

Origins of the Java Language

- Significance of Java translation process
 - Writing a compiler (translation program) for each type of appliance processor would have been very costly
 - Instead, developed intermediate language that is the same for all types of processors: Java byte-code
 - Therefore, only a small, easy to write program was needed to translate byte-code into the machine code for each processor

Copyright © 2012 Pearson Addison-Wesley.
All rights reserved.

1-4

Introduction To Java

- Most people are familiar with Java as a language for Internet applications
- We will study Java as a general purpose programming language
 - The syntax of expressions and assignments will be similar to that of other high-level languages
 - Details concerning the handling of strings and console output will probably be new

Copyright © 2012 Pearson Addison-Wesley.

1-2

Origins of the Java Language

- Patrick Naughton and Jonathan Payne at Sun Microsystems developed a Web browser that could run programs over the Internet (1994)
 - Beginning of Java's connection to the Internet
 - Original browser evolves into HotJava
- Netscape made its Web browser capable of running Java programs (1995)
 - Other companies follow suit

Copyright © 2012 Pearson Addison-Wesley.

1-

Origins of the Java Language

- Created by Sun Microsystems team led by James Gosling (1991)
- Originally designed for programming home appliances
 - Difficult task because appliances are controlled by a wide variety of computer processors
 - Team developed a two-step translation process to simplify the task of compiler writing for each class of appliances

Objects and Methods

- · Java is an *object-oriented programming (OOP)* language
 - Programming methodology that views a program as consisting of *objects* that interact with one another by means of actions (called *methods*)
 - Objects of the same kind are said to have the same type or be in the same class

Copyright © 2012 Pearson Addison-Wesley.

1-3

Copyright © 2012 Pearson Addison-Wesley.

Terminology Comparisons

- Other high-level languages have constructs called procedures, methods, functions, and/or subprograms
 - These types of constructs are called methods in
 - All programming constructs in Java, including methods, are part of a class

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-7

Java Application Programs

- There are two types of Java programs: applications and applets
- A Java application program or "regular" Java program is a class with a method named main
 - When a Java application program is run, the run-time system automatically invokes the method named main
 - All Java application programs start with the main method

Copyright © 2012 Pearson Addison-Wesley.

1-9

Applets

- A Java applet (little Java application) is a Java program that is meant to be run from a Web browser
 - Can be run from a location on the Internet
 - Can also be run with an applet viewer program for debugging
 - Applets always use a windowing interface
- In contrast, application programs may use a windowing interface or console (i.e., text) I/O

A Sample Java Application Program Display 1.1 A Sample Java Program Name of class (program) The main method public class FirstProgram public static void main(String[] args) System.out.println("Hello reader."); System.out.println("Welcome to Java."); System.out.println("Let's demonstrate a simple calculation."); System.out.println("2 plus 2 is " + answer); SAMPLE DIALOGUE I Welcome to Java Let's demonstrate a simple calculation. 2 plus 2 is 4 Copyright © 2012 Pearson Addison-Wesley. 1-10

System.out.println

- Java programs work by having things called objects perform actions
 - System.out: an object used for sending output to the screen
- The actions performed by an object are called methods
 - println: the method or action that the System.out object performs

Copyright © 2012 Pearson Addison-Wesley.

All rights reserved.

System.out.println

- Invoking or calling a method: When an object performs an action using a method
 - Also called *sending a message* to the object
 - Method invocation syntax (in order): an object, a dot (period), the method name, and a pair of parentheses
 - Arguments: Zero or more pieces of information needed by the method that are placed inside the parentheses

System.out.println("This is an argument");

Copyright © 2012 Pearson Addison-Wesley.

Variable declarations

- Variable declarations in Java are similar to those in other programming languages
 - Simply give the type of the variable followed by its name and a semicolon

int answer;

Copyright © 2012 Pearson Addison-Wesley.

1-13

Using = and +

- · In Java, the equal sign (=) is used as the assignment operator
 - The variable on the left side of the assignment operator is assigned the value of the expression on the right side of the assignment operator

answer = 2 + 2;

- · In Java, the plus sign (+) can be used to denote addition (as above) or *concatenation*
 - Using +, two strings can be connected together

System.out.println("2 plus 2 is " + answer);

Copyright © 2012 Pearson Addison-Wesley.

