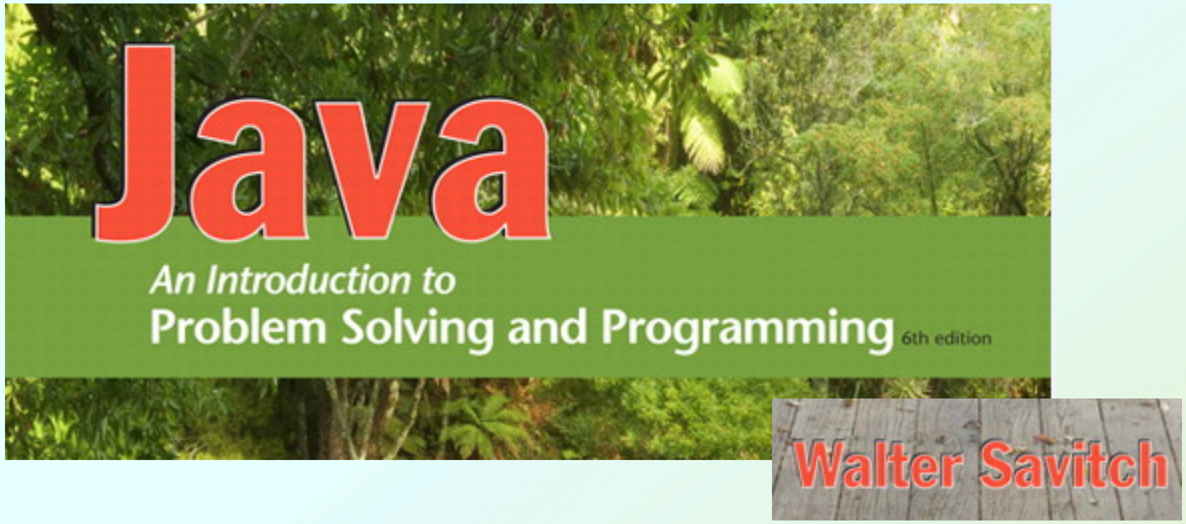


Basic Computation

Chapter 2

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Objectives

- Describe the Java data types used for simple data
- Write Java statements to declare variables, define named constants
- Write assignment statements, expressions containing variables and constants
- Define strings of characters, perform simple string processing

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Objectives

- Write Java statements that accomplish keyboard input, screen output
- Adhere to stylistic guidelines and conventions
- Write meaningful comments
- Creating a GUI application using the **`JFrame`** class
- Use the class **`JOptionPane`** for window-based input and output

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Outline

- Variables and Expressions
- The Class **`String`**
- Keyboard and Screen I/O
- Documentation and Style
- Graphics Supplement

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Variables and Expressions: Outline

- Variables
- Data Types
- Java Identifiers
- Assignment Statements
- Simple Input
- Simple Screen Output
- Constants
- Named Constants

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Variables and Expressions: Outline

- Assignment Compatibilities
- Type Casting
- Arithmetic Operations
- Parentheses and Precedence Rules
- Specialized Assignment Operators
- *Case Study:* Vending Machine Change
- Increment and Decrement Operators

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Variables

- *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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Variables

- View sample program listing 2.1
  - **`Class EggBasket`**

```
If you have
6 eggs per basket and
10 baskets, then
the total number of eggs is 60
```

Sample Screen Output

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Variables and Values

- Variables

```
numberOfBaskets
eggsPerBasket
totalEggs
```

- Assigning values

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


ISBN 0132162709 © 2012 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Syntax and Examples

- Syntax

  `type variable_1, variable_2, …;`

  `(variable_1` is a generic variable called a *syntactic variable*)

