Primitive Types, Strings, and Console I/O

Chapter 2

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020.

Objectives

- become familiar with Java primitive types (numbers, characters, etc.)
- learn about assignment statements and expressions
- learn about strings
- become familiar with classes, methods, and objects
- learn about simple keyboard input and screen output

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Outline

- Primitive Types and Expressions
- The Class `String`
- Keyboard and Screen I/O
- Documentation and Style

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Variables and Values

- *Variables* store data such as numbers and letters.
  - Think of them as places to store data.
  - They are implemented as memory locations.
- The data stored by a variable is called its *value.*
  - The value is stored in the memory location.
- Its value can be changed.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Variables and Values

- variables

```
numberOfBaskets
eggsPerBasket
totalEggs
```

- assigning values

```
eggsPerBasket = 6;
eggsPerBasket = eggsPerBasket - 2;
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Naming and Declaring Variables

- Choose names that are helpful such as `count` or `speed`, but not `c` or `s`.
- When you *declare* a variable, you provide its name and type.

```
int numberOfBaskets, eggsPerBasket;
```

- A variable's *type* determines what kinds of values it can hold (`int, double, char`, etc.).
- A variable must be declared before it is used.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Syntax and Examples

- syntax
  type variable_1, variable_2, …;
- examples
  ```
  int styleChoice, numberOfChecks;
  double balance, interestRate;
  char jointOrIndividual;
  ```


Types in Java

- A *class type*
  - a class of objects and has both data and methods.
  - `"Think Whirled Peas"` is a value of class type `String`
- A *primitive type*
  - simple, nondecomposable values such as an individual number or individual character.
  - `int`, `double`, and `char` are primitive types.

Naming Conventions

- Class types
  - begin with an uppercase letter (e.g. `String`).
- Primitive types
  - begin with a lowercase letter (e.g. `int`).
- Variables of both class and primitive types
  - begin with a lowercase letters (e.g. `myName`, `myBalance`).
  - Multiword names are "punctuated" using uppercase letters.

Where to Declare Variables

- Declare a variable
  - just before it is used or
  - at the beginning of the section of your program that is enclosed in {}.

```
public static void main(String[] args)
{ /* declare variables here */

 …

}
```


Java Identifiers

- An *identifier*
  - a name, such as the name of a variable.
- Identifiers may contain only
  - letters
  - digits (0 through 9)
  - the underscore character (_)
  - and the dollar sign symbol ($) which has a special meaning
  - *but the first character cannot be a digit.*

Java Identifiers, cont.

- identifiers may not contain any spaces, dots (`.`), asterisks (`*`), or other characters:
  `7-11  netscape.com  util.*` (not allowed)
- Identifiers can be arbitrarily long.
- Since Java is *case sensitive*, `stuff`, `Stuff`, and `STUFF` are different identifiers.

Keywords or Reserved Words

- Words such as `if` are called *keywords* or *reserved words* and have special, predefined meanings.
- Keywords *cannot* be used as identifiers.
- See Appendix 1 for a complete list of Java keywords.
- other keywords: `int`, `public`, `class`
- Appendix 1

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.

Primitive Types

- four integer types (`byte`, `short`, `int`, and `long`)
  - `int` is most common
- two floating-point types (`float` and `double`)
  - `double` is more common
- one character type (`char`)
- one boolean type (`boolean`)

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Primitive Types, cont.

Type Name	Kind of Value	Memory Used	Size Range
`byte`	*integer*	*1 byte*	−128 to 127
`short`	*integer*	*2 bytes*	−32768 to 32767
`int`	*integer*	*4 bytes*	−2147483648 *to* 2147483647
`long`	*integer*	*8 bytes*	−9223372036854775808 *to* 9223372036854775807
`float`	*floating-point number*	*4 bytes*	$\pm 3.40282347 \times 10^{+38}$ *to* $\pm 1.40239846 \times 10^{-45}$
`double`	*floating-point number*	*8 bytes*	$\pm 1.76769313486231570 \times 10^{+308}$ *to* $\pm 4.94065645841246544 \times 10^{-324}$
`char`	*single character (Unicode)*	*2 bytes*	*all Unicode characters*
`boolean`	`true` *or* `false`	*1 bit*	*not applicable*

Display 2.2
Primitive Types

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Examples of Primitive Values

- integer types

```
0   -1   365   12000
```

- floating-point types

```
0.99   -22.8   3.14159  5.0
```

- character type

```
`a`   `A`   `#`   ` `
```

- boolean type

```
true   false
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Assignment Statements

- An assignment statement is used to assign a value to a variable.

```
answer = 42;
```

- The “equal sign” is called the *assignment operator.*
- We say, “The variable named `answer` is assigned a value of 42,” or more simply, “`answer` is assigned 42.”