1-14

Computer Language Levels

- High-level language: A language that people can read, write, and understand
 - A program written in a high-level language must be translated into a language that can be understood by a computer before it can be run
- Machine language: A language that a computer can understand
- Low-level language: Machine language or any language similar to machine language
- Compiler: A program that translates a high-level language program into an equivalent low-level language program
 - This translation process is called compiling

Byte-Code and the Java Virtual Machine

- The compilers for most programming languages translate high-level programs directly into the machine language for a particular computer
 - Since different computers have different machine languages, a different compiler is needed for each one
- In contrast, the Java compiler translates Java programs into byte-code, a machine language for a fictitious computer called the Java Virtual Machine
 - Once compiled to byte-code, a Java program can be used on any computer, making it very portable

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-16

Byte-Code and the Java Virtual Machine

- Interpreter: The program that translates a program written in Java byte-code into the machine language for a particular computer when a Java program is executed
 - The interpreter translates and immediately executes each byte-code instruction, one after another
 - Translating byte-code into machine code is relatively easy compared to the initial compilation step

Copyright © 2012 Pearson Addison-Wesley.

1-17

Program terminology

- · Code: A program or a part of a program
- · Source code (or source program): A program written in a high-level language such as Java
 - The input to the compiler program
- · Object code: The translated low-level program
 - The output from the compiler program, e.g., Java bytecode
 - In the case of Java byte-code, the input to the Java byte-code interpreter

Copyright © 2012 Pearson Addison-Wesley.

1-15

Copyright © 2012 Pearson Addison-Wesley.

Class Loader

- · Java programs are divided into smaller parts called classes
 - Each class definition is normally in a separate file and compiled separately
- · Class Loader: A program that connects the byte-code of the classes needed to run a Java program
 - In other programming languages, the corresponding program is called a *linker*

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-19

Syntax and Semantics

- Syntax: The arrangement of words and punctuations that are legal in a language, the grammar rules of a language
- Semantics: The meaning of things written while following the syntax rules of a language

Copyright © 2012 Pearson Addison-Wesley.

1-22

Compiling a Java Program or Class

- Each class definition must be in a file whose name is the same as the class name followed by . java
 - The class FirstProgram must be in a file named FirstProgram.java
- Each class is compiled with the command javac followed by the name of the file in which the class resides

javac FirstProgram.java

 The result is a byte-code program whose filename is the same as the class name followed by .class

FirstProgram.class

Copyright © 2012 Pearson Addison-Wesley.

Il rights reserved

1-20

Tip: Error Messages

- Bug: A mistake in a program
 - The process of eliminating bugs is called debugging
- Syntax error: A grammatical mistake in a program
 - The compiler can detect these errors, and will output an error message saying what it thinks the error is, and where it thinks the error is

Copyright © 2012 Pearson Addison-Wesley.

1-23

Running a Java Program

- A Java program can be given the run command (java) after all its classes have been compiled
 - Only run the class that contains the main method (the system will automatically load and run the other classes, if any)
- The main method begins with the line: public static void main (String[] args)
- Follow the run command by the name of the class only (no .java or .class extension)
 java FirstProgram

Tip: Error Messages

- Run-time error: An error that is not detected until a program is run
 - The compiler cannot detect these errors: an error message is not generated after compilation, but after execution
- Logic error: A mistake in the underlying algorithm for a program
 - The compiler cannot detect these errors, and no error message is generated after compilation or execution, but the program does not do what it is supposed to do

Copyright © 2012 Pearson Addison-Wesley.

1-21

Copyright © 2012 Pearson Addison-Wesley.

Identifiers

- · *Identifier*: The name of a variable or other item (class, method, object, etc.) defined in a program
 - A Java identifier must not start with a digit, and all the characters must be letters, digits, or the underscore symbol
 - Java identifiers can theoretically be of any length
 - Java is a case-sensitive language: Rate, rate, and RATE are the names of three different variables

Copyright © 2012 Pearson Addison-Wesley.

