

Chapter 2 Elementary Programming

Example: ComputeArea

```
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);  
    }  
}
```

radius

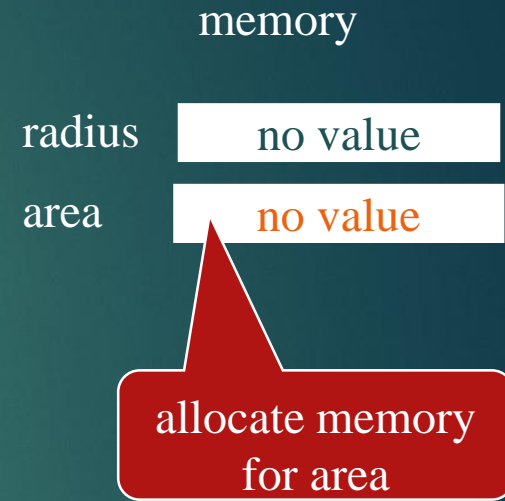
allocate memory
for radius

no value

Trace a Program Execution

3

```
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

4

```
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);  
    }  
}
```

radius

20

area

no value

assign 20 to radius

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

compute area and assign it to variable area

Trace a Program Execution

6

```
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

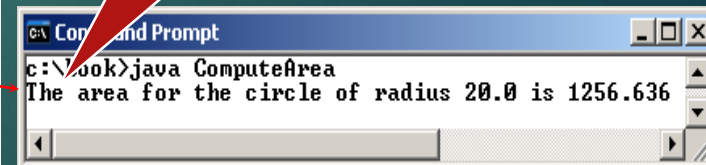
radius

20

area

1256.636

print a message to the
console



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

Reading Input from the Console₇

1. Create a Scanner object

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

2. Use the methods nextInt(), nextFloat(), nextDouble(), to obtain to an int, float, or double. For example,

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

Identifiers

8

- ▶ 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.

Variables

9

```
// 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

10

```
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

11

```
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

12

▶ `int x = 1;`

▶ `double d = 1.4;`

Constants

13

```
final datatype CONSTANTNAME = VALUE;
```

```
final double PI = 3.14159;
```

```
final int SIZE = 3;
```

Numerical Data Types

14

Name	Range	Storage Size
byte	-2^7 (-128) to 2^7-1 (127)	8-bit signed
short	-2^{15} (-32768) to $2^{15}-1$ (32767)	16-bit signed
int	-2^{31} (-2147483648) to $2^{31}-1$ (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

15

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

16

+, -, *, /, and %

$5 / 2$ yields an integer 2.

$5.0 / 2$ yields a double value 2.5

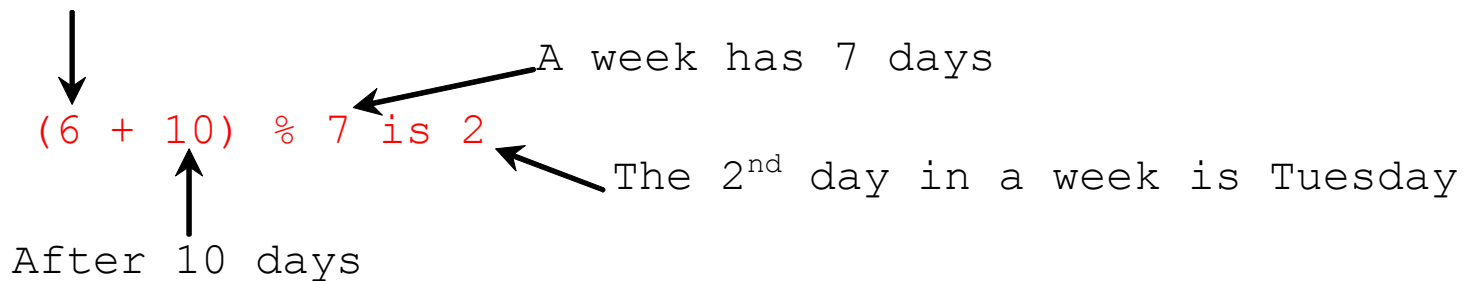
$5 \% 2$ yields 1 (the remainder of the division)

Remainder Operator

17

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



Problem: Displaying Time

18

Write a program that obtains hours and minutes from seconds.

DisplayTime.java

```
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

20

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 0.50000000000000000001, not 0.5, and

```
System.out.println(1.0 - 0.9);
```

displays 0.099999999999999999998, not 0.1. Integers are stored precisely. Therefore, calculations with integers yield a precise integer result.

