Chapter 3

Numerical Data

OBJECTIVES
After you have read and studied this chapter, you should be able to

- Select proper types for numerical data.
- Write arithmetic expressions in Java.
- Evaluate arithmetic expressions using the precedence rules.
- Describe how the memory allocation works for objects and primitive data values.
- Write mathematical expressions using methods in the Math class.
- Write programs that input and output data using the InputBox and OutputBox classes from the javabook package.
- Apply the incremental development technique in writing programs.
- (Optional) Describe how the integers and real numbers are represented in memory.

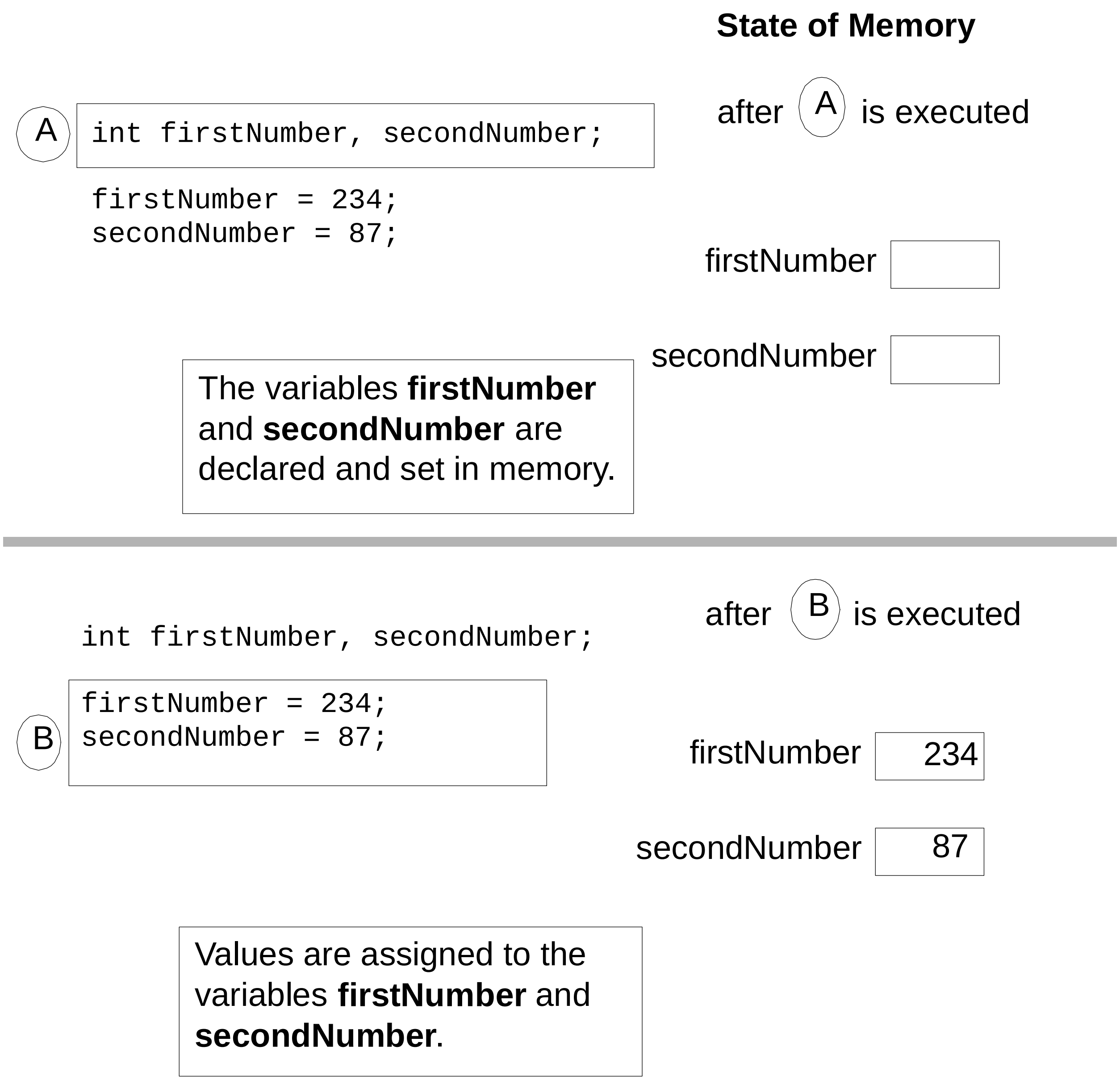
McGraw-Hill Publishing

TABLE 3.1 Java numerical data types and their precisions.