1-25

Variable Declarations

- Every variable in a Java program must be declared before it is used
 - A variable declaration tells the compiler what kind of data (type) will be stored in the variable
 - The type of the variable is followed by one or more variable names separated by commas, and terminated with a semicolon
 - Variables are typically declared just before they are used or at the start of a block (indicated by an opening brace {)
 - Basic types in Java are called *primitive types*

int numberOfBeans;

double oneWeight, totalWeight;

Copyright © 2012 Pearson Addison-Wesley.

1-28

Identifiers

- Keywords and Reserved words: Identifiers that have a predefined meaning in Java
- Do not use them to name anything else public class void static
- Predefined identifiers: Identifiers that are defined in libraries required by the Java language standard
 - Although they can be redefined, this could be confusing and dangerous if doing so would change their standard meaning

System String println

Copyright © 2012 Pearson Addison-Wesley.

All rights reserved

1-26

1-27

Primitive Types

isplay 1.2	Primitive Types		
TYPE NAME	KIND OF VALUE	MEMORY USED	SIZE RANGE
boolean	true or false	ı byte	not applicable
char	single character (Unicode)	2 bytes	all Unicode characters
byte	integer	ı 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	$-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$
double	floating-point number	8 bytes	±1.76769313486231570 × 10 ⁺³⁰⁸ to ±4.94065645841246544 × 10 ⁻³²⁴

Copyright © 2012 Pearson Addison-Wesley.

All rights reserved

Naming Conventions

- Start the names of variables, classes, methods, and objects with a lowercase letter, indicate "word" boundaries with an uppercase letter, and restrict the remaining characters to digits and lowercase letters topSpeed bankRate1 timeOfArrival
- Start the names of classes with an uppercase letter and, otherwise, adhere to the rules above

FirstProgram MyClass String

the right side of the operator
Variable = Expression;

 An expression consists of a variable, number, or mix of variables, numbers, operators, and/or method invocations

- The equal sign (=) is used as the assignment operator

- An assignment statement consists of a variable on

the left side of the operator, and an expression on

Assignment Statements With Primitive Types

In Java, the assignment statement is used to

change the value of a variable

temperature = 98.6;

count = numberOfBeans;

Copyright © 2012 Pearson Addison-Wesley.

Assignment Statements With Primitive Types

- When an assignment statement is executed, the expression is first evaluated, and then the variable on the left-hand side of the equal sign is set equal to the value of the expression

distance = rate * time;

 Note that a variable can occur on both sides of the assignment operator

count = count + 2;

The assignment operator is automatically executed from right-to-left, so assignment statements can be chained number2 = number1 = 3;

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-31

Tip: Initialize Variables

- A variable that has been declared but that has not yet been given a value by some means is said to be uninitialized
- In certain cases an uninitialized variable is given a default value
 - It is best not to rely on this
 - Explicitly initialized variables have the added benefit of improving program clarity

Copyright © 2012 Pearson Addison-Wesley.

Tip: Initialize Variables

The declaration of a variable can be combined with its initialization via an assignment statement

```
int count = 0;
```

double distance = 55 * .5;

char grade = 'A';

 Note that some variables can be initialized and others can remain uninitialized in the same declaration

int initialCount = 50, finalCount;

Shorthand Assignment Statements

- Shorthand assignment notation combines the assignment operator (=) and an arithmetic operator
- It is used to change the value of a variable by adding, subtracting, multiplying, or dividing by a specified value
- The general form is

Variable Op = Expression

which is equivalent to

Variable = Variable Op (Expression)

- The Expression can be another variable, a constant, or a more complicated expression
- Some examples of what Op can be are +, -, *, /, or %

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-34

Shorthand Assignment Statomonto

Statements			
Example:	Equivalent To:		
count += 2;	count = count + 2;		
<pre>sum -= discount;</pre>	<pre>sum = sum - discount;</pre>		
bonus *= 2;	bonus = bonus * 2;		
time /=	time =		
rushFactor;	<pre>time / rushFactor;</pre>		
change %= 100;	change = change % 100;		
amount *=	<pre>amount = amount *</pre>		
<pre>count1 + count2;</pre>	(count1 + count2);		
Copyright © 2012 Pearson	1-35		

Assignment Compatibility

In general, the value of one type cannot be stored in a variable of another type

int intVariable = 2.99; //Illegal

Addison-Wesley. All rights reserved.

