Elementary Programming
رنيتج يسرمبي

By: Mamoun Nawahdah (PhD)
2020

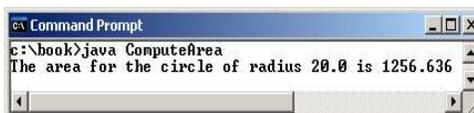# Trace a Program Execution

```
public class ComputeArea {
 /** Main method */
 public static void main(String[] args) {
  double radius;
  double area;

  // Assign a radius
  radius = 20;

  // Compute area
  area = radius * radius * 3.14159;

  // Display results
  System.out.println("The area for the circle of radius " +
   radius + " is " + area);
 }
}
```

memory

| radius | 20 |
| area | 1256.636 |

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

2

Identifiers

❖ An **identifier** is a sequence of characters that consist of letters, digits, underscores (_), and dollar signs (**$**).

❖ An identifier must start with a letter, an underscore (_), or a dollar sign (**$**). It cannot start with a digit.

  ▪ An identifier cannot be a reserved word. (See **Appendix A**, "Java Keywords").

❖ An identifier cannot be **true**, **false**, or **null**.

❖ An identifier can be of any length.

3

Declaring Variables

int x;   // Declare x to be an integer variable

double radius;   // Declare radius to be a double variable

char a;   // Declare a to be a character variable

Assignment Statements

   x = 1;   // Assign 1 to x

   radius = 1.0;   // Assign 1.0 to radius

   a = 'A';   // Assign 'A' to a

4

Declaring and Initializing in One Step

int x = 1;

double d = 1.4;

Named Constants

final datatype CONSTANTNAME = VALUE;

final double PI = 3.14159;

final int SIZE = 3;

5

Naming Conventions

❖ Choose **meaningful** and descriptive names.

❖ Variables and method names:
- Use lowercase.
- If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name.
- For example, the variables **radius** and **area**, and the method **computeArea**.

6

Naming Conventions, cont.

❖ **Class names:**
- Capitalize the first letter of each word in the name.
- For example, the class name **ComputeArea**.

❖ **Constants**:
- Capitalize all letters in constants, and use underscores to connect words.
- For example, the constant **PI** and **MAX_VALUE**

7

Numerical Data Types

Name	Range	Storage Size
byte	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed
short	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
int	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
long	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
float	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
double	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

8

double vs. float

The double type values are more accurate than the float type values. For example,

```
System.out.println("1.0 / 3.0 is " + 1.0 / 3.0);
```

displays 1.0 / 3.0 is 0.3333333333333333

16 digits

```
System.out.println("1.0F / 3.0F is " + 1.0F / 3.0F);
```

displays 1.0F / 3.0F is 0.33333334

7 digits

9

Increment and Decrement Operators

Operator	Name	Description	Example (assume i = 1)
++var	preincrement	Increment var by 1, and use the new var value in the statement	int j = ++i; // j is 2, i is 2
var++	postincrement	Increment var by 1, but use the original var value in the statement	int j = i++; // j is 1, i is 2
--var	predecrement	Decrement var by 1, and use the new var value in the statement	int j = --i; // j is 0, i is 0
var--	postdecrement	Decrement var by 1, and use the original var value in the statement	int j = i--; // j is 1, i is 0

10

5

Numeric Type Conversion

Consider the following statements:

byte i = 100;

long k = i * 3 + 4;

double d = i * 3.1 + k / 2;

11

Conversion Rules

❖ When performing a binary operation involving two operands of different types, Java automatically converts the operand based on the following rules:

1. If one of the operands is **double**, the other is converted into double.

2. Otherwise, if one of the operands is **float**, the other is converted into float.

3. Otherwise, if one of the operands is **long**, the other is converted into long.

4. Otherwise, both operands are converted into **int**.

12

Type Casting

Implicit casting

double d = 3;   (type widening)

Explicit casting

int i = (int)3.0;   (type narrowing)
int i = (int)3.9;   (Fraction part is truncated)

What is wrong?   int x = 6 / 2.0;

range increases

→

byte, short, int, long, float, double

13

Character Data Type

char letter = 'A';   (ASCII)
char numChar = '4';   (ASCII)

char letter = '**\u**0041';   (Unicode)

char numChar = '**\u**0034'; (Unicode)

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

char ch = 'a';

System.out.println(++ch);

14

The **S**tring Type

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

String message = "Welcome to Java!";

❖ String is actually a predefined class in the Java library.

❖ The String type is not a primitive type. It is known as a *reference* type.

15

String Concatenation

// Three strings are concatenated

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

// String Chapter is concatenated with number 2
String s = "Chapter" + 2; // s becomes **Chapter2**

// String Supplement is concatenated with character B
String s1 = "Supplement" + 'B'; // s1 becomes **SupplementB**

16

Console Input

❖ You can use the **Scanner** class for console input.

❖ Java uses **System.in** to refer to the standard input device (i.e. Keyboard).

```java
import  java.util.Scanner;
public class Test{
   public static void main(String[] s){
      Scanner input = new Scanner(System.in);
      System.out.println("Enter  X : ");
       int x = input.nextInt();
       System.out.println("You entered: "+ x);
   }
}
```

17

Reading Numbers from the Keyboard

Scanner input = new Scanner(System.in);
int value = input.nextInt();

Method	Description
nextByte()	reads an integer of the byte type.
nextShort()	reads an integer of the short type.
nextInt()	reads an integer of the int type.
nextLong()	reads an integer of the long type.
nextFloat()	reads a number of the float type.
nextDouble()	reads a number of the double type.

18

9