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Assignment Statements, cont.

- Syntax

```
variable = expression ;
```

where *`expression`* can be

- another variable,
- a *literal* or *constant* (such as a number),
- or something more complicated which combines variables and literals using *operators* (such as + and -)

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Assignment Examples

```
amount = 3.99;
firstInitial = 'W';
score = numberOfCards + handicap;
eggsPerBasket = eggsPerBasket - 2;
```

(last line looks weird in mathematics, why?)

Assignment Evaluation

- The expression on the right-hand side of the assignment operator (=) is evaluated first.
- The result is used to set the value of the variable on the left-hand side of the assignment operator.

```
score = numberOfCards + handicap;
eggsPerBasket = eggsPerBasket - 2;
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Specialized Assignment Operators

- Assignment operators can be combined with arithmetic operators (including `-`, `*`, `/`, and `%`, discussed later).

```
amount = amount + 5;
```

can be written as

```
amount += 5;
```

yielding the same results.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Simple Screen Output

```
System.out.println("The count is " + count);
```

- outputs the Sting literal `"The count is "` followed by the current value of the variable `count`.
- + means concatenation if one argument is a string

(an example of which of the three properties of OO languages?)

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Simple Input

- Sometimes the data needed for a computation are obtained from the user at run time.
- Keyboard input requires

```
import java.util.*
```

at the beginning of the file.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Simple Input, cont.

- Data can be entered from the keyboard using

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

followed, for example, by

```
eggsPerBasket = keyboard.nextInt();
```

which reads one `int` value from the keyboard and assigns it to `eggsPerBasket`.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Simple Input, cont.

- class EggBasket2

Number Constants

- Literal expressions such as `2`, `3.7`, or `'y'` are called *constants*.
- Integer constants can be preceded by a `+` or `-` sign, but cannot contain commas.
- Floating-point constants can be written
  - with digits after a decimal point or
  - using *e notation.*

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

e Notation

- e notation is also called *scientific notation* or *floating-point notation.*
- examples
  - `865000000.0` can be written as `8.65e8`
  - `0.000483` can be written as `4.83e-4`
- The number in front of the `e` does not need to contain a decimal point, eg. `4e-4`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Assignment Compatibilities

- Java is said to be *strongly typed.*
  - You can't, for example, assign a floating point value to a variable declared to store an integer.
- Sometimes conversions between numbers are possible.

```
doubleVariable = 7;
doubleVariable = intVariable ;
```

is possible even if `doubleVariable` is of type `double`, for example.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Assignment Compatibilities, cont.

- A value of one type can be assigned to a variable of any type further to the right

```
byte --> short --> int --> long
--> float --> double
```

but not to a variable of any type further to the left.

- You can assign a value of type `char` to a variable of type `int`.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Type Casting

- A *type cast* **creates** a value in a new type from the original type.
- For example,

```
double distance;
distance = 9.0;
int points;
points = (int)distance;
```

(illegal without `(int)`)

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Type Casting, cont.

- The value of `(int)distance` is `9`, but the value of `distance`, both before and after the cast, is `9.0`.
- The type of `distance` does NOT change and remains `float`.
- Any nonzero value to the right of the decimal point is *truncated,* rather than *rounded.*

Characters as Integers

- Characters are actually stored as integers according to a special code
  - each printable character (letter, number, punctuation mark, space, and tab) is assigned a different integer code
  - the codes are different for upper and lower case
  - for example 97 may be the integer value for 'a' and 65 for 'A'
- *ASCII* and *Unicode* are common character codes

Unicode Character Set

- Most programming languages use the *ASCII* character set.
- Java uses the *Unicode* character set which includes the ASCII character set (Appendix 3)
- The Unicode character set includes characters from many different alphabets other than English (but you probably won't use them).

ASCII/Unicode

32	33	34	35	36	37	38	39	40	41
	!	"	#	$	%	&	'	(	)

48	…	57	…	65	…	90	…	97	…	122
0		9		A		Z		a		z

Casting a **`char`** to an **`int`**

- Casting a `char` value to `int` produces the ASCII/Unicode value
- For example, what would the following display?

