1 to x;

radius = 1.0;           // Assign 1.0 to radius;

a = 'A';                // Assign 'A' to a;
```

Declaring and Initializing in One Step

- `int x = 1;`
- `double d = 1.4;`

Named Constants

```
final –datatype- -CONSTANTNAME- = -someValue-;
```

```
Examples:
final double PI = 3.14159;
final int SIZE = 3;
```

Naming Conventions

- Choose meaningful and descriptive names.
- Variables and method names:
  - Use lowercase.
  - `Examples:`
    - `radius`
    - `area`
    - the method `computeArea`.

Naming Conventions, cont.

- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `ComputeArea`.

- Constants:
  - Capitalize all letters with underscores
  - For example:
    - `PI`
    - `MAX_VALUE`

Numerical Data Types

Name	Range	Storage Size
`byte`	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed
`short`	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
`int`	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
`long`	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
`float`	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
`double`	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

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

Integer Division

+, -, *, /, and %

5 / 2 yields an integer 2.

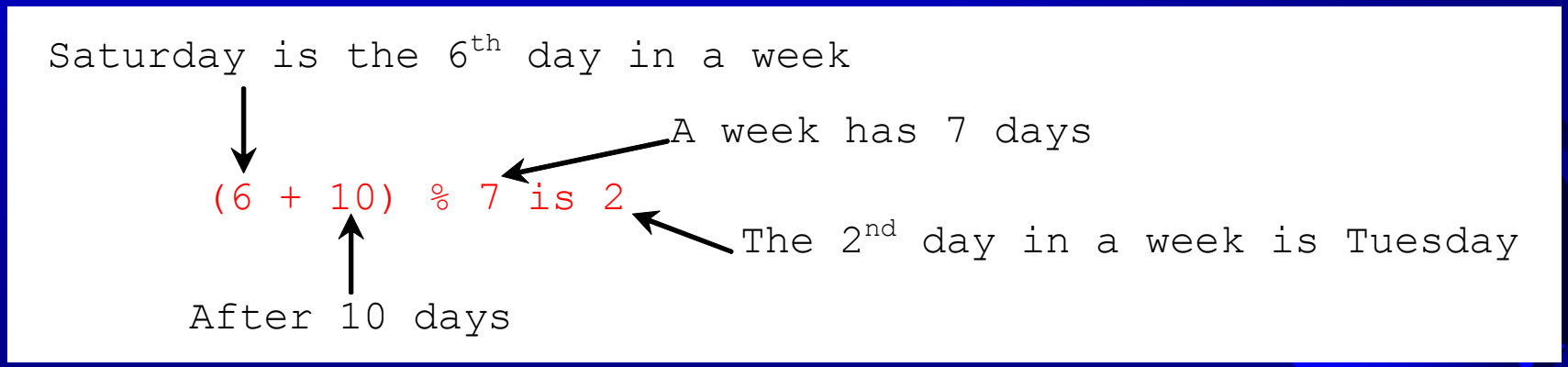
5.0 / 2 yields a double value 2.5

5 % 2 yields 1 (the remainder of the division)

Remainder Operator

For example, an even number % 2 is always 0 and an odd number % 2 is always 1. So you can use this property to determine whether a number is even or odd.

Suppose today is Saturday and you and your friends are going to meet in 10 days. What day is in 10 days? You can find that day is Tuesday using the following expression:



Problem: Displaying Time

Write a program that obtains hours and minutes from seconds.

```
import java.util.Scanner;

public class DisplayTime {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    // Prompt the user for input
    System.out.print("Enter an integer for seconds: ");
    int seconds = input.nextInt();

    int minutes = seconds / 60; // Find minutes in seconds
    int remainingSeconds = seconds % 60; // Seconds remaining
    System.out.println(seconds + " seconds is " + minutes +
      " minutes and " + remainingSeconds + " seconds");
  }
}
```

NOTE

Calculations involving floating-point numbers are approximated because these numbers are not stored with complete accuracy.

For example:

System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);

displays 0.5000000000000001, not 0.5

System.out.println(1.0 - 0.9);

displays 0.09999999999999998, not 0.1

Integers are stored precisely so calculations with integers yield a precise integer result.

Exponent Operations