Data Type	Content	Default Value	Minimum Value	Maximum Value
`byte`	Integer	0	–128	127
`short`	Integer	0	–32768	32767
`int`	Integer	0	–2147483648	2147483647
`long`	Integer	0	–9223372036854775808	9223372036854775807
`float`	Real	0.0	–3.40282347E+38[a]	3.40282347E+38
`double`	Real	0.0	–1.79769313486231570E+308	1.79769313486231570E+308

[a] The character E indicates a number is expressed in scientific notation.

FIGURE 3.1 A diagram showing how two memory locations (variables) with names **firstNumber** and **secondNumber** are declared, and values are assigned to them.



State of Memory

```
int firstNumber, secondNumber;
firstNumber = 234;
secondNumber = 87;
```

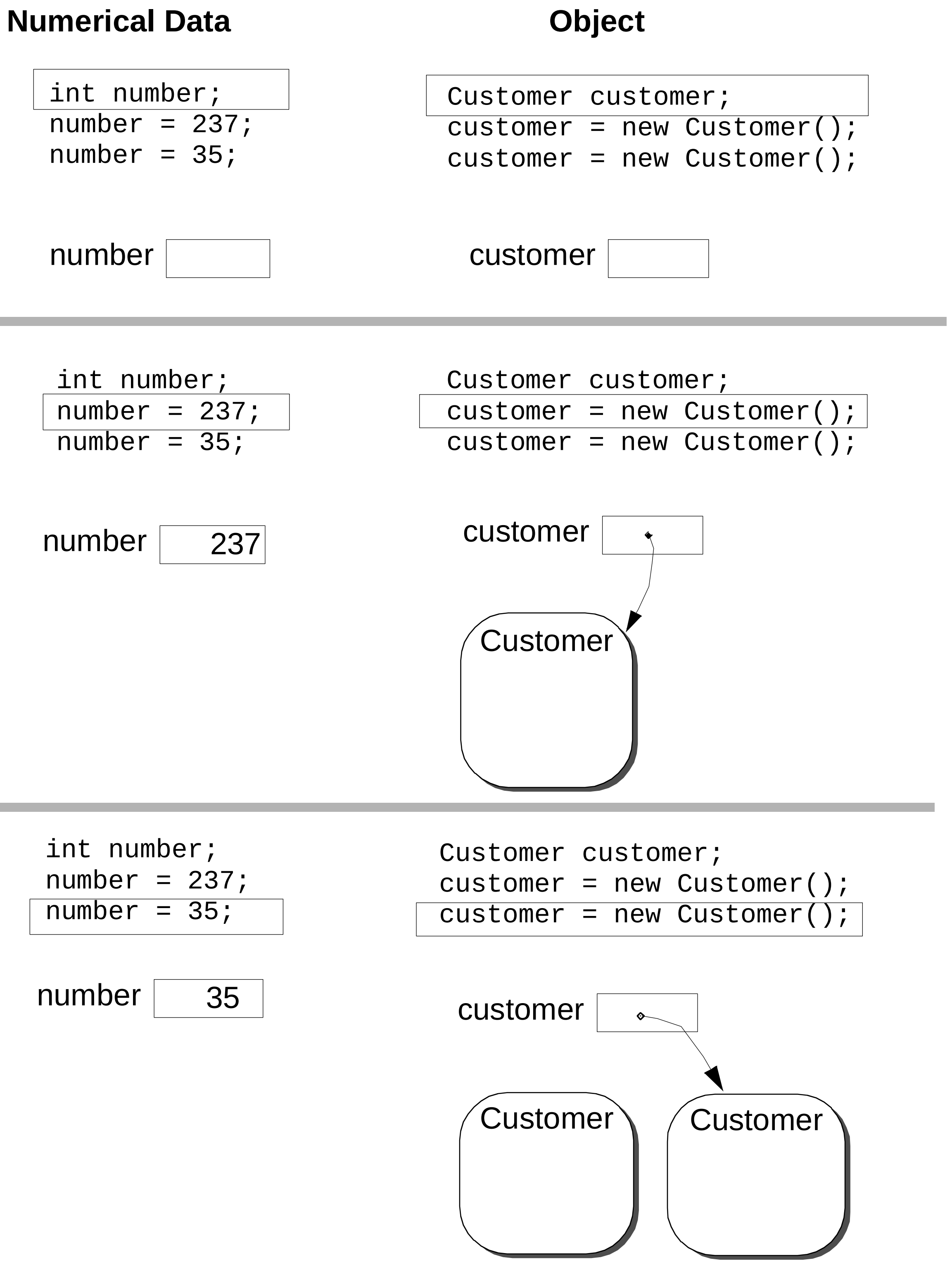
The variables **firstNumber** and **secondNumber** are declared and set in memory.

```
int firstNumber, secondNumber;
firstNumber = 234;
secondNumber = 87;
```