Number Literals

21

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

22

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

23

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.

Scientific Notation

24

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

25

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

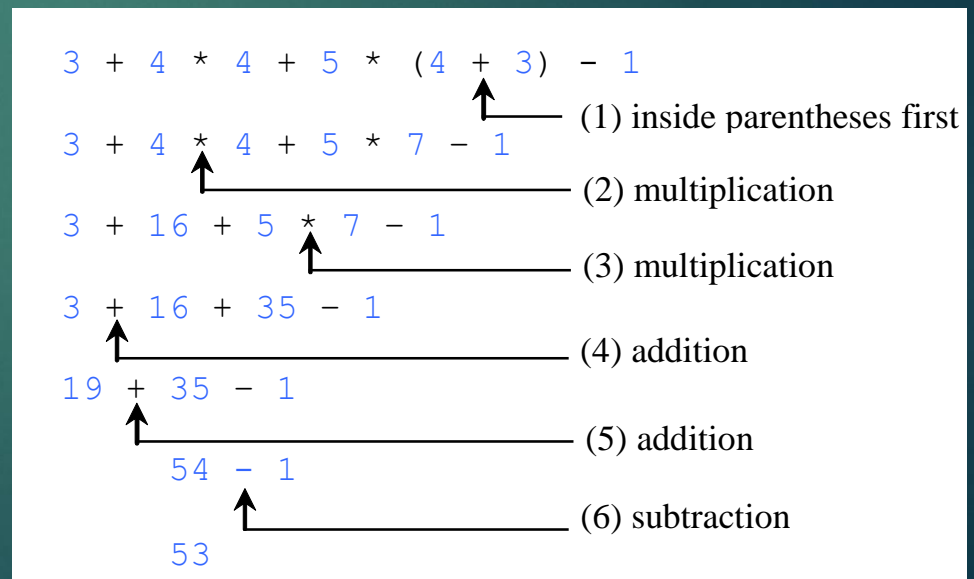
is translated to

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

How to Evaluate an Expression

26

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

27

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

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

```
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");
    }
}
```

Shortcut Assignment Operators

29

<i>Operator</i>	<i>Example</i>	<i>Equivalent</i>
<code>+=</code>	<code>i += 8</code>	<code>i = i + 8</code>
<code>-=</code>	<code>f -= 8.0</code>	<code>f = f - 8.0</code>
<code>*=</code>	<code>i *= 8</code>	<code>i = i * 8</code>
<code>/=</code>	<code>i /= 8</code>	<code>i = i / 8</code>
<code>%=</code>	<code>i %= 8</code>	<code>i = i % 8</code>

Increment and Decrement Operators

30

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

Increment and Decrement Operators, cont.

31

```
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;
```

Numeric Type Conversion

32

Consider the following statements:

```
byte i = 100;
```

```
long k = i * 3 + 4;
```

```
double d = i * 3.1 + k / 2;
```