- The above example results in a type mismatch because a double value cannot be stored in an int variable
- However, there are exceptions to this double doubleVariable = 2;
 - For example, an int value can be stored in a double type

Copyright © 2012 Pearson Addison-Wesley.

Assignment Compatibility

 More generally, a value of any type in the following list can be assigned to a variable of any type that appears to the right of it

 $\texttt{byte} {\rightarrow} \texttt{short} {\rightarrow} \texttt{int} {\rightarrow} \texttt{long} {\rightarrow} \texttt{float} {\rightarrow} \texttt{double}$

char

- Note that as your move down the list from left to right, the range of allowed values for the types becomes larger
- An explicit type cast is required to assign a value of one type to a variable whose type appears to the left of it on the above list (e.g., double to int)
- Note that in Java an int cannot be assigned to a variable of type boolean, nor can a boolean be assigned to a variable of type int

Copyright © 2012 Pearson Addison-Wesley.
All rights reserved.

1-37

Arithmetic Operators and Expressions

- As in most languages, expressions can be formed in Java using variables, constants, and arithmetic operators
 - These operators are + (addition), (subtraction),
 * (multiplication), / (division), and % (modulo, remainder)
 - An expression can be used anyplace it is legal to use a value of the type produced by the expression

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-40

Constants

- · Constant (or literal): An item in Java which has one specific value that cannot change
 - Constants of an integer type may not be written with a decimal point (e.g., 10)
 - Constants of a floating-point type can be written in ordinary decimal fraction form (e.g., 367000.0 or 0.000589)
 - Constant of a floating-point type can also be written in scientific (or floating-point) notation (e.g., 3.67e5 or 5.89e-4)
 - · Note that the number before the e may contain a decimal point, but the number after the e may not

Copyright © 2012 Pearson Addison-Wesley.

Il rights reserved

1-38

Arithmetic Operators and Expressions

- If an arithmetic operator is combined with int operands, then the resulting type is int
- If an arithmetic operator is combined with one or two double operands, then the resulting type is double
- If different types are combined in an expression, then the resulting type is the right-most type on the following list that is found within the expression

byte-short-int-long-float-double

char

 Exception: If the type produced should be byte or short (according to the rules above), then the type produced will actually be an int

Copyright © 2012 Pearson Addison-Wesley.

1-41

Constants

- Constants of type char are expressed by placing a single character in single quotes (e.g., 'Z')
- Constants for strings of characters are enclosed by double quotes (e.g., "Welcome to Java")
- There are only two boolean type constants, true and false
 - Note that they must be spelled with all lowercase letters

Parentheses and Precedence Rules

- An expression can be fully parenthesized in order to specify exactly what subexpressions are combined with each operator
- If some or all of the parentheses in an expression are omitted, Java will follow precedence rules to determine, in effect, where to place them
 - However, it's best (and sometimes necessary) to include them

Precedence Rules

Display 1.3 Precedence Rules

Highest Precedence

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

Lowest Precedence

Copyright © 2012 Pearson Addison-Wesley.

1-43

Pitfall: Round-Off Errors in Floating-Point Numbers

- Floating point numbers are only approximate quantities
 - Mathematically, the floating-point number 1.0/3.0 is equal to 0.3333333 . . .
 - A computer has a finite amount of storage space
 - · It may store 1.0/3.0 as something like 0.333333333, which is slightly smaller than one-third
 - Computers actually store numbers in binary notation, but the consequences are the same: floating-point numbers may lose accuracy

Copyright © 2012 Pearson Addison-Wesley.