Values are assigned to the variables **firstNumber** and **secondNumber**.

McGraw-Hill Publishing

FIGURE 3.2 A difference between object declaration and numerical data declaration.



Numerical Data

Object

```
int number;
number = 237;
number = 35;
```

```
Customer customer;
customer = new Customer();
customer = new Customer();
```

```
int number;
number = 237;
number = 35;
```

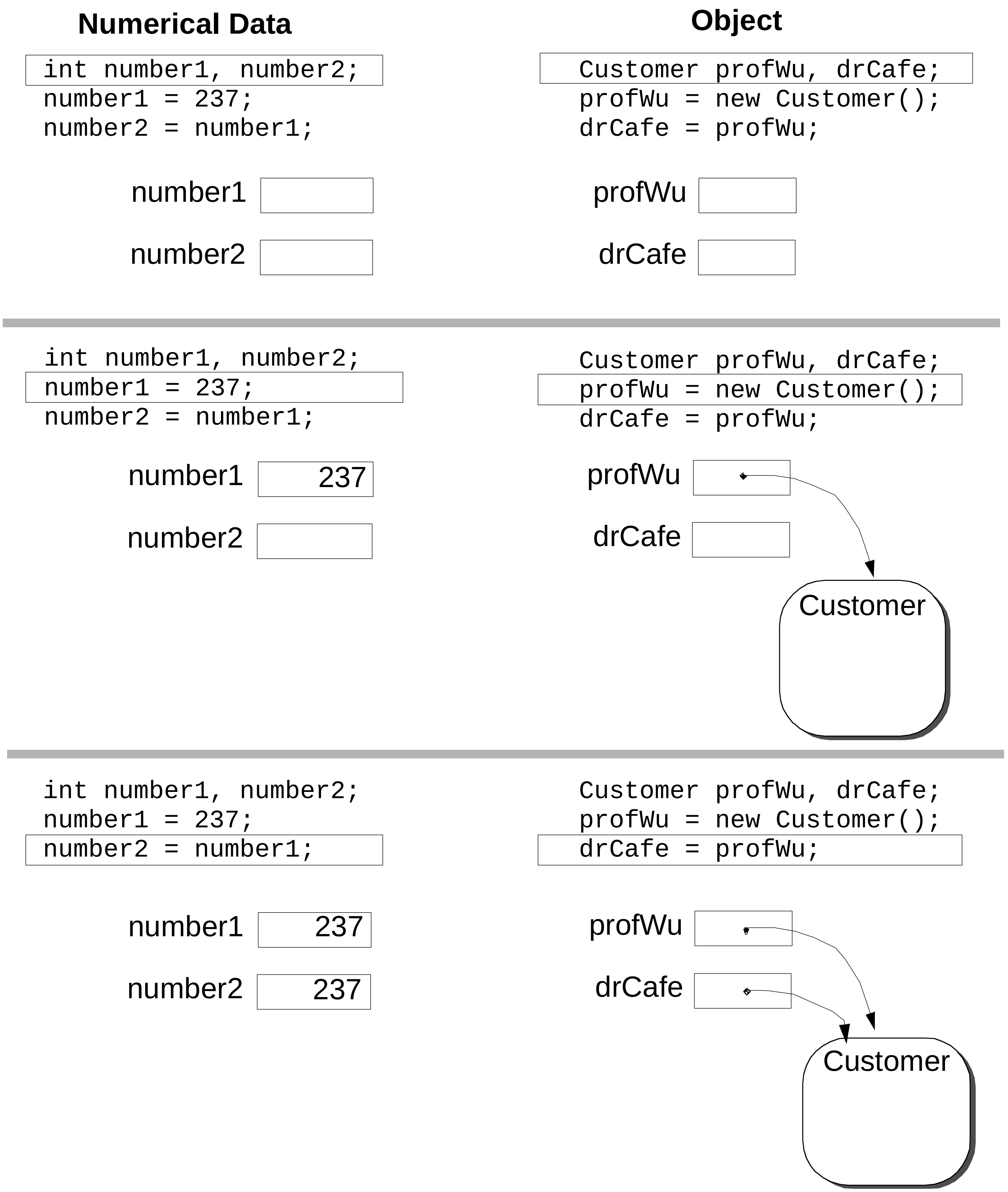
```
Customer customer;
customer = new Customer();
customer = new Customer();
```

```
int number;
number = 237;
number = 35;
```

```
Customer customer;
customer = new Customer();
customer = new Customer();
```