Conversion Rules

33

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

34

Implicit casting

```
double d = 3; (type widening)
```

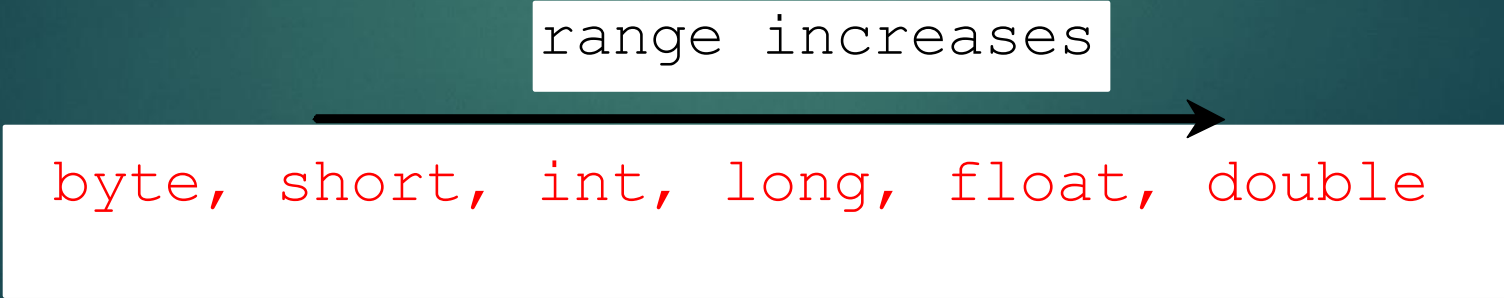
Explicit casting

```
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



byte, short, int, long, float, double

Problem: Keeping Two Digits 35 After Decimal Points

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

```
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 Payments³⁷

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

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

```
import java.util.Scanner;
public class ComputeLoan {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Enter yearly interest rate
        System.out.print("Enter yearly interest rate, for example 8.25: ");
        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, for example 5: ");
        int numberOfYears = input.nextInt(); // Enter loan amount
        System.out.print("Enter loan amount, for example 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);
    }
}
```

Character Data Type

39

```
char letter = 'A'; (ASCII)
char numChar = '4'; (ASCII)
char letter = '\u0041'; (Unicode)
char numChar = '\u0034'; (Unicode)
```

Four hexadecimal digits.

NOTE: The increment and decrement operators can also be used on char variables to get the next or preceding Unicode character. For example, the following statements display character b.

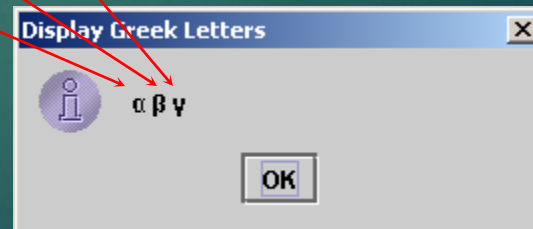
```
char ch = 'a';
System.out.println(++ch);
```


Unicode Format

40

Java characters use *Unicode*, a 16-bit encoding scheme established by the Unicode Consortium to support the interchange, processing, and display of written texts in the world's diverse languages. Unicode takes two bytes, preceded by `\u`, expressed in four hexadecimal numbers that run from `\u0000` to `\uFFFF`. So, Unicode can represent $65535 + 1$ characters.

Unicode `\u03b1` `\u03b2` `\u03b3` for three Greek letters



Escape Sequences for Special Characters 41

<i>Description</i>	<i>Escape Sequence</i>	<i>Unicode</i>
Backspace	<code>\b</code>	<code>\u0008</code>
Tab	<code>\t</code>	<code>\u0009</code>
Linefeed	<code>\n</code>	<code>\u000A</code>
Carriage return	<code>\r</code>	<code>\u000D</code>
Backslash	<code>\\</code>	<code>\u005C</code>
Single Quote	<code>\'</code>	<code>\u0027</code>
Double Quote	<code>\"</code>	<code>\u0022</code>

Casting between char and Numeric Types

42

```
int i = 'a'; // Same as int i = (int)'a';
```

```
char c = 97; // Same as char c = (char)97;
```

Problem: Monetary Units

43

This program lets the user enter the amount in decimal representing dollars and cents and output a report listing the monetary equivalent in single dollars, quarters, dimes, nickels, and pennies.

Your program should report maximum number of dollars, then the maximum number of quarters, and so on, in this order.