```
System.out.println(Math.pow(2, 3));
// Displays 8.0
System.out.println(Math.pow(4, 0.5));
// Displays 2.0
System.out.println(Math.pow(2.5, 2));
// Displays 6.25
System.out.println(Math.pow(2.5, -2));
// Displays 0.16
```

Numeric 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.

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 of type int

has value 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.

Floating-Point Literals

Floating-point literals have a decimal point.

Are treated as a double type value.

For example, 5.0 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.

These are float: 100.2f or 100.2F

These are double: 100.2d or 100.2D

Scientific Notation

Floating-point literals can also be specified in scientific notation, for example,

123.456 = 1.23456e+2 & 1.23456e2

0.0123456 = 1.23456e-2.

E (or e) represents an exponent and it can be either in lowercase or uppercase.

Arithmetic Expressions

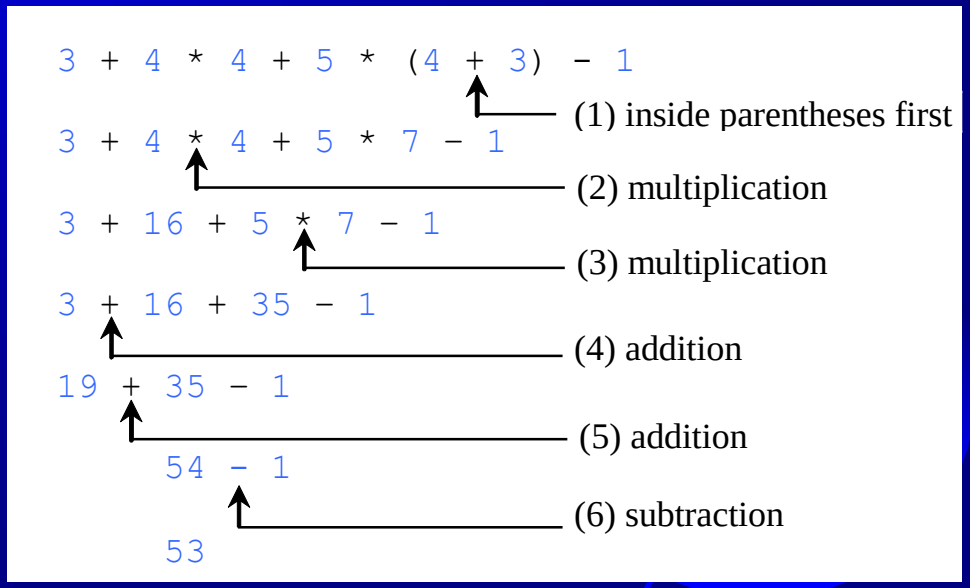
$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

is translated to

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

How to Evaluate an Expression

Though Java has its own way to evaluate an expression behind the scene, the result of a Java expression and its corresponding arithmetic expression are the same. Therefore, you can safely apply the arithmetic rule for evaluating a Java expression.



Problem: Converting Temperatures

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

celsius = (5.0 / 9) * (fahrenheit – 32)

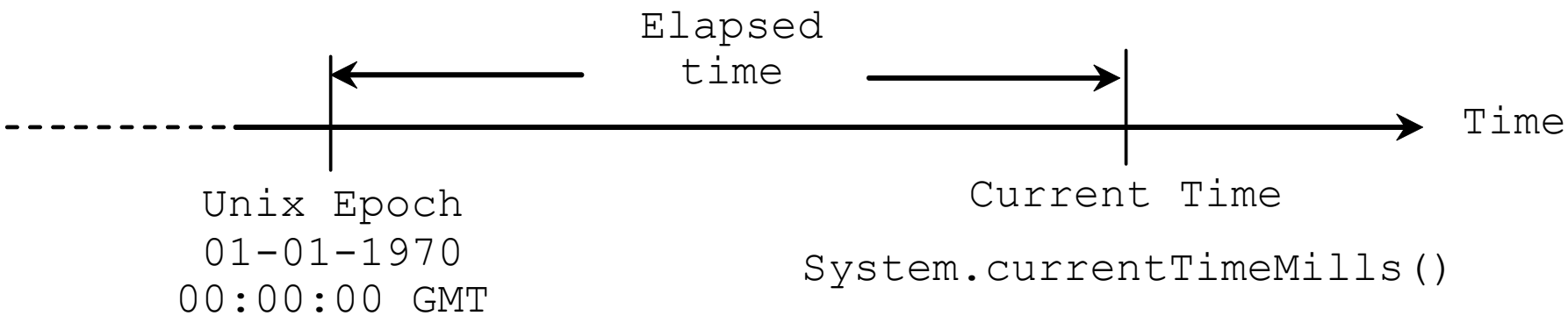
```
import java.util.Scanner;

public class FahrenheitToCelsius {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter a degree in Fahrenheit: ");
    double fahrenheit = input.nextDouble();

    // Convert Fahrenheit to Celsius
    double celsius = (5.0 / 9) * (fahrenheit - 32);
    System.out.println("Fahrenheit " + fahrenheit + " is " +  celsius + " in Celsius");
  }
}
```

Problem: Displaying Current Time

display current time in GMT in the format: hour:minute:second such as 1:45:19.

Use System.currentTimeMillis

Gives the time in milliseconds since the midnight, January 1, 1970 GMT.



```
public class ShowCurrentTime {
  public static void main(String[] args) {
    // Obtain the total milliseconds since midnight, Jan 1, 1970
    long totalMilliseconds = System.currentTimeMillis();

    // Obtain the total seconds since midnight, Jan 1, 1970
    long totalSeconds = totalMilliseconds / 1000;

    // Compute the current second in the minute in the hour
    long currentSecond = totalSeconds % 60;

    // Obtain the total minutes
    long totalMinutes = totalSeconds / 60;

    // Compute the current minute in the hour
    long currentMinute = totalMinutes % 60;

    // Obtain the total hours
    long totalHours = totalMinutes / 60;

    // Compute the current hour
    long currentHour = totalHours % 24;

    // Display results
    System.out.println("Current time is " + currentHour + ":"
      + currentMinute + ":" + currentSecond + " GMT");
  }
}
```

Shortcut Assignment Operators

(Pick one style and stick with it)

Operator	*Example*	*Equivalent*
`+=`	`i += 8`	`i = i + 8`
`-=`	`f -= 8.0`	`f = f - 8.0`
`*=`	`i *= 8`	`i = i * 8`
`/=`	`i /= 8`	`i = i / 8`
`%=`	`i %= 8`	`i = i % 8`

More shortcuts
Increment and Decrement Operators
(Pick one style and stick with it)

Operator	Name	Description
++var	preincrement	The expression (++var) increments var by 1 and evaluates to the *new* value in var *after* the increment.
var++	postincrement	The expression (var++) evaluates to the *original* value in var and increments var by 1.
--var	predecrement	The expression (--var) decrements var by 1 and evaluates to the *new* value in var *after* the decrement.
var--	postdecrement	The expression (var--) evaluates to the *original* value in var and decrements var by 1.

More shortcuts
Increment and Decrement Operators.

```
int i = 10;
int newNum = 10 * i++;
```

Same effect as

```
int newNum = 10 * i;
i = i + 1;
```

```
int i = 10;
int newNum = 10 * (++i);
```

Same effect as

```
i = i + 1;
int newNum = 10 * i;
```

More shortcuts
Increment and Decrement Operators

Using increment and decrement operators makes expressions short…

But… it also makes them **complex and difficult to read.**

Avoid using these operators in expressions that modify multiple variables, or the same variable for multiple times such as this: int k = ++i + i.

The main idea here is to AVOID them!

Assignment Expressions and Assignment Statements

Prior to Java 2, all the expressions can be used as statements. Since Java 2, only the following types of expressions can be statements:

variable op= expression; // Where op is +, -, *, /, or %

++variable;

variable++;

--variable;

variable--;