■
McGraw-Hill Publishing

FIGURE 3.3 An effect of assigning the content of one variable to another.



Numerical Data

```
int number1, number2;
number1 = 237;
number2 = number1;
```

Object

```
Customer profWu, drCafe;
profWu = new Customer();
drCafe = profWu;
```

TABLE 3.2 Arithmetic operators.

Operation	Java Operator	Example	Value (x=10, y=7, z =2.5)
Addition	+	x + y	17
Subtraction	-	x - y	3
Multiplication	*	x * y	70
Division	/	x / y	1
		x / z	4.0
Modulo division (remainder)	%	x % y	3

TABLE 3.3 Precedence rules for arithmetic operators and parentheses.

Order	Group	Operator	Rule
High	subexpression	( )	Subexpressions are evaluated first. If parentheses are nested, the innermost subexpression is evaluated first. If two or more pairs of parentheses are on the same level, then they are evaluated from left to right.
↕	unary operator	-, +	Unary minuses and pluses are evaluated second.
	multiplicative operator	*, /, %	Multiplicative operators are evaluated third. If two or more multiplicative operators are in an expression, then they are evaluated from left to right.
Low	additive operator	+, -	Additive operators are evaluated last. If two or more additive operators are in an expression, then they are evaluated from left to right.

TABLE 3.4 Rules for arithmetic promotion.

Operator Type	Promotion Rule
`Unary`	1. If the operand is of type byte or short, then it is converted to int. 2. Otherwise, the operand remains the same type.
`Binary`	1. If either operand is of type double, then the other operand is converted to double. 2. Otherwise, if either operand is of type float, then the other operand is converted to float. 3. Otherwise, if either operand is of type long, then the other operand is converted to long. 4. Otherwise, both operands are converted to int.

TABLE 3.5 **Math** class methods for commonly used mathematical functions.

Class Method	Argument Type	Result Type	Description	Example
`abs( a )`	`int`	`int`	Returns the absolute int value of **a.**	abs(10) › 10 abs(-5) › 5
	`long`	`long`	Returns the absolute long value of **a.**	
	`float`	`float`	Returns the absolute float value of **a.**	
`acos( a )`[a]	`double`	`double`	Returns the arc cosine of **a.**	acos(-1) › 3.14159
`asin( a )`†	`double`	`double`	Returns the arc sine of **a.**	asin(1) › 1.57079
`atan( a )`†	`double`	`double`	Returns the arc tangent of **a**.	atan(1) › 0.785398
`ceil( a )`	`double`	`double`	Returns the smallest whole number greater than or equal to **a**.	ceil(5.6) › 6.0 ceil(5.0) › 5.0 ceil(-5.6) › -5.0

TABLE 3.5 **Math** class methods for commonly used mathematical functions. (Continued)