Trace ComputeChange

44

Suppose amount is 11.56

```
int remainingAmount = (int)(amount * 100);
```

```
// Find the number of one dollars
```

```
int numberOfOneDollars = remainingAmount / 100;  
remainingAmount = remainingAmount % 100;
```

```
// Find the number of quarters in the remaining amount
```

```
int numberOfQuarters = remainingAmount / 25;  
remainingAmount = remainingAmount % 25;
```

```
// Find the number of dimes in the remaining amount
```

```
int numberOfDimes = remainingAmount / 10;  
remainingAmount = remainingAmount % 10;
```

```
// Find the number of nickels in the remaining amount
```

```
int numberOfNickels = remainingAmount / 5;  
remainingAmount = remainingAmount % 5;
```

```
// Find the number of pennies in the remaining amount
```

```
int numberOfPennies = remainingAmount;
```

remainingAmount

1156

remainingAmount
initialized

Trace ComputeChange

Suppose amount is 11.56

```
int remainingAmount = (int)(amount * 100);  
  
// Find the number of one dollars  
int numberOfOneDollars = remainingAmount / 100;  
remainingAmount = remainingAmount % 100;  
  
// Find the number of quarters in the remaining amount  
int numberOfQuarters = remainingAmount / 25;  
remainingAmount = remainingAmount % 25;  
  
// Find the number of dimes in the remaining amount  
int numberOfDimes = remainingAmount / 10;  
remainingAmount = remainingAmount % 10;  
  
// Find the number of nickels in the remaining amount  
int numberOfNickels = remainingAmount / 5;  
remainingAmount = remainingAmount % 5;  
  
// Find the number of pennies in the remaining amount  
int numberOfPennies = remainingAmount;
```

remainingAmount

1156

numberOfOneDollars

11

numberOfOneDollars
assigned

45

Trace ComputeChange

46

Suppose amount is 11.56

```
int remainingAmount = (int)(amount * 100);
```

```
// Find the number of one dollars
```

```
int numberOfOneDollars = remainingAmount / 100;
```

```
remainingAmount = remainingAmount % 100;
```

```
// Find the number of quarters in the remaining amount
```

```
int numberOfQuarters = remainingAmount / 25;
```

```
remainingAmount = remainingAmount % 25;
```

```
// Find the number of dimes in the remaining amount
```

```
int numberOfDimes = remainingAmount / 10;
```

```
remainingAmount = remainingAmount % 10;
```

```
// Find the number of nickels in the remaining amount
```

```
int numberOfNickels = remainingAmount / 5;
```

```
remainingAmount = remainingAmount % 5;
```

```
// Find the number of pennies in the remaining amount
```

```
int numberOfPennies = remainingAmount;
```

remainingAmount

56

numberOfOneDollars

11

remainingAmount
updated

Trace ComputeChange

47

Suppose amount is 11.56

```
int remainingAmount = (int)(amount * 100);
```

```
// Find the number of one dollars
```

```
int numberOfOneDollars = remainingAmount / 100;
```

```
remainingAmount = remainingAmount % 100;
```

```
// Find the number of quarters in the remaining amount
```

```
int numberOfQuarters = remainingAmount / 25;
```

```
remainingAmount = remainingAmount % 25;
```

```
// Find the number of dimes in the remaining amount
```

```
int numberOfDimes = remainingAmount / 10;
```

```
remainingAmount = remainingAmount % 10;
```

```
// Find the number of nickels in the remaining amount
```

```
int numberOfNickels = remainingAmount / 5;
```

```
remainingAmount = remainingAmount % 5;
```

```
// Find the number of pennies in the remaining amount
```

```
int numberOfPennies = remainingAmount;
```

remainingAmount

56

numberOfOneDollars

11

numberOfOneQuarters

2

numberOfOneQuarters
assigned

Trace ComputeChange

48

Suppose amount is 11.56

```
int remainingAmount = (int)(amount * 100);
```

```
// Find the number of one dollars
```

```
int numberOfOneDollars = remainingAmount / 100;  
remainingAmount = remainingAmount % 100;
```

```
// Find the number of quarters in the remaining amount
```

```
int numberOfQuarters = remainingAmount / 25;
```

```
remainingAmount = remainingAmount % 25;
```

```
// Find the number of dimes in the remaining amount
```

```
int numberOfDimes = remainingAmount / 10;  
remainingAmount = remainingAmount % 10;
```

```
// Find the number of nickels in the remaining amount
```

```
int numberOfNickels = remainingAmount / 5;  
remainingAmount = remainingAmount % 5;
```

```
// Find the number of pennies in the remaining amount
```

```
int numberOfPennies = remainingAmount;
```

remainingAmount

6

numberOfOneDollars

11

numberOfQuarters

2

remainingAmount
updated

The String Type

49

The char type only represents one character. To represent a string of characters, use the data type called String. For example,

```
String message = "Welcome to Java";
```