1-46

Precedence and Associativity Rules

 When the order of two adjacent operations must be determined, the operation of higher precedence (and its apparent arguments) is grouped before the operation of lower precedence

base + rate * hours is evaluated as

base + (rate * hours)

 When two operations have equal precedence, the order of operations is determined by associativity rules

Copyright © 2012 Pearson Addison-Wesley.

l rights reserved

1_///

Integer and Floating-Point Division

When one or both operands are a floating-point type, division results in a floating-point type

15.0/2 evaluates to 7.5

- When both operands are integer types, division results in an integer type
 - Any fractional part is discarded
 - The number is not rounded

15/2 evaluates to 7

 Be careful to make at least one of the operands a floatingpoint type if the fractional portion is needed

Copyright © 2012 Pearson Addison-Wesley.

1-47

Precedence and Associativity Rules

 Unary operators of equal precedence are grouped right-toleft

+-+rate is evaluated as + (- (+rate))

 Binary operators of equal precedence are grouped left-toright

base + rate + hours is evaluated as

(base + rate) + hours

Exception: A string of assignment operators is grouped right-to-left

n1 = n2 = n3; is evaluated as n1 = (n2 = n3);

Copyright © 2012 Pearson Addison-Wesley.

1-45

The % Operator

 The % operator is used with operands of type int to recover the information lost after performing integer division

15/2 evaluates to the quotient 7

15%2 evaluates to the remainder 1

- The % operator can be used to count by 2's, 3's, or any other number
 - To count by twos, perform the operation number % 2, and when the result is 0, number is even

Copyright © 2012 Pearson Addison-Wesley.

Type Casting

- A type cast takes a value of one type and produces a value of another type with an "equivalent" value
 - If ${\bf n}$ and ${\bf m}$ are integers to be divided, and the fractional portion of the result must be preserved, at least one of the two must be type cast to a floating-point type before the division operation is performed

double ans = n / (double)m;

- Note that the desired type is placed inside parentheses immediately in front of the variable to be cast
- Note also that the type and value of the variable to be cast does not change

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-49

Increment and Decrement Operators

- When either operator precedes its variable, and is part of an expression, then the expression is evaluated using the changed value of the variable
 - If n is equal to 2, then 2* (++n) evaluates to 6
- When either operator follows its variable, and is part of an expression, then the expression is evaluated using the original value of the variable, and only then is the variable value changed
 - If n is equal to 2, then 2* (n++) evaluates to 4

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-52

More Details About Type Casting

- When type casting from a floating-point to an integer type, the number is truncated, not rounded
 - (int) 2.9 evaluates to 2, not 3
- When the value of an integer type is assigned to a variable of a floating-point type, Java performs an automatic type cast called a type coercion

double d = 5;

In contrast, it is illegal to place a double value into an int variable without an explicit type cast

int i = 5.5; // Illegal

int i = (int)5.5 // Correct

Copyright © 2012 Pearson Addison-Wesley.

The Class String

- There is no primitive type for strings in Java
- The class String is a predefined class in Java that is used to store and process strings
- Objects of type **String** are made up of strings of characters that are written within double quotes
- Any quoted string is a constant of type String "Live long and prosper."
- A variable of type String can be given the value of a String object

String blessing = "Live long and prosper.";

Copyright © 2012 Pearson Addison-Wesley.

Increment and Decrement Operators

- · The increment operator (++) adds one to the value of a variable
 - If n is equal to 2, then n++ or ++n will change the value of n to 3
- · The decrement operator (--) subtracts one from the value of a variable
 - If n is equal to 4, then n-- or --n will change the value of n to 3

Concatenation of Strings

- Concatenation: Using the + operator on two strings in order to connect them to form one longer string
 - If greeting is equal to "Hello ", and javaClass is equal to "class", then greeting + javaClass is equal to "Hello class"
- Any number of strings can be concatenated together
- When a string is combined with almost any other type of item, the result is a string
 - "The answer is " + 42 evaluates to "The answer is 42"

Copyright © 2012 Pearson Addison-Wesley.

1-51

Copyright © 2012 Pearson Addison-Wesley.