```
char answer = 'y';
System.out.println(answer);
System.out.println((int)answer);
```

- ```
  >y
  >121
  >
  ```


Initializing Variables

- A variable that has been declared, but no yet given a value is said to be *uninitialized.*
- Uninitialized class variables have the value `null`.
- Uninitialized primitive variables may have a default value.
- It's good practice *not* to rely on a default value, which could be arbitrary.
  ```

Initializing Variables, cont.

- To protect against an uninitialized variable (and to keep the compiler happy), assign a value at the time the variable is declared.
- Examples:

```
int count = 0;
char grade = 'A';   // default is an A
```


Initializing Variables, cont.

- syntax

```
type variable_1 = expression_1, variable_2 =
expression_2, …;
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Binary Representation

- Assume an 8-bit type
- 5 as an integer
  - 00000101
- '5' as a character
  - 00110101 (53 decimal, ASCII)
- 5.0 as a floating point number
  - How?
  - What about 5.5?

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Binary Real Numbers

…	2^3	2^2	2^1	2^0	.	2^{-1}	…

- 5.5
  - 101.1
- 5.25
  - 101.01
- 5.125
  - 101.001
- 5.75
  - 101.11

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

8 bits only

2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}

- 5.5
  - 101.1 -> 000101 10
- 5.25
  - 101.01 -> 000101 01
- 5.125
  - 101.001 -> ??
- With only 2 places after the point, the precision is .25
- What if the point is allowed to move around?

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Floating-point Numbers

- Decimal
  - 54.3
  - 5.43×10^1 [scientific notation]
- Binary
  - 101.001
  - 10.1001×2^1 [more correctly: 10.1001×10^1]
  - 1.01001×2^2 [more correctly: 1.01001×10^{10}]
  - What can we say about the most significant bit?

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Floating-point Numbers

- General form: *sign* 1.*mantissa* x $2^{exponent}$
  - the most significant digit is right before the dot
    - Always 1 [no need to represent it]
  - (more details are not discussed here: mantissa has no sign, but sign is embedded in exponent…)
- 1.01001 x 2^2
  - Sign: positive (0)
  - Mantissa: 01001
  - Exponent: 10 (decimal 2)
    - [IEEE standard: "biased exponent"
      - exponent + $2^{numBits-1}$ - 1
      - Example: 2 + 2^{2-1} – 1 = 3 => 11 in binary]

Java Floating-point Numbers

sign	exponent	mantissa

- Sign:
  - 1 bit [0 is positive]
- Mantissa:
  - 23 bits in `float`
  - 52 bits in `double`
- Exponent:
  - 8 bits in `float`
  - 11 bits in `double`

Imprecision in Floating-Point Numbers

- Floating-point numbers often are only approximations since they are stored with a finite number of bits.
- Hence `1.0/3.0` is slightly less than 1/3.
- `1.0/3.0 + 1.0/3.0 + 1.0/3.0` could be less than 1.
- `www.cs.fit.edu/~pkc/classes/cse1001/FloatEquality.java`

Arithmetic Operations

- Arithmetic expressions can be formed using the `+`, `-`, `*`, and `/` *operators*
  - together with variables or numbers referred to as *operands*.
  - When both operands are of the same type
    - the result is of that type.
  - When one of the operands is a floating-point type and the other is an integer
    - the result is a floating point type.

Arithmetic Operations, cont.

- Example

  If `hoursWorked` is an `int` to which the value `40` has been assigned, and `payRate` is a `double` to which `8.25` has been assigned

```
hoursWorked * payRate
```

is a `double` with a value of `500.0`.

Arithmetic Operations, cont.

- Expressions with two or more operators can be viewed as a series of steps, each involving only two operands.
  - The result of one step produces one of the operands to be used in the next step.
- example

```
balance + (balance * rate)
```

Operators with integer and floating point numbers

- if at least one of the operands is a floating-point type and the rest are integers
  - the result will be a floating point type.
- The result is the rightmost type from the following list that occurs in the expression.

```
byte --> short --> int --> long
--> float --> double
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020.