String Concatenation

50

// Three strings are concatenated

String message = "Welcome " + "to " + "Java";

// String Chapter is concatenated with number 2

String s = "Chapter" + 2; // s becomes Chapter2

// String Supplement is concatenated with
character B

String s1 = "Supplement" + 'B'; // s1 becomes
SupplementB

Programming Style and Documentation

51

- ▶ Appropriate Comments
- ▶ Naming Conventions
- ▶ Proper Indentation and Spacing Lines
- ▶ Block Styles

Appropriate Comments

52

Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instructor, date, and a brief description at the beginning of the program.

Naming Conventions

53

- ▶ 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.

54

▶ 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`

Proper Indentation and Spacing

55

- ▶ Indentation
 - ▶ Indent two spaces.
- ▶ Spacing
 - ▶ Use blank line to separate segments of the code.

Block Styles

56

Use end-of-line style for braces.

*Next-line
style*

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Styles");
    }
}
```

*End-of-line
style*

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Block Styles");
    }
}
```


Programming Errors

57

- ▶ Syntax Errors
 - ▶ Detected by the compiler
- ▶ Runtime Errors
 - ▶ Causes the program to abort
- ▶ Logic Errors
 - ▶ Produces incorrect result

Syntax Errors

58

```
public class ShowSyntaxErrors {  
    public static void main(String[] args) {  
        i = 30;  
        System.out.println(i + 4);  
    }  
}
```

Runtime Errors

59

```
public class ShowRuntimeErrors {  
    public static void main(String[] args) {  
        int i = 1 / 0;  
    }  
}
```

Logic Errors

60

```
public class ShowLogicErrors {
    // Determine if a number is between 1 and 100 inclusively
    public static void main(String[] args) {
        // Prompt the user to enter a number
        String input = JOptionPane.showInputDialog(null,
            "Please enter an integer:",
            "ShowLogicErrors", JOptionPane.QUESTION_MESSAGE);
        int number = Integer.parseInt(input);

        // Display the result
        System.out.println("The number is between 1 and 100, " +
            "inclusively? " + ((1 < number) && (number < 100)));

        System.exit(0);
    }
}
```

JOptionPane Input

61

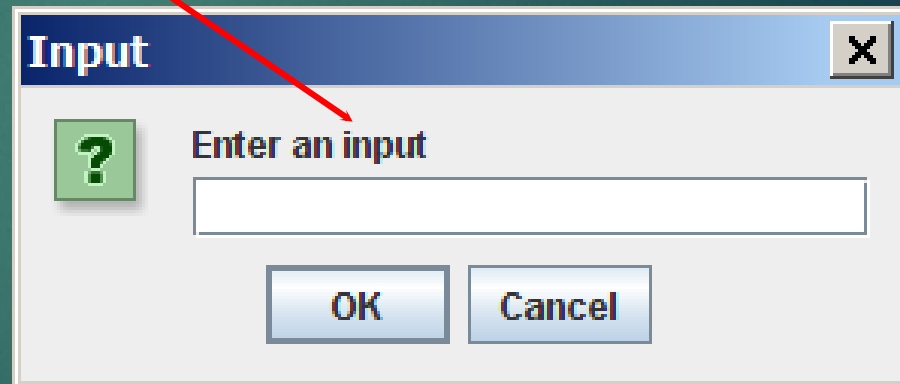
This book provides two ways of obtaining input.