- Examples

```
int styleChoice, numberOfChecks;
double balance, interestRate;
char jointOrIndividual;
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Data Types

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Primitive Types

- Figure 2.1 Primitive Types

Type Name	Kind of Value	Memory Used	Range of Values
`byte`	Integer	1 byte	–128 to 127
`short`	Integer	2 bytes	–32,768 to 32,767
`int`	Integer	4 bytes	–2,147,483,648 to 2,147,483,647
`long`	Integer	8 bytes	–9,223,372,036,8547,75,808 to 9,223,372,036,854,775,807
`float`	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$
`double`	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
`char`	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
`boolean`		1 bit	True or false

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Java Identifiers

- An *identifier* is 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
- The first character <u>cannot</u> be a digit.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Java Identifiers

- 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Keywords or Reserved Words

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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 */
      . . .
}
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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`)

ISBN 0132162709 © 2012 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

ISBN 0132162709 © 2012 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."

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Assignment Statements

- 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 -)

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Assignment Examples

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


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Initializing Variables

- 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';
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Initializing Variables

- syntax

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


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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;
```


ISBN 0132162709 © 2012 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.Scanner`

  at the beginning of the file.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Simple Input

- 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`.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Simple Input

- View sample program listing 2.2
  `class EggBasket2`

```
Enter the number of eggs in each basket:
6
Enter the number of baskets:
10
If you have
6 eggs per basket and
10 baskets, then
the total number of eggs is 60
Now we take two eggs out of each basket.
You now have
4 eggs per basket and
10 baskets.
The new total number of eggs is 40
```

Sample screen output

ISBN 0132162709 © 2012 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`.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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.*

ISBN 0132162709 © 2012 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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` is less than 1.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Named Constants

- Java provides mechanism to …
  - Define a variable
  - Initialize it
  - Fix the value so it cannot be changed

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

  - Example

```
public static final double PI = 3.14159;
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 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;
  ```

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Assignment Compatibilities

- 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`**.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Type Casting

- A *type cast* temporarily changes the value of a variable from the declared type to some other type.
- For example,

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

- Illegal without `(int)`

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Type Casting

- The value of `(int)distance` is `9`,
- The value of `distance`, both before and after the cast, is `9.0`.
- Any nonzero value to the right of the decimal point is *truncated* rather than *rounded.*

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Arithmetic Operators

- 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Arithmetic Operations

- 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`.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Arithmetic Operations

- 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)
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Arithmetic Operations

- 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, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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`.

ISBN 0132162709 © 2012 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.

ISBN 0132162709 © 2012 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 expressions is evaluated according to the *rules of precedence.*

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Precedence Rules

- Figure 2.2 Precedence Rules

Highest Precedence

First: the unary operators +, -, !, ++, and --

Second: the binary arithmetic operators *, /, and %

Third: the binary arithmetic operators + and –

Lowest Precedence

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Precedence Rules

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Precedence Rules

- When unary operators have equal precedence, the operator on the right acts before the operation(s) on the left.
- 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Sample Expressions

- Figure 2.3 Some Arithmetic Expressions in Java

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

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 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.

ISBN 0132162709 © 2012 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- 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
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- Variables needed

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


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- 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.
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study,cont.

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

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

  - Update the algorithm.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- Algorithm – second version

```
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.
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- View Java code that *implements* the algorithm written in pseudocode – listing 2.3

```
Enter a whole number from 1 to 99.
I will find 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
```

Sample Screen Output

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- 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.
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- How do we determine the remaining amount?
- The remaining amount can be determined using the **`mod`** operator

    `55 % 25 = 5 and 65 % 25 = 15`

- Similarly for dimes and nickels.
- Pennies are simply **`amount % 5.`**

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Case Study

- 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.

ISBN 0132162709 © 2012 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`

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Increment and Decrement Operators

- equivalent operations

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

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


ISBN 0132162709 © 2012 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, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

String Constants and Variables

- Declaring

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

or

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

or

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

- Printing

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


ISBN 0132162709 © 2012 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Concatenating Strings and Integers

```
String solution;
solution = "The answer is " + 42;
System.out.println (solution);
```