The Division Operator

- The division operator (`/`) behaves as expected
  - if one of the operands is a floating-point type.
- When both operands are integer types
  - the result is truncated, not rounded.
  - Hence, `99/100` has a value of `0`.
  - called integer division or integer divide

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

The `mod` Operator

- The `mod` (`%`) operator is used with operators of integer type to obtain
  - the remainder after integer division.
- 14 divided by 4 is 3 *with a remainder of 2.*
  - Hence, `14 % 4` is equal to `2`.
- The mod operator has many uses, including
  - determining if an integer is odd or even
  - determining if one integer is evenly divisible by another integer.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Case Study:Vending Machine Change

- requirements
  - The user enters an amount between 1 cent and 99 cents.
  - The program determines a combination of coins equal to that amount.
  - For example, 55 cents can be two quarters and one nickel.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Case Study, cont.

- sample dialog

```
Enter a whole number from 1 to 99.
The machine will determine a combination of coins.
87
87 cents in coins:
  3 quarters
  1 dime
  0 nickels
  2 pennies
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Case Study, cont.

- variables needed

```
int amount, quarters, dimes, nickels, pennies;
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Case Study, cont.

- algorithm - first version
  1. Read the amount.
  2. Find the maximum number of quarters in the amount.
  3. Subtract the value of the quarters from the amount.
  4. Repeat the last two steps for dimes, nickels, and pennies.
  5. Print the original amount and the quantities of each coin.

Case Study,cont.

- The algorithm doesn't work properly, because the original amount is changed by the intermediate steps.
  - The original value of `amount` is lost.
- Change the list of variables

```
int amount, originalAmount, quarters, dimes,
nickles, pennies;
```

- and update the algorithm.

Case Study, cont.

1. Read the amount.
2. Make a copy of the amount.
3. Find the maximum number of quarters in the amount.
4. Subtract the value of the quarters from the amount.
5. Repeat the last two steps for dimes, nickels, and pennies.
6. Print the original amount and the quantities of each coin.

Case Study, cont.

- Write Java code that *implements* the algorithm written in pseudocode.

Case Study, cont.

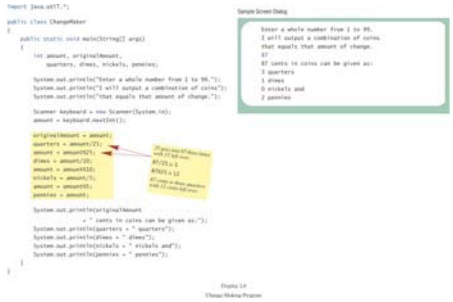
- How do we determine the number of quarters (or dimes, nickels, or pennies) in an amount?
  - There are 2 quarters in 55 cents, but there are also 2 quarters in 65 cents.
  - That's because
    `55 / 2 = 2` and `65 / 25 = 2.`

Case Study, cont.

- How do we determine the remaining amount?
  - using the mod operator
    `55 % 25 = 5` and `65 % 25 = 15`
  - similarly for dimes and nickels.
  - Pennies are simply `amount % 5.`

Case Study, cont.

- `class ChangeMaker`



```
import java.util.*;

public class ChangeMaker
{
    public static void main(String[] args)
    {
        int amount, originalAmount,
            quarters, dimes, nickels, pennies;

        System.out.println("Enter a whole number from 1 to 99.");
        System.out.println("I will output a combination of coins");
        System.out.println("that equals that amount of change.");

        Scanner keyboard = new Scanner(System.in);
        amount = keyboard.nextInt();

        originalAmount = amount;
        quarters = amount/25;
        amount = amount%25;
        dimes = amount/10;
        amount = amount%10;
        nickels = amount/5;
        amount = amount%5;
        pennies = amount;

        System.out.println(originalAmount
                     + " cents in coins can be given as:");
        System.out.println(quarters + " quarters");
        System.out.println(dimes + " dimes");
        System.out.println(nickels + " nickels and");
        System.out.println(pennies + " pennies");
    }
}
```

Sample Screen Dialog

```
Enter a whole number from 1 to 99.
I will output a combination of coins
that equals that amount of change.
87
87 cents in coins can be given as:
3 quarters
1 dimes
0 nickels and
2 pennies
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020.

Case Study—testing the implementation

- The program should be tested with several different amounts.
- Test with values that give zero values for each possible coin denomination.
- Test with amounts close to
  - extreme values such as 0, 1, 98 and 99
  - coin denominations such as 24, 25, and 26
    - Boundary values.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Increment (and Decrement) Operators

- used to increase (or decrease) the value of a variable by 1
- easy to use, important to recognize
- the increment operator

  `count++` or `++count`
- the decrement operator

  `count--` or `--count`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Increment (and Decrement) Operators

- "mostly" equivalent operations

```
count++;
++count;
count = count + 1;

count--;
--count;
count = count - 1;
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Increment (and Decrement) Operators in Expressions

- after executing

```
int m = 4;
int result = 3 * (++m)
```

  `result` has a value of `15` and `m` has a value of `5`
- after executing

```
int m = 4;
int result = 3 * (m++)
```

  result has a value of `12` and `m` has a value of `5`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Increment and Decrement Operator Examples

common code

```
int n = 3;
int m = 4;
int result;
```

What will be the value of m and result after each of these executes?