1. Using the Scanner class (console input)
2. Using JOptionPane input dialogs

Getting Input from Input Dialog Boxes

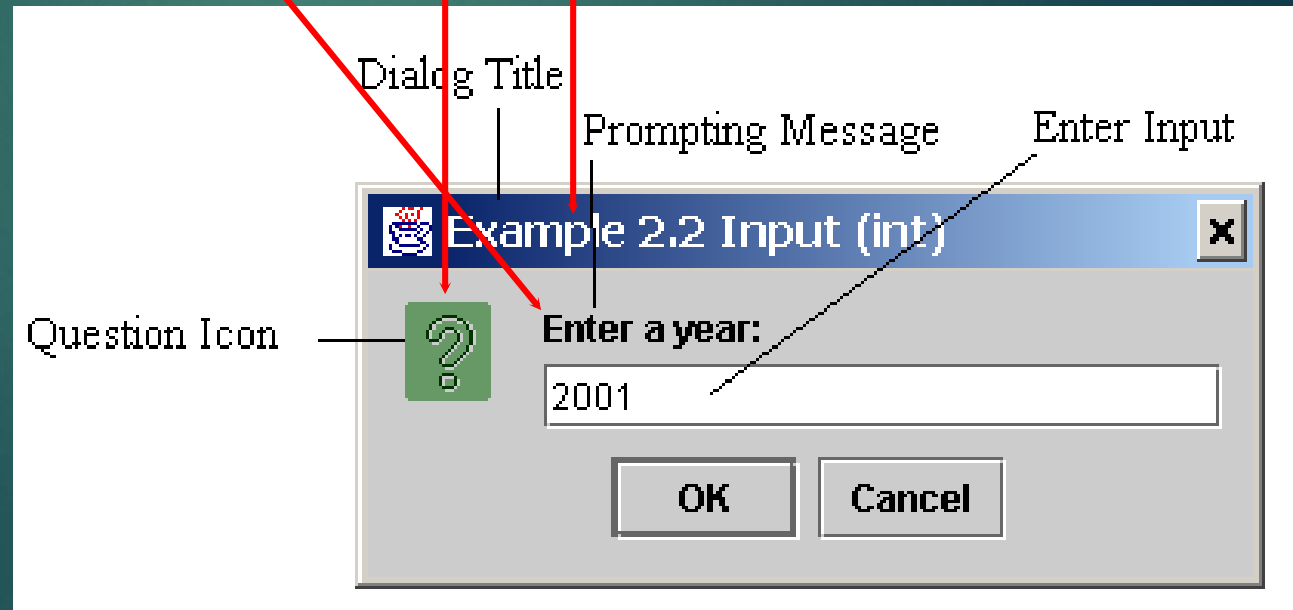
62

```
String input = JOptionPane.showInputDialog(  
    "Enter an input");
```



Getting Input from Input Dialog Boxes ⁶³

```
String string = JOptionPane.showInputDialog(  
    null, "Prompting Message", "Dialog Title",  
    JOptionPane.QUESTION_MESSAGE);
```



Two Ways to Invoke the Method 64

There are several ways to use the `showInputDialog` method. For the time being, you only need to know two ways to invoke it.

One is to use a statement as shown in the example:

```
String string = JOptionPane.showInputDialog(null, x,  
y, JOptionPane.QUESTION_MESSAGE);
```

where `x` is a string for the prompting message, and `y` is a string for the title of the input dialog box.

The other is to use a statement like this:

```
JOptionPane.showInputDialog(x);
```

where `x` is a string for the prompting message.

Converting Strings to Integers

65

The input returned from the input dialog box is a string. If you enter a numeric value such as 123, it returns "123". To obtain the input as a number, you have to convert a string into a number.

To convert a string into an int value, you can use the static parseInt method in the Integer class as follows:

```
int intValue = Integer.parseInt(intString);
```

where intString is a numeric string such as "123".

Converting Strings to Doubles

66

To convert a string into a double value, you can use the static parseDouble method in the Double class as follows:

```
double doubleValue  
=Double.parseDouble(doubleString);
```

where doubleString is a numeric string such as "123.45".

Problem: Computing Loan Payments Using Input Dialogs

67

Same as the preceding program for computing loan payments, except that the input is entered from the input dialogs and the output is displayed in an output dialog.

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