```
The answer is 42
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

String Methods

- An object of the `String` class stores data consisting of a sequence of characters.
- Objects have methods as well as data
- The `length()` method returns the number of characters in a particular `String` object.

```
String greeting = "Hello";
int n = greeting.length();
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

The Method `length()`

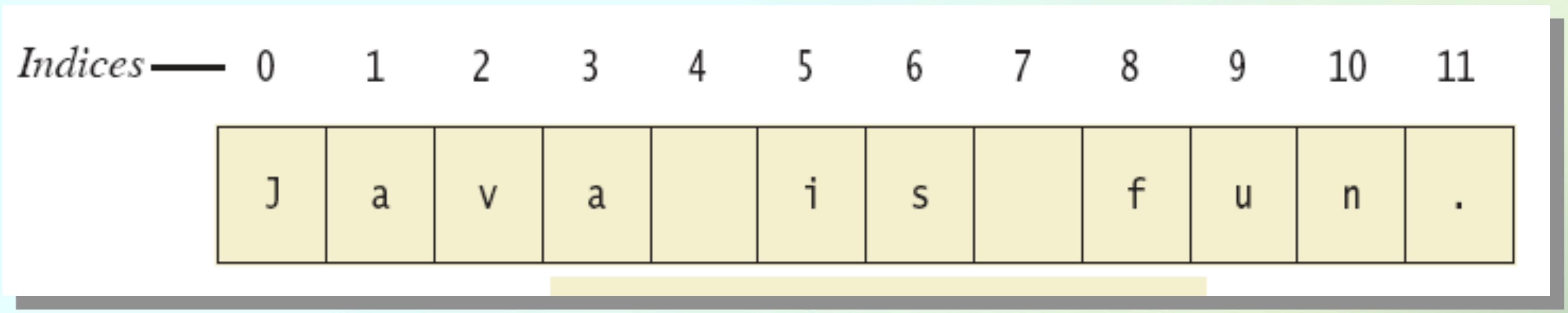
- The method **`length()`** returns an **`int`**.
- You can use a call to method **`length()`** anywhere an **`int`** can be used.

```
int count = command.length();
System.out.println("Length is " +
    command.length());
count = command.length() + 3;
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`String` Indices

- Figure 2.4



- Positions start with 0, not 1.
  - The 'J' in "Java is fun." is in position 0
- A position is referred to an an *index.*
  - The `'f'` in `"Java is fun."` is at index 8.

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`String` Methods

`charAt(`*Index*`)`
Returns the character at *Index* in this string. Index numbers begin at 0.

`compareTo(`*A_String*`)`
Compares this string with *A_String* to see which string comes first in the lexicographic ordering. (Lexicographic ordering is the same as alphabetical ordering when both strings are either all uppercase letters or all lowercase letters.) Returns a negative integer if this string is first, returns zero if the two strings are equal, and returns a positive integer if *A_String* is first.

`concat(`*A_String*`)`
Returns a new string having the same characters as this string concatenated with the characters in *A_String*. You can use the ⇓ operator instead of `concat`.

`equals(`*Other_String*`)`
Returns true if this string and *Other_String* are equal. Otherwise, returns false.

Figure 2.5a

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`String` Methods

`equalsIgnoreCase(`*Other_String*`)` Behaves like the method `equals`, but considers uppercase and lowercase versions of a letter to be the same.
`indexOf(`*A_String*`)` Returns the index of the first occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.
`lastIndexOf(`*A_String*`)` Returns the index of the last occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.

Figure 2.5b

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`String` Methods

`length()` Returns the length of this string.	
`toLowerCase()` Returns a new string having the same characters as this string, but with any uppercase letters converted to lowercase.	
`toUpperCase()` Returns a new string having the same characters as this string, but with any lowercase letters converted to uppercase.	

Figure 2.5c

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`String` Methods

`replace(`*OldChar, NewChar*`)` Returns a new string having the same characters as this string, but with each occurrence of *OldChar* replaced by *NewChar*.
`substring(`*Start*`)` Returns a new string having the same characters as the substring that begins at index *Start* of this string through to the end of the string. Index numbers begin at 0.
`substring(`*Start, End*`)` Returns a new string having the same characters as the substring that begins at index *Start* of this string through, but not including, index *End* of the string. Index numbers begin at 0.
`trim()` Returns a new string having the same characters as this string, but with leading and trailing whitespace removed.