Classes, Objects, and Methods

- A class is the name for a type whose values are objects
- Objects are entities that store data and take actions
 - Objects of the String class store data consisting of strings of characters
- · The actions that an object can take are called methods
 - Methods can return a value of a single type and/or perform an action
 - All objects within a class have the same methods, but each can have different data values

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-55

Some Methods in the Class String (Part 1 of 8) Display 1.4 Some Methods in the Class String int length() Returns the length of the calling object (which is a string) as a value of type int EXAMPLE After program executes String greeting = "Hello!"; greeting.length() returns 6. boolean equals (Other_String) Returns true if the calling object string and the Other_String are equal. Otherwise, returns false. After program executes String greeting = "Hello"; greeting.equals("Hello") returns true greeting.equals("Good-Bye") returns false greeting.equals("hello") returns false Note that case matters. "Hello" and "hello" are not equal because one starts with an uppercase letter and the other starts with a lowercase letter. Copyright © 2012 Pearson Addison-Wesley. 1-58 All rights reserved.

Classes, Objects, and Methods

- Invoking or calling a method: a method is called into action by writing the name of the calling object, followed by a dot, followed by the method name, followed by parentheses
 - This is sometimes referred to as sending a message to the obiect
 - The parentheses contain the information (if any) needed by the method
 - This information is called an argument (or arguments)

Copyright © 2012 Pearson Addison-Wesley.

String Methods

- The String class contains many useful methods for stringprocessing applications
 - A String method is called by writing a String object, a dot, the name of the method, and a pair of parentheses to enclose any
- If a String method returns a value, then it can be placed anywhere that a value of its type can be used
 String greeting = "Hello";

int count = greeting.length();

System.out.println("Length is " + greeting.length());

Always count from zero when referring to the position or index of a character in a string

Copyright © 2012 Pearson Addison-Wesley.

1-57

Some Methods in the Class String (Part 2 of 8)

Display 1.4 Some Methods in the Class String

boolean equalsIgnoreCase(Other_String)

Returns true if the calling object string and the Other_String are equal, considering uppercase and lowercase versions of a letter to be the same. Otherwise, returns false.

After program executes String name = "mary!" greeting.equalsIgnoreCase("Mary!") returns true

String toLowerCase()

Returns a string with the same characters as the calling object string, but with all letter characters converted to lowercase.

EXAMPLE

After program executes String greeting = "Hi Mary!"; greeting.toLowerCase() returns "hi mary!"

(continued)

Copyright © 2012 Pearson Addison-Wesley.

Some Methods in the Class String (Part 3 of 8)

Display 1.4 Some Methods in the Class String

String toUpperCase()

Returns a string with the same characters as the calling object string, but with all letter characters converted to uppercase.

After program executes String greeting = "Hi Mary!"; greeting.toUpperCase() returns "HI MARY!".

Returns a string with the same characters as the calling object string, but with leading and trailing white space removed. Whitespace characters are the characters that print as white space on paper, such as the blank (space) character, the tab character, and the new-line character '\n'

EXAMPLE

After program executes String pause = " Hmm "; pause.trim() returns "Hmm".

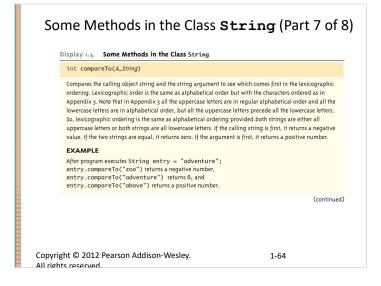
(continued)

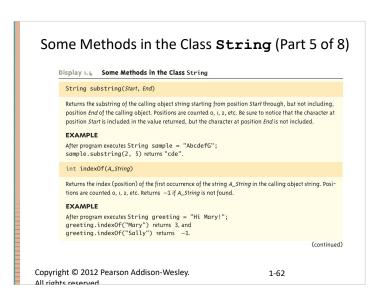
1-60

Copyright © 2012 Pearson Addison-Wesley.