Class Method	Argument Type	Result Type	Description	Example
`cos( a )`†	`double`	`double`	Returns the trigonometric cosine of **a**.	cos(π/2) ›0.0
`exp( a )`	`double`	`double`	Returns the natural number e (2.718...) raised to the power of **a**.	exp(2) › 7.389056099
`floor( a )`	`double`	`double`	Returns the largest whole number less than or equal to **a**.	floor(5.6) › 5.0 floor(5.0) › 5.0 floor(-5.6)› -6.0
`log( a )`	`double`	`double`	Returns the natural logarithm (base e) of **a**.	log(100) › 2.0
`max( a, b )`	`int`	`int`	Returns the larger of **a** and **b**.	max(10, 20) › 20
	`long`	`long`	Same as above.	
	`float`	`float`	Same as above.	
`min( a, b )`	`int`	`int`	Returns the smaller of **a** and **b**.	min(10, 20) › 10
	`long`	`long`	Same as above.	
	`float`	`float`	Same as above.	
`pow( a, b )`	`double`	`double`	Returns the number **a** raised to the power of **b**.	pow(2.0, 3.0) › 8.0
`random( )`	`<no argument>`	`double`	Generates a random number greater than or equal to 0.0 and less than 1.0	Examples given in Chapter 6.
`round( a )`	`float`	`int`	Returns the int value of **a** rounded to the nearest whole number.	round(5.6) › 6 round(5.4) › 5 round(–5.6) › –6
	`double`	`long`	Returns the float value of **a** rounded to the nearest whole number.	
`sin( a )`†	`double`	`double`	Returns the trigonometric sine of **a**.	sin(π/2) › 1.0

TABLE 3.5 **Math** class methods for commonly used mathematical functions. (Continued)

Class Method	Argument Type	Result Type	Description	Example
`sqrt( a )`	`double`	`double`	Returns the square root of **a**.	sqrt(9.0) › 3.0
`tan( a )`[†]	`double`	`double`	Returns the trigonometric tangent of **a**.	tan(π/4) › 1.0

a.All trigonometric functions are computed in radians.

FIGURE 3.4 The **InputBox** dialog after its method **getInteger** is executed.

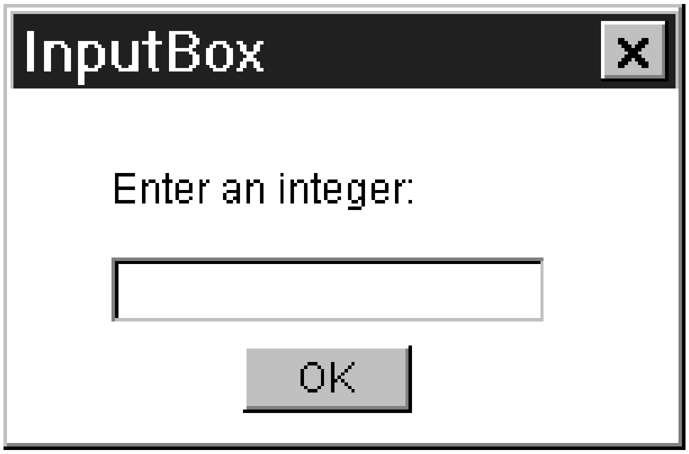
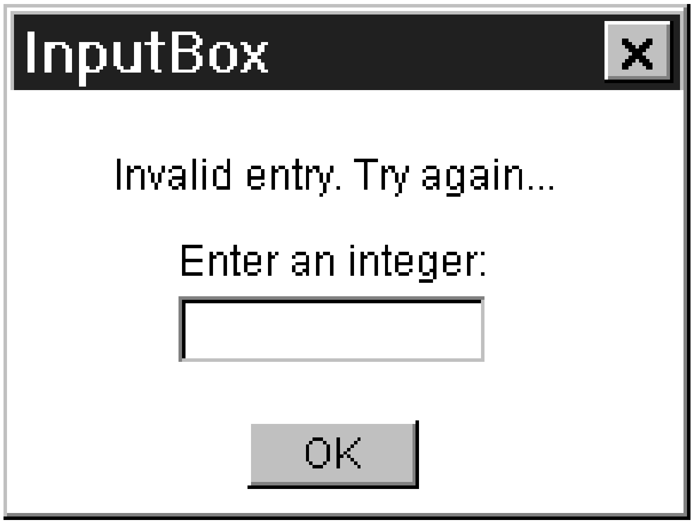


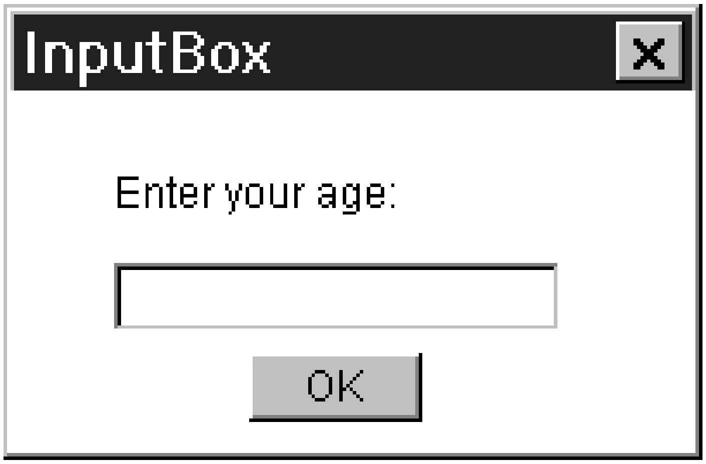
FIGURE 3.5 The **InputBox** dialog after a noninteger value is entered by the user.



Invalid entry. Try again...

Enter an integer:

FIGURE 3.6 An **InputBox** object with a programmer-specified prompt.



Enter your age:

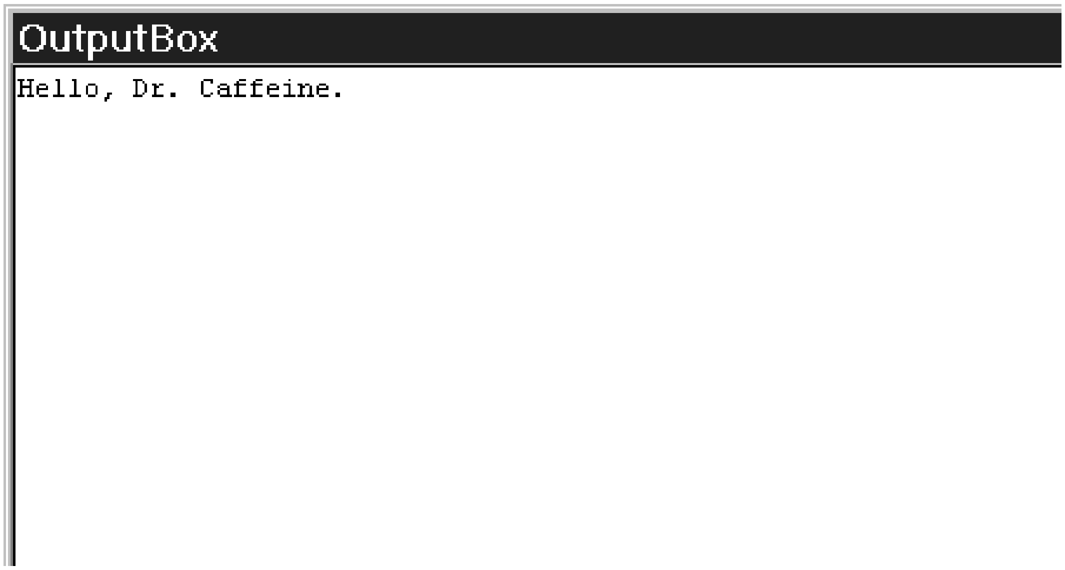
TABLE 3.6 A partial list of **InputBox** methods.

CLASS:	**InputBox**	
Method	**Argument**	**Description**

`getFloat`	`<none>`
	`or`
	`text`
		Allows the user to enter a real number, a number with or without a decimal point. The InputBox dialog object will not close until the user enters a valid real number. If there is no argument, then the default prompt Enter a Float is displayed in the dialog. If a text value is passed as the argument, then it is used as a prompt in the dialog.

`getInteger`	`<none>`
	`or`
	`text`
		Allows the user to enter an integer, a number without a decimal point. The InputBox dialog object will not close until the user enters a valid integer. If there is no argument, then the default prompt Enter an Integer is displayed in the dialog. If a text value is passed as the argument, then it is used as a prompt in the dialog.

FIGURE 3.7 Result of executing **outputBox.print("Hello, Dr. Caffeine.")**.

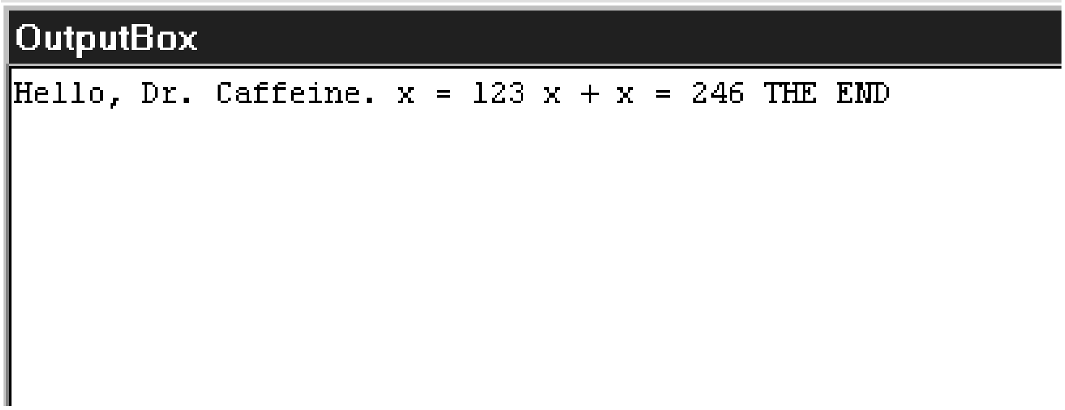


OutputBox

```
Hello, Dr. Caffeine.
```

FIGURE 3.8 Result of sending five **print** messages to **outputBox** of Figure 3.7.

```
    int x, y;
    x = 123;
    y = x + x;
    outputBox.print(" x = ");
    outputBox.print( x );
    outputBox.print(" x + x = ");
    outputBox.print( y );
    outputBox.print(" THE END");
```

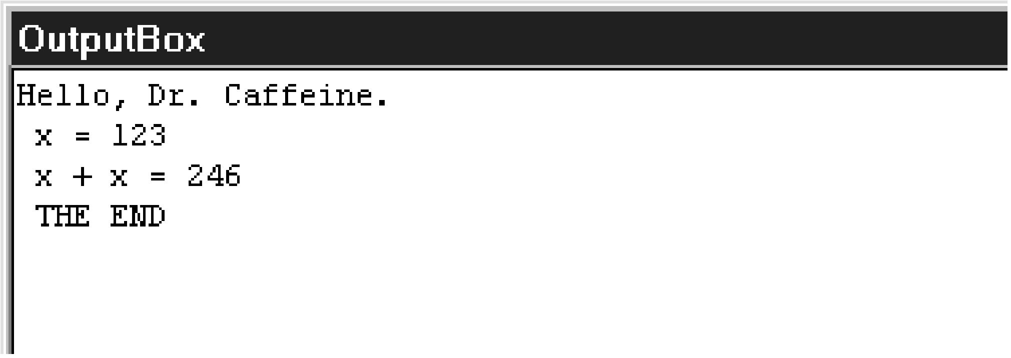


OutputBox

```
Hello, Dr. Caffeine. x = 123 x + x = 246 THE END
```

FIGURE 3.9 Result of sending four **printLine** messages to **outputBox**.

```
int x, y;
x = 123;
y = x + x;
outputBox.printLine("Hello, Dr. Caffeine.");
outputBox.print(" x = ");
outputBox.printLine( x );
outputBox.print(" x + x = ");
outputBox.printLine( y );
outputBox.printLine(" THE END");
```



```
Hello, Dr. Caffeine.
 x = 123
 x + x = 246
 THE END
```

TABLE 3.7 A partial list of **OutputBox** methods.

CLASS:	**OutputBox**	
Method	**Argument**	**Description**

`print`	`number`
	`or`
	`text`
		Prints out the number or text passed as an argument in the dialog. Printing will continue from the end of currently displayed output.

`printLine`	`number`
	`or`
	`text`
		Same as the print method, but the line is skipped after the output so the next output will continue from the next line.

`skipLine`	`integer`	Skips N lines where N is an integer passed as an argument.
`saveToFile`	`filename`	Saves the contents of an OutputBox to a file whose name is passed as an argument. If the designated file already exists, then the current contents of the file are erased and replaced by the contents of the OutputBox.
`appendToFile`	`filename`	Appends the contents of an OutputBox to a file whose name is passed as an argument. If the designated file does not exist, then this method works like the saveToFile method.