Figure 2.5d

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

String Processing

- No methods allow you to change the value of a **String** object.
- But you can change the value of a **String** variable.
- View sample program **StringDemo** listing 2.4

```
Text processing is hard!
012345678901234567890123
The word "hard" starts at index 19
The changed string is:
TEXT PROCESSING IS EASY!
```

Sample Screen Output

ISBN 0132162709 © 2012 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.");
```


ISBN 0132162709 © 2012 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.

- Figure 2.6
- Each escape sequence is a single character even though it is written with two symbols.

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 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, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 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.
- The Unicode character set includes characters from many different alphabets (but you probably won't use them).

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Keyboard and Screen I/O: Outline

- Screen Output
- Keyboard Input

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Screen Output

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

```
System.out.println("Lucky number = "
  + 13 +
  "Secret number = " + number);
```

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Screen Output

- Alternatively, use **`print()`**

```
System.out.print("One, two,");
System.out.print(" buckle my shoe.");
System.out.println(" Three, four,");
System.out.println(" shut the door.");
```

ending with a **`println()`**.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Keyboard Input

- Java has reasonable facilities for handling keyboard input.
- These facilities are provided by the **`Scanner`** class in the **`java.util`** package.
  - A *package* is a library of classes.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Using the Scanner Class

- Near the beginning of your program, insert

```
import java.util.Scanner;
```

- 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();
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Keyboard Input Demonstration

- View sample program

  class ScannerDemo, listing 2.5

```
Enter two whole numbers
separated by one or more spaces:
  42   43
You entered 42 and 43
Next enter two numbers.
A decimal point is OK.
 9.99  21
You entered 9.99 and 21.0
Next enter two words:
plastic spoons
You entered "plastic" and "spoons"
Next enter a line of text:
May the hair on your toes grow long and curly.
You entered "May the hair on your toes grow long and curly."
```

Sample Screen Output

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Some `Scanner` Class Methods

- Figure 2.7a

Scannner_Object_Name`.next()`
Returns the `String` value consisting of the next keyboard characters up to, but not including, the first delimiter character. The default delimiters are whitespace characters.

Scannner_Object_Name`.nextLine()`
Reads the rest of the current keyboard input line and returns the characters read as a value of type `String`. Note that the line terminator `'\n'` is read and discarded; it is not included in the string returned.

Scannner_Object_Name`.nextInt()`
Returns the next keyboard input as a value of type `int`.

Scannner_Object_Name`.nextDouble()`
Returns the next keyboard input as a value of type `double`.

Scannner_Object_Name`.nextFloat()`
Returns the next keyboard input as a value of type `float`.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Some `Scanner` Class Methods

- Figure 2.7b

Scannner_Object_Name`.nextLong()`
Returns the next keyboard input as a value of type `long`.

Scannner_Object_Name`.nextByte()`
Returns the next keyboard input as a value of type `byte`.

Scannner_Object_Name`.nextShort()`
Returns the next keyboard input as a value of type `short`.

Scannner_Object_Name`.nextBoolean()`
Returns the next keyboard input as a value of type `boolean`. The values of `true` and `false` are entered as the words *true* and *false*. Any combination of uppercase and lowercase letters is allowed in spelling *true* and *false*.

Scanner_Object_Name`.useDelimiter(`*Delimiter_Word*`);`
Makes the string *Delimiter_Word* the only delimiter used to separate input. Only the exact word will be a delimiter. In particular, blanks, line breaks, and other whitespace will no longer be delimiters unless they are a part of *Delimiter_Word*.

This is a simple case of the use of the `useDelimiter` method. There are many ways to set the delimiters to various combinations of characters and words, but we will not go into them in this book.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`nextLine()` Method Caution

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`nextLine()` Method Caution

- Example – given following declaration.