Some Methods in the Class String (Part 4 of 8) Display 1.4. Some Methods in the Class String char charAt(Position) Returns the character in the calling object string at the Position. Positions are counted 0, 1, 2, etc. EXAMPLE After program executes String greeting = "Hello!"; greeting.charAt(0) returns 'H', and greeting.charAt(1) returns 'e'. String substring(Start) Returns the substring of the calling object string starting from Start through to the end of the calling object. Positions are counted 0, 1, 2, etc. Be sure to notice that the character at position Start is included in the value returned. EXAMPLE After program executes String sample = "AbcdefG"; sample.substring(2) returns "cdefG". (continued)

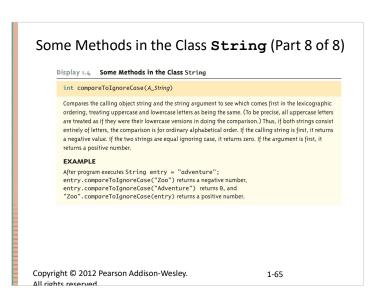
1-61

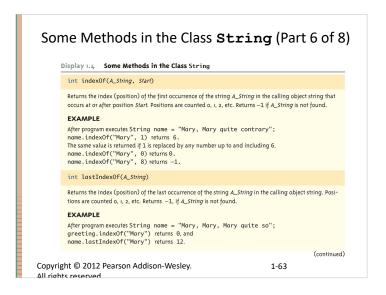


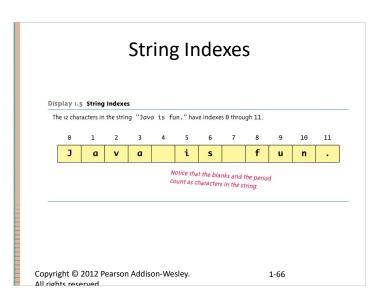


Copyright © 2012 Pearson Addison-Wesley.

All rights reserved.







Escape Sequences

- · A backslash (\) immediately preceding a character (i.e., without any space) denotes an escape sequence or an escape character
 - The character following the backslash does not have its usual meaning
 - Although it is formed using two symbols, it is regarded as a single character

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-67

Character Sets

- ASCII: A character set used by many programming languages that contains all the characters normally used on an Englishlanguage keyboard, plus a few special characters
 - Each character is represented by a particular number
- Unicode: A character set used by the Java language that includes all the ASCII characters plus many of the characters used in languages with a different alphabet from English

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

1-70

Escape Sequences

Display 1.6 Escape Sequences

- \" 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. White space up to the next tab stop.

Copyright © 2012 Pearson Addison-Wesley.

Naming Constants

Instead of using "anonymous" numbers in a program, always declare them as named constants, and use their name instead public static final int INCHES_PER_FOOT = 12;

public static final double RATE = 0.14;

- This prevents a value from being changed inadvertently
- It has the added advantage that when a value must be modified, it need only be changed in one place
- Note the naming convention for constants: Use all uppercase letters, and designate word boundaries with an underscore character

Copyright © 2012 Pearson Addison-Wesley.

String Processing

- A String object in Java is considered to be immutable, i.e., the characters it contains cannot be changed
- There is another class in Java called StringBuffer that has methods for editing its string objects
- However, it is possible to change the value of a String variable by using an assignment statement

String name = "Soprano";

name = "Anthony " + name;

Comments

- A line comment begins with the symbols //, and causes the compiler to ignore the remainder of the line
 - This type of comment is used for the code writer or for a programmer who modifies the code
- A block comment begins with the symbol pair /*, and ends with the symbol pair */
 - The compiler ignores anything in between
 - This type of comment can span several lines
 - This type of comment provides documentation for the users of the program

Copyright © 2012 Pearson Addison-Wesley

1-69

Copyright © 2012 Pearson Addison-Wesley.

Program Documentation

- Java comes with a program called javadoc that will automatically extract documentation from block comments in the classes you define
 - As long as their opening has an extra asterisk (/**)
- Ultimately, a well written program is selfdocumenting
 - Its structure is made clear by the choice of identifier names and the indenting pattern
 - When one structure is nested inside another, the inside structure is indented one more level

Copyright © 2012 Pearson Addison-Wesley.

1-73

Comments and a Named Constant