(a) **`result = n * ++m; //preincrement m`**

(b) **`result = n * m++; //postincrement m`**

(c) **`result = n * --m; //predecrement m`**

(d) **`result = n * m--; //postdecrement m`**

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Answers to Increment/Decrement Operator Questions

(a) 1) `m = m + 1;` `//m = 4 + 1 = 5`
2) `result = n * m;` `//result = 3 * 5 = 15`

(b) 1) `result = n * m;` `//result = 3 * 4 = 12`
2) `m = m + 1;` `//m = 4 + 1 = 5`

(c) 1) `m = m - 1;` `//m = 4 - 1 = 3`
2) `result = n * m;` `//result = 3 * 3 = 9`

(b) 1) `result = n * m;` `//result = 3 * 4 = 12`
2) `m = m - 1;` `//m = 4 - 1 = 3`

Summary of Operators

- `+, -, *, /`
- `%`
- `++, --`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Parentheses and Precedence

- Parentheses can communicate the order in which arithmetic operations are performed
- examples:

```
(cost + tax) * discount
cost + (tax * discount)
```

- Without parentheses, an expression is evaluated according to the *rules of precedence.*

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Precedence Rules

Highest Precedence

First: the unary operators: +, –, ++, – –, and !
Second: the binary arithmetic operators: *, /, and %
Third: the binary arithmetic operators: + and –

Lowest Precedence

Display 2.4
Precedence Rules

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Precedence Rules—Binary Operators

- The *binary* arithmetic operators `*`, `/`, and `%`
  - have *lower precedence* than the *unary* operators `+`, `-`, `++`, `--`, and `!`
  - but have *higher precedence* than the binary arithmetic operators `+` and `-`. (Appendix 2)
- When binary operators have equal precedence
  - the operator on the left has higher precedence than the operator(s) on the right.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Precedence Rules—Unary Operators

- When unary operators have equal precedence
  - the operator on the right has higher precedence than the operation(s) on the left
    - opposite order to binary operators
  - if `x` is 10
    - `-++x` is -11 and `x` is 11 afterwards
    - same as `-(++x)`
  - if `x` is 10
    - `-x++` is -10 and `x` is 11 afterwards
    - same as `-(x++)`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Use Parentheses

- Even when parentheses are not needed, they can be used to make the code clearer.

```
balance + (interestRate * balance)
```

- [Spaces also make code clearer

```
balance + interestRate*balance
```

but spaces do not dictate precedence.]

Sample Expressions

Ordinary Mathematical Expression	Java Expression (Preferred Form)	Equivalent Fully Parenthesized Java Expression
$rate^2 + delta$	`rate*rate + delta`	`(rate*rate) + delta`
$2(salary + bonus)$	`2*(salary + bonus)`	`2*(salary + bonus)`
$\frac{1}{time + 3\,mass}$	`1/(time + 3*mass)`	`1/(time + (3*mass))`
$\frac{a - 7}{t + 9v}$	`(a - 7)/(t + 9*v)`	`(a - 7)/(t + (9*v))`

Display 2.5
Arithmetic Expressions in Java

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

The Class `String`

- We've used constants of type `String` already.

```
"Enter a whole number from 1 to 99."
```

- A value of type `String` is a sequence of characters treated as a single item.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Declaring and Printing Strings

- declaring

```
String greeting;
 greeting = "Hello!";
```

or

```
String greeting = "Hello!";
```

or

```
String greeting = new String("Hello!");
```

- printing

```
System.out.println(greeting);
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Concatenation of Strings

- Two strings are *concatenated* using the `+` operator.

```
String greeting = "Hello";
String sentence;
sentence = greeting + " officer";
System.out.println(sentence);
```

- Any number of strings can be concatenated using the `+` operator.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Concatenating Strings and Integers

```
String solution;
solution = "The temperature is " + 72;
System.out.println (solution);
```

`The temperature is 72`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Classes

- A *class* is a type used to produce objects.
- An *object* is an entity that stores data and can take actions defined by *methods.*
- An object of the `String` class stores data consisting of a sequence of characters.
- The `length()` method returns the number of characters in a particular `String` object.

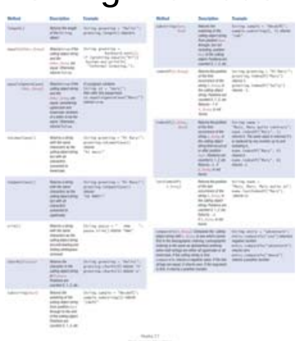
```
int howMany = solution.length()
```


Objects, Methods, and Data

- Objects within a class
  - have the same methods
  - have the same kind(s) of data but the data can have different values.
- Primitive types have values, but no methods.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

String Methods



JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

The Method `length()`

- The method `length()` returns an `int`.
- You can use a call to method `length()` anywhere an `int` can be used.

```
int count = solution.length();
System.out.println(solution.length());
spaces = solution.length() + 3;
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

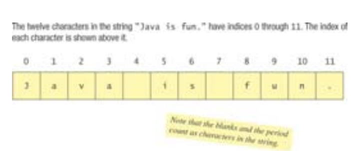
Positions in a `String`

- positions start with 0, not 1.
  - The `'J'` in `"Java is fun."` is in position 0

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Positions in a `String`, cont.

- A position is referred to an an *index.*
  - The `'f'` in `"Java is fun."` is at index 9.



Display 2.8
String Indices

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Indexing Characters within a String

- `charAt(position)`method
  - returns the char at the specified position
- `substring(start, end)` method
  - returns the string from `start` upto **excluding** `end`
- For example:

```
String greeting = "Hi, there!";
greeting.charAt(0)       returns  H
greeting.charAt(2)       returns  ,
greeting.substring(4,7)  returns  the
```

H	i	,		t	h	e	r	e	!
0	1	2	3	4	5	6	7	8	9

Using the `String` Class

- `class StringDemo`

```
public class StringDemo
{
    public static void main(String[] args)
    {
        String sentence = "Text processing is hard!";
        int position;
        position = sentence.indexOf("hard");
        System.out.println(sentence);
        System.out.println("012345678901234567890123");
        System.out.println("The word \"hard\" starts at index "
                                  + position);
        sentence = sentence.substring(0, position) + "easy!";
        System.out.println("The changed string is:");
        System.out.println(sentence);
    }
}
```

Sample Screen Dialog

```
Text processing is hard!
012345678901234567890123
The word "hard" starts at index 19
The changed string is:
Text processing is easy!
```

The meaning of \" is discussed in the subsection entitled "Escape Characters."

Display 2.9
Using the String Class

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Escape Characters

- How would you print
  `“Java” refers to a language.`?
- The compiler needs to be told that the quotation marks (“) do not signal the start or end of a string, but instead are to be printed.

```
System.out.println(
“\”Java\” refers to a language.”);
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Escape Characters

`\"`	Double quote.
`\'`	Single quote.
`\\`	Backslash.
`\n`	New line. Go to the beginning of the next line.
`\r`	Carriage return. Go to the beginning of the current line.
`\t`	Tab. Add whitespace up to the next tab stop.

Display 2.10
Escape Characters

- Each escape sequence is a single character even though it is written with two symbols.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Examples

```
System.out.println(“abc\\def”);
abc\def
System.out.println(“new\nline”);
new
line
char singleQuote = ‘\’’;
System.out.println(singleQuote);
’
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

The Unicode Character Set

- Most programming languages use the *ASCII* character set.
- Java uses the *Unicode* character set which includes the ASCII character set
  - Backward compatible to ASCII
- The Unicode character set includes characters from many different alphabets (but you probably won’t use them).

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Keyboard and Screen I/O: Outline

- Screen Output
- Keyboard Input

Screen Output

- We've seen several examples of screen output already.
- `System.out` is an object that is part of Java.
- `println()` is one of the methods available to the `System.out` object.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 0131149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Screen Output, cont.

- The concatenation operator (+) is useful when everything does not fit on one line.

```
System.out.println("When everything " +
  "does not fit on one line, use the" +
  " concatenation operator (\'+\')");
```

  – Do not break the line except immediately before or after the concatenation operator (+).

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 0131149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Screen Output, cont.

- Alternatively, use `print()`

```
System.out.print("When everything ");
System.out.print("does not fit on ");
System.out.print("one line, use the ");
System.out.print("\"print\" ");
System.out.println("statement");
```

  ending with a `println()`.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 0131149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Screen Output, cont.

- syntax

```
System.out.println(output_1 + output_2 + ...+
  output_n);
```

- example

```
System.out.println (1967 + " " + "Oldsmobile"
  + " " + 442);
```

`1967 Oldsmobile 442`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 0131149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

`printf` (or `format`) Method for Output Formatting

- Heavily influenced by C
- *`outputStream`*`.printf(`*`formatString, args…`*`)`
  – `System.out.printf(…)`
  – `smileyOutStream.printf(…)`
- *`formatString`* specifies how to format *`args`*
- `System.out.printf("%s %d %f%n", name, id, gpa);`
  – `System.out.println(name + "  " + id + " " + gpa);`
- Useful for "right justified" numbers
- Numbers in `println` and `print` are "left justified"

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 0131149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Formatting String

- % *width conversion*
- *width* specifies how many slots are available for output
- If *width* > number of characters, spaces are printed first before the characters—"right justified"
- `printf("%5d", count)`
  - `Count`
    - `32901:`

3	2	9	0	1

    - `2004:`

	2	0	0	4

    - `22:`

			2	2

    - `6747280:` all digits are printed, width is ignored

Conversion Characters

Conversion	Argument	Description
d	integer	Decimal integer
f	floating point	Decimal float
s	general (String, Boolean, …)	String
n		New line
c	character	Character (unicode)
e	floating point	Decimal scientific notation
o	integer	Octal integer
x	integer	Hexadecimal integer
%		% (%% to output %)

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Floating-point Precision

- *width.precision conversion*
- `printf("%5.2f", PI)`

	3	.	1	4

- `printf("%7.4f", PI)`

	3	.	1	4	1	5

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Left Justified

- Spaces are added (padded) on the right
- Minus (`-`) sign before the *width*
- …`printf("%-7s %-4d", name, age)`

J	o	h	n					2	0		

- Why are there 4 spaces after "John" instead of 3?

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Example

- http://www.cs.fit.edu/~pkc/classes/cse1001/Printf.java

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Keyboard Input

- Starting from Java 5.0
  - Java has reasonable facilities for handling keyboard input.
- `Scanner` class in the `java.util` package
  - A *package* is a library of classes.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Using the Scanner Class

- Near the beginning of your program, insert
```
import java.util.*
```
- Create an object of the `Scanner` class
```
Scanner keyboard =
   new Scanner(System.in)
```
- Read data (an `int` or a `double`, for example)
```
int n1 = keyboard.nextInt();
double d1 = keyboard.nextDouble();
```


Some `Scanner` Class Methods

- syntax
```
Int_Variable =    Object_Name.nextInt();
Double_Variable = Object_Name.nextDouble();
Float_Variable =  Object_Name.nextFloat();

String_Variable = Object_Name.next();
String_Variable = Object_Name.nextLine();

Boolean_Variable = Object_Name.nextBoolean();

nextByte(), nextShort(), nextLong()
```


Some Scanner Class Methods, cont.

- examples
```
int count = keyboard.nextInt();
double distance = keyboard.nextDouble();
String word = keyboard.next();
String wholeLine = keyboard.nextLine();
```
- Remember to prompt the user for input, e.g.
```
System.out.print("Enter an integer: ");
```


Keyboard Input Demonstration

- `class ScannerDemo`

`nextLine()` Method Caution

- The `nextLine()` method reads the remainder of the current line, even if it is empty.

`nextLine()` Method Caution, cont.

- example
```
int n;
String s1, s2;
n = keyboard.nextInt();
s1 = keyboard.nextLine();
s2 = keyboard.nextLine();
```
`5440`
`or bust`

`n` is set to `5440`

but `s1` is set to the empty string.

The Empty String

- String with zero characters

```
String s3 = "";
```

- Good for String initialization

Other Input Delimiters

- Characters for separating "words"
  - Default is "whitespace": space, tab, newline
- Change the delimiter to `"##"`

```
keyboard2.useDelimiter("##");
```

  - whitespace will no longer be a delimiter for `keyboard2` input

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Other Input Delimiters, cont.

- `class DelimitersDemo`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Documentation and Style: Outline

- Meaningful Names
- Self-Documentation and Comments
- Indentation
- Named Constants
- www.cs.fit.edu/~pkc/classes/cse1001/FirstProgramOneLine.java

- Grading
  - 10% on documentation and comments
  - 10% on style (variable naming, indentation)

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Documentation and Style

- Most programs are modified over time to respond to new requirements.
- Programs which are easy to read and understand are easy to modify.
- Even if it will be used only once, you have to read it in order to debug it .

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Meaningful Names for Variables

- A variable's name should suggest its use.
- Observe conventions in choosing names for variables.
  - Use only letters and digits.
  - Use more than one character.
  - "Punctuate" using uppercase letters at word boundaries (e.g. `taxRate`).
  - Start variables with lowercase letters.
  - Start class names with uppercase letters.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Documentation and Comments

- The best programs are self-documenting.
  - clean style
  - well-chosen names
- Comments are written into a program as needed explain the program.
  - They are useful to the programmer, but they are ignored by the compiler.

When to Use Comments

- Begin each program file with an explanatory comment
  - what the program does
  - the name of the author
  - contact information for the author
  - date of the last modification.
- Provide only those comments which the expected reader of the program file will need in order to understand it.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Comments Example

- `class CircleCalculation`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Template for CSE 1001

```
/**
  Name: <name>
  E-mail address: <email>
  Course: cse1001
  Section: <?>
  Assignment: <HW?>
  Date: <date>

  Description: <description>

*/

import java.util.*;

public class <SameClassAndFileNames>
{
    public static void main(String[] args)
    {
        // input from the keyboard
        Scanner keyboard = new Scanner(System.in);



    }
}
```

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Comments

- A program can usually be broken into segments/blocks based on the algorithm, e.g. in HW2:
  - Prompt the user for input
  - Input from the keyboard
  - Calculation
  - Output to the screen
- Blank line between two segments
- A description (comment) before each segment

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Pseudocode and Comments

- Solving a problem
  1. Devise an algorithm (steps to solve the problem)
  2. Write the algorithm in pseudocode (semi English, semi Java)
  3. English part of pseudocode becomes comments in your program
- Tip:
  1. type the English part of pesudocode as comments into your program first
  2. write the detailed Java instructions to satisfy/implement the pesudocode
- Advantages:
  1. Each line of pseudocode helps you focus on a small task
  2. Pseudocode tells you what steps you want to achieve
  3. No need to add comments later on

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Comments

- A comment can begin with //.
  – Everything after these symbols and to the end of the line is treated as a comment and is ignored by the compiler.

```
double radius; //in centimeters
```


Comments, cont.

- A comment can begin with /* and end with */
  – Everything between these symbols is treated as a comment and is ignored by the compiler.

```
/* the simplex method is used to
  calculate the answer*/
```


Comments, cont.

- A *javadoc* comment, begins with /** and ends with */.
  – It can be extracted automatically from Java software.

```
/** method change requires the number of coins
  to be nonnegative */
```


Indentation

- Indentation should communicate nesting clearly.
- A good choice is four spaces for each level of indentation.
- Indentation should be consistent.
- Indentation should be used for second and subsequent lines of statements which do not fit on a single line.

Indentation, cont.

- Indentation does not change the behavior of the program.
- Improper indentation can miscommunicate the behavior of the program.

Named Constants

- To avoid confusion, always name constants (and variables).

```
circumference = PI * radius;
```

  is clearer than

```
circumference = 3.14159 * 6.023;
```

- Place constants near the beginning of the program.

Named Constants, cont.

- The value of a constant cannot be changed once it is initialized

```
public static final double INTEREST_RATE = 6.65;
```

- Consider the interest rate is used many times in the program:
  - What if you type 6.65 in some places, but 6.56 in others?
  - What if the interested rate has changed to 7.3?
  - Is `balance * INTEREST_RATE` easier to read than `balance * 6.65` ?

Declaring Constants

- syntax

```
public static final Type Name = Constant;
```

- examples

```
public static final double PI = 3.14159;
public static final String MOTTO = "The
  customer is always right.";
```

  - By convention, uppercase letters are used for constants.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Named Constants

- `class CircleCalculation2`

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Summary

- You have become familiar with Java primitive types (numbers, characters, etc.).
- You have learned about assignment statements and expressions.
- You have learned about stings.
- You have become familiar with the basics of classes, methods, and objects.
- You have learned about simple keyboard input and screen output.

JAVA: An Introduction to Problem Solving & Programming, Fourth Edition by Walter Savitch.
ISBN 013149020. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.