```
int n;
String s1, s2;
n = keyboard.nextInt();
s1 = keyboard.nextLine();
s2 = keyboard.nextLine();
```

- Assume input shown

```
42
and don't you
forget it.
```

`n` is set to `42`

but `s1` is set to the empty string.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

The Empty String

- A string can have any number of characters, including zero.
- The string with zero characters is called the *empty* string.
- The empty string is useful and can be created in many ways including

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


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Other Input Delimiters (optional)

- Almost any combination of characters and strings can be used to separate keyboard input.
- to change the delimiter to `"##"`

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

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Other Input Delimiters

- View sample program

class DelimitersDemo, listing 2.6

```
Enter a line of text with two words:
funny wo##rd##
The two words are "funny" and "wor##rd##"
Enter a line of text with two words
delimited by ##:
funny wor##rd##
The two words are "funny wo" and "rd"
```

Sample Screen Output

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Documentation and Style: Outline

- Meaningful Names
- Comments
- Indentation
- Named Constants

ISBN 0132162709 © 2012 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 .

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Meaningful Variable Names

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

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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.

ISBN 0132162709 © 2012 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
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Comments

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

```
/**
This program should only
be used on alternate Thursdays,
except during leap years, when it should
only be used on alternate Tuesdays.
*/
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Comments

- 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
  */
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Comments Example

- View sample program

  `class CircleCalculation,` listing 2.7

```
Enter the radius of a circle in inches:
2.5
A circle of radius 2.5 inches
has an area of 19.6349375 square inches.
```

Sample Screen Output

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Indentation

- Indentation does not change the behavior of the program.
- Proper indentation helps communicate to the human reader the nested structures of the program

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Using Named Constants

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

```
area = PI * radius * radius;
```

is clearer than

```
area = 3.14159 * radius * radius;
```

- Place constants near the beginning of the program.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Named Constants

- Once the value of a constant is set (or changed by an editor), it can be used (or reflected) throughout the program.

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

- If a literal (such as 6.65) is used instead, every occurrence must be changed, with the risk than another literal with the same value might be changed unintentionally.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Declaring Constants

- Syntax

```
public static final
  Variable_Type = 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.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Named Constants

- View sample program
  class CircleCalculation2, listing 2.8

```
Enter the radius of a circle in inches:
2.5
A circle of radius 2.5 inches
has an area of 19.6349375 square inches.
```

Sample Screen Output

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Graphics Supplement: Outline

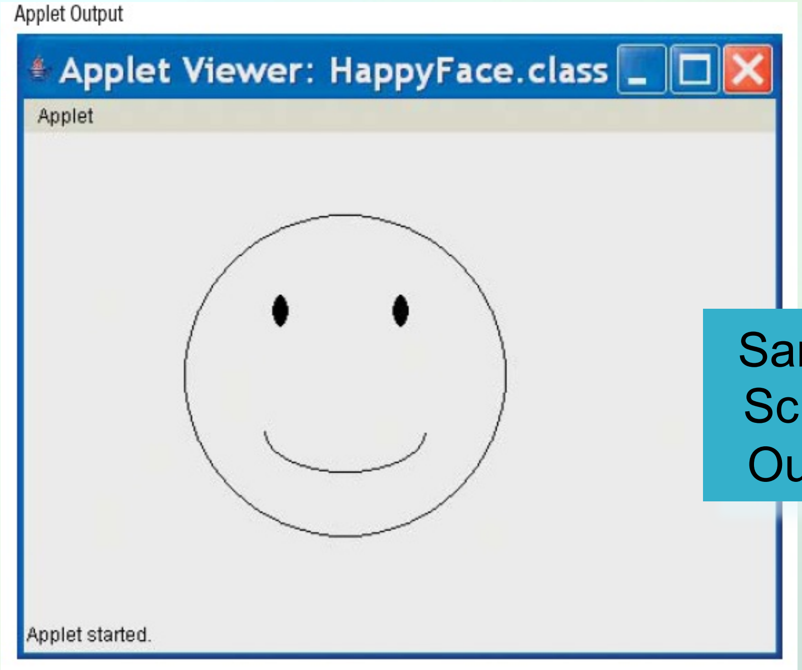
- Style Rules Applied to a Graphics Applet
- **`JFrame`**
- **`JOptionPane`**
- Inputting Numeric Types
- Multi-Line Output Windows

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Style Rules Applied to a Graphics Applet

- View sample program

class HappyFace, listing 2.9



Sample Screen Output

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Style Rules Applied to a Graphics Applet

- Named constants makes it easier to find values.
- Comments and named constants make changing the code much easier.
- Named constants protect against changing the wrong value.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Creating a `JFrame` Application

- The class `JFrame` can be used to create a standalone GUI application
- Full explanation of how this work in Chapter 8
- Similar to creating an applet
  1. Replace the line

     ```
     import javax.swing.JApplet;
     ```

     with

     ```
     import javax.swing.JFrame;
     ```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Creating a **JFrame** Application

2. Replace the text

