Chapter 2 Elementary Programming

Introducing Programming with an Example

- Listing 2.1 Computing the Area of a Circle

This program computes the area of the circle.

Outline (algorithm) :

1. Read in the circle's radius.
2. Compute the area using the following formula:

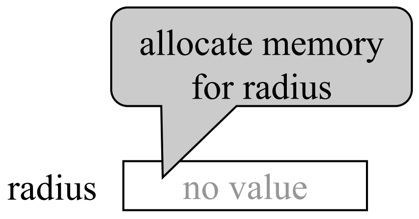
   $area = radius \times radius \times \pi$

3. Display the result

animation

Trace a Program Execution

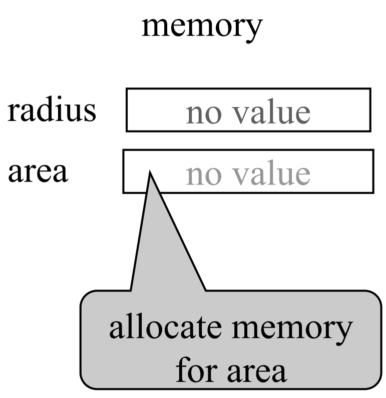
```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // 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

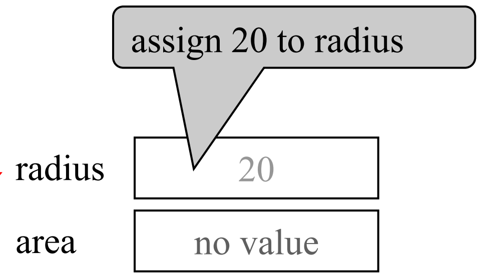
```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // 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

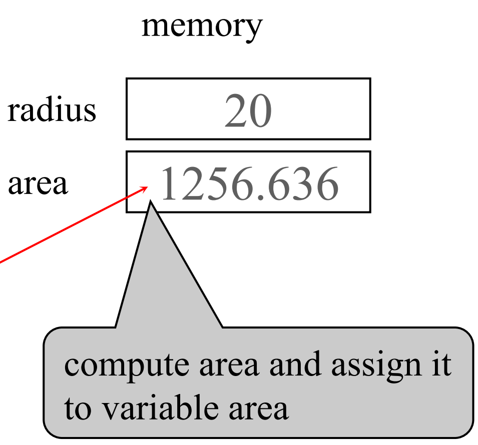
```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radius;
    double area;

    // 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);
  }
}
```



animation

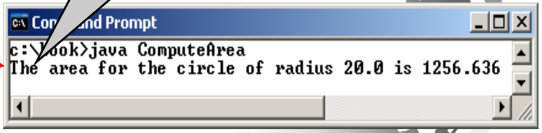
Trace a Program Execution

```
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



```
c:\book>java ComputeArea
The area for the circle of radius 20.0 is 1256.636
```

Identifiers

Identifiers are the names that identify the elements such as classes, methods, and variables in a program

An identifier is a sequence of characters that consists 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.

Variables

```
// Compute the first area
radius = 1.0;
area = radius * radius * 3.14159;
System.out.println("The area is “ + area +
  " for radius "+radius);

// Compute the second area
radius = 2.0;
area = radius * radius * 3.14159;
System.out.println("The area is “ + area +
  " for radius "+radius);
```

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;
```

Reading Input from the Console

1. Create a Scanner object

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

2. Use the method nextDouble() to obtain to a double value. For example,

```
System.out.print("Enter a double value: ");
Scanner input = new Scanner(System.in);
double d = input.nextDouble();
```

Reading Input from the Console

Listing 2.2 ComputeAreaWithConsoleInput.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 results
    System.out.println("The area for the circle of radius " +
radius + " is " + area);
  }
}
```

Reading Input from the Console

Listing 2.3 ComputeAverage.java

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

Implicit Import and Explicit Import

```
java.util.* ; // Implicit import

java.util.Scanner; // Explicit Import
```

No performance difference

Named Constants

```
final datatype CONSTANTNAME = VALUE;

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

Named Constants

Listing 2.4 ComputeAreaWithConsoleInput.java

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

public class ComputeAreaWithConsoleInput {
  public static void main(String[] args) {
    final double PI = 3.14159; // Declare a constant
    // 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 * PI;

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

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`.

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 in constants, and use underscores to connect words. For example, the constant `PI and` 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

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.

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 %

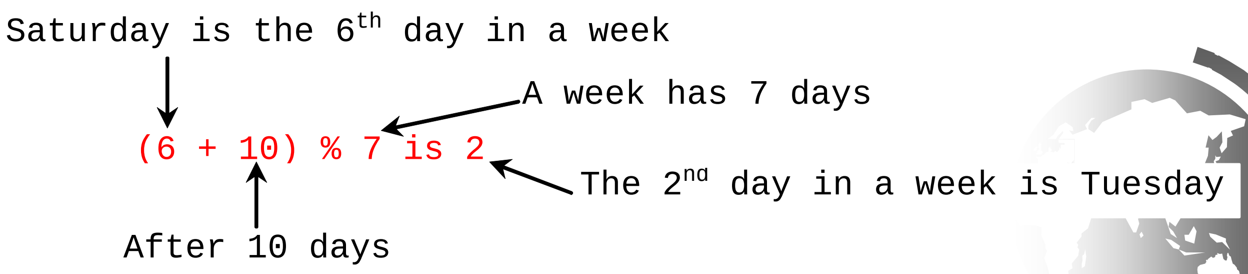
5 / 2 yields an integer 2.

5.0 / 2 yields a double value 2.5

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

Remainder Operator

Remainder is very useful in programming. 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:



Saturday is the 6th day in a week
A week has 7 days
(6 + 10) % 7 is 2
The 2nd day in a week is Tuesday
After 10 days

Problem: Displaying Time

Listing 2.5 DisplaTime.java

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

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, and

System.out.println(1.0 - 0.9);

displays 0.09999999999999998, not 0.1. Integers are stored precisely. Therefore, 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
```

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 $\texttt{1.0 / 3.0 is 0.}\underbrace{\texttt{3333333333333333}}_{\text{16 digits}}$

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

displays $\texttt{1.0F / 3.0F is 0.}\underbrace{\texttt{33333334}}_{\text{7 digits}}$

Scientific Notation

Floating-point literals can also be specified in scientific notation, for example, 1.23456e+2, same as 1.23456e2, is equivalent to 123.456, and 1.23456e-2 is equivalent to 0.0123456. E (or e) represents an exponent and it can be either in lowercase or uppercase.

Arithmetic Expressions

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

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.

```
3 + 4 * 4 + 5 * (4 + 3) - 1
                   ↑
                   └──── (1) inside parentheses first
3 + 4 * 4 + 5 * 7 – 1
      ↑
      └───────────────── (2) multiplication
3 + 16 + 5 * 7 – 1
           ↑
           └──────────── (3) multiplication
3 + 16 + 35 – 1
  ↑
  └───────────────────── (4) addition
19 + 35 – 1
   ↑
   └──────────────────── (5) addition
     54 - 1
        ↑
        └─────────────── (6) subtraction
     53
```

Problem: Converting Temperatures

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

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

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

Problem: Converting Temperatures

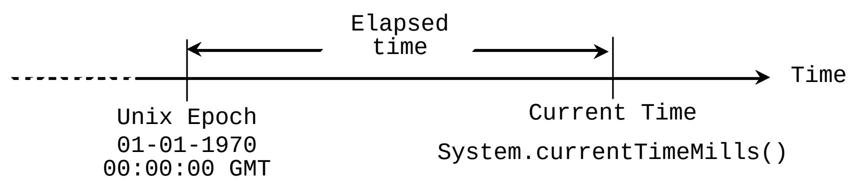
Listing 2.6 FahrenheitToCelsius.java

```
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

Write a program that displays current time in GMT in the format hour:minute:second such as 1:45:19.

The currentTimeMillis method in the System class returns the current time in milliseconds since the midnight, January 1, 1970 GMT. (1970 was the year when the Unix operating system was formally introduced.) You can use this method to obtain the current time, and then compute the current second, minute, and hour as follows.



ShowCurrentTime.java

Listing 2.7 ShowCurrentTime.java

```
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");
} }
```

Augmented Assignment Operators

Operator	*Name*	*Example*	*Equivalent*
`+=`	Addition assignment	`i += 8`	`i = i + 8`
`-=`	Subtraction assignment	`i -= 8`	`i = i - 8`
`*=`	Multiplication assignment	`i *= 8`	`i = i * 8`
`/=`	Division assignment	`i /=  8`	`i = i / 8`
`%=`	Remainder assignment	`i %= 8`	`i = i % 8`

Increment and Decrement Operators

Operator	*Name*	*Description*	*Example (assume i = 1)*
`++var`	preincrement	Increment `var` by `1`, and use the new `var` value in the statement	`int j = ++i;` `// j is 2, i is 2`
`var++`	postincrement	Increment `var` by `1`, but use the original `var` value in the statement	`int j = i++;` `// j is 1, i is 2`
`--var`	predecrement	Decrement `var` by `1`, and use the new `var` value in the statement	`int j = --i;` `// j is 0, i is 0`
`var--`	postdecrement	Decrement `var` by `1`, and use the original `var` value in the statement	`int j = i--;` `// j is 1, i is 0`

Increment and Decrement Operators, cont.

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

Same effect as $\longrightarrow$

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

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

Same effect as $\longrightarrow$

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

Numeric Type Conversion

Consider the following statements:

```
byte i = 100;
long k = i * 3 + 4;
double d = i * 3.1 + k / 2;
```

Conversion Rules

When performing a binary operation involving two operands of different types, Java automatically converts the operand based on the following rules:

1. If one of the operands is double, the other is converted into double.
2. Otherwise, if one of the operands is float, the other is converted into float.
3. Otherwise, if one of the operands is long, the other is converted into long.
4. Otherwise, both operands are converted into int.

Type Casting

Implicit casting

`double d = 3;` (type widening)

Explicit casting

`int i = (int)3.0;` (type narrowing)

`int i = (int)3.9;` (Fraction part is truncated)

What is wrong? int x = 5 / 2.0;

`range increases`

$\longrightarrow$

`byte, short, int, long, float, double`

Problem: Keeping Two Digits After Decimal Points

Write a program that displays the sales tax with two digits after the decimal point.

Problem: Keeping Two Digits After Decimal Points

Listing 2.8 SalesTax.java

```
import java.util.Scanner;

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

    System.out.print("Enter purchase amount: ");
    double purchaseAmount = input.nextDouble();

    double tax = purchaseAmount * 0.06;
    System.out.println("Sales tax is $" + (int)(tax * 100) / 100.0);
  }
}
```

Problem: Computing Loan Payments

This program lets the user enter the interest rate, number of years, and loan amount, and computes monthly payment and total payment.

$$monthlyPayment = \frac{loanAmount \times monthlyInterestRate}{1 - \frac{1}{(1 + monthlyInterestRate)^{numberOfYears \times 12}}}$$

$$totalPayment = monthlyPayment \times numberOfYears \times 12$$

Problem: Computing Loan Payments

Listing 2.9 ComputeLoan.java

```
import java.util.Scanner;

public class ComputeLoan {
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);
    // Enter annual interest rate in percentage, e.g., 7.25
    System.out.print("Enter annual interest rate in percentage, e.g., 7.25 : ");
    double annualInterestRate = input.nextDouble();
    // Obtain monthly interest rate
    double monthlyInterestRate = annualInterestRate / 1200;
    // Enter number of years
    System.out.print("Enter number of years as an integer, e.g., 5: ");
    int numberOfYears = input.nextInt();

(Cont...)
```

Problem: Computing Loan Payments

```
(Cont...)

    // Enter loan amount
    System.out.print("Enter loan amount, e.g., 120000.95: ");
    double loanAmount = input.nextDouble();
    // Calculate payment
    double monthlyPayment = loanAmount * monthlyInterestRate / (1 - 1 /
Math.pow(1+monthlyInterestRate, numberOfYears * 12));
    double totalPayment = monthlyPayment * numberOfYears * 12;
    // Display results
    System.out.println("The monthly payment is $" + (int)(monthlyPayment * 100) / 100.0);
   System.out.println("The total payment is $" + (int)(totalPayment * 100) / 100.0);
  }
}
```