```
extends JApplet;
```

with

```
extends JFrame;
```

3. Add what is called a constructor

```
public NameOfClass()
{
        setSize(600,400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
}
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Creating a **JFrame** Application

4. Add a main method that displays the window

```
public static void main(String[] args)
{
    HappyFace guiWindow = new HappyFace();
    guiWindow.setVisible(true);
}
```

5. View sample program

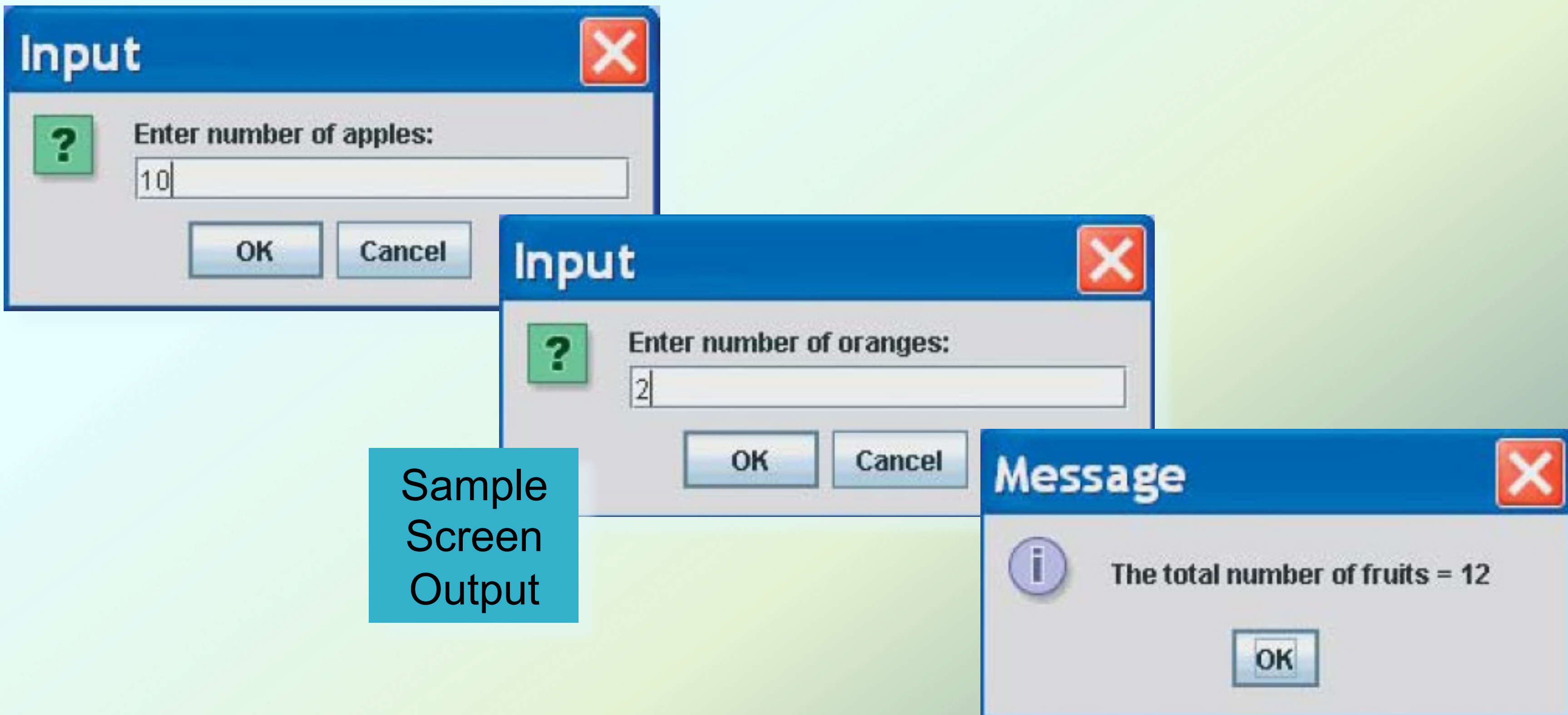
class HappyFaceJFrame, listing 2.10

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

JOptionPane

- View sample program

class JOptionPaneDemo, listing 2.11



Sample Screen Output

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

JOptionPane

- `JOptionPane` can be used to construct windows that interact with the user.
- The `JOptionPane` class is imported by

```
import javax.swing.JApplet;
```

- The `JOptionPane` class produces windows for obtaining input or displaying output.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

JOptionPane

- Use `showInputDialog()` for input .
- Only string values can be input.
- To convert an input value from a string to an integer use the `parseInt()` method from the `Integer` class, use

```
appleCount =
  Integer.parseInt(appleString);
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

JOptionPane

- Output is displayed using the **showMessageDialog** method.

```
JOptionPane.showMessageDialog(null,
  "The total number of fruits = " +
  totalFruitCount);
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

JOptionPane

- Syntax
  - Input

```
String_Variable =
JOptionPane.showInputDialogue
(String_Expression);
```

  - Output

```
JOptionPane.showMessageDialog
(null, String_Expression);
```

- `System.exit(0)` ends the program.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

`JOptionPane` Cautions

- If the input is not in the correct format, the program will *crash.*
- If you omit the last line (`System.exit(0)`), the program will not end, even when the OK button in the output window is clicked.
- Always label any output.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Inputting Numeric Types

- **`JOptionPane.showInputDialog`** can be used to input any of the numeric types.
- Figure 2.8 Methods for converting strings to numbers

Result Type	Method for Converting
`byte`	`Byte.parseByte(`*String_To_Convert*`)`
`short`	`Short.parseShort(`*String_To_Convert*`)`
`int`	`Integer.parseInt(`*String_To_Convert*`)`
`long`	`Long.parseLong(`*String_To_Convert*`)`
`float`	`Float.parseFloat(`*String_To_Convert*`)`
`double`	`Double.parseDouble(`*String_To_Convert*`)`

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Multi-Line Output Windows

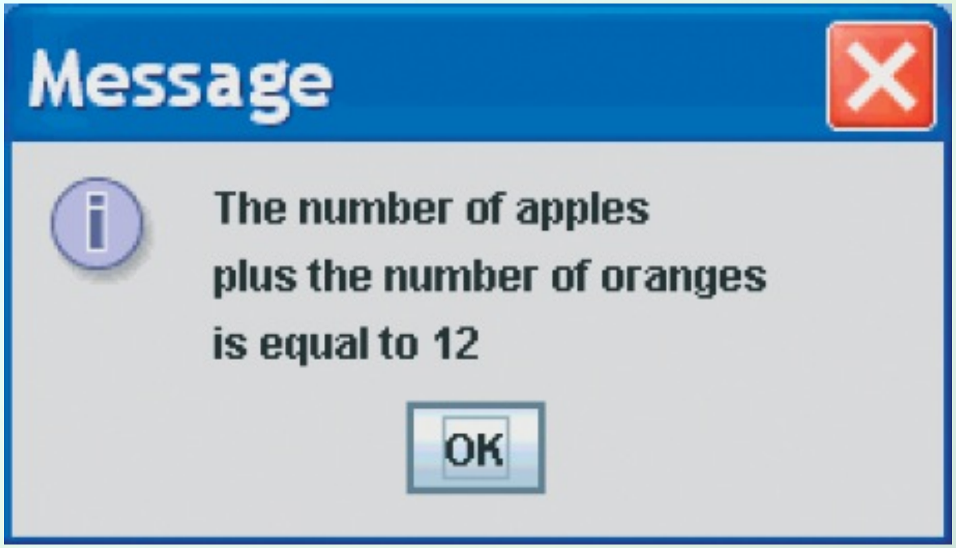
- To output multiple lines using the method **JOptionPane.showMessageDialog**, insert the new line character **'\n'** into the string used as the second argument.

```
OptionPane.showMessageDialog(null,
  "The number of apples\n" +
  "plus the number of oranges\n" +
  "is equal to " + totalFruit);
```


ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Multi-Line Output Windows

- Figure 2.9 A dialog window containing multiline output

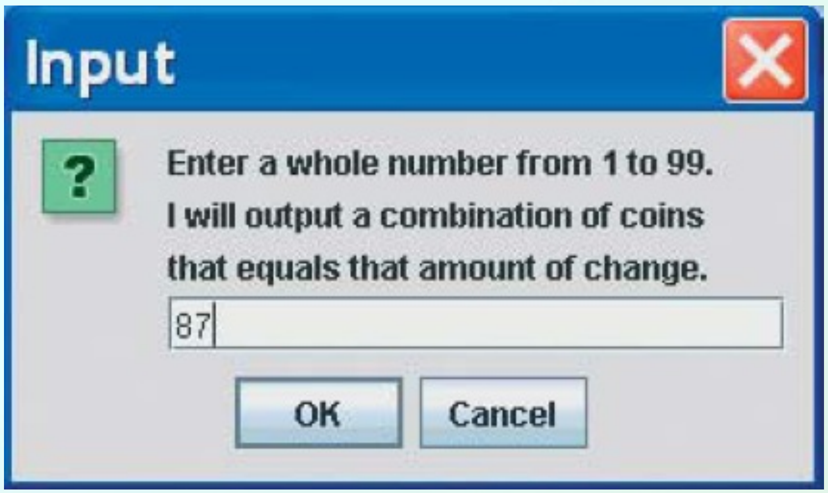


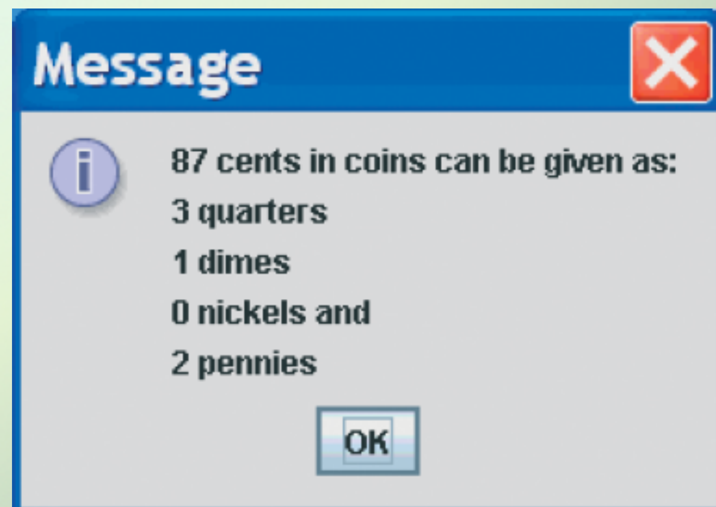
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Programming Example

- View sample program

`class ChangeMakerWindow,` listing 2.12





Sample Screen Output

ISBN 0132162709 © 2012 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 classes, methods, and objects.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Summary

- You have learned about simple keyboard input and screen output.
- You have learned about windows-based input and output using the **`JOptionPane`** class.

ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved