جامعة الأميرة نورة بنت عبد الرحمن
Princess Nora Bint Abdul Rahman University

CS110: PROGRAMMING LANGUAGE I

Computer Science Department

Lecture 4: Java Basics (II)

A java Program

Class in file .java

class keyword

braces { , } delimit a class body

main Method

// indicates a comment.

white space

Java is case sensitive

```
1  // Fig. 2.1: Welcome1.java
2  // Text-printing program.
3
4  public class Welcome1
5  {
6     // main method begins execution of Java application
7     public static void main( String[] args )
8     {
9        System.out.println( "Welcome to Java Programming!" );
10    } // end method main
11 } // end class Welcome1
```

```
Welcome to Java Programming!
```

Fig. 2.1 | Text-printing program.

"Everything must be in a class"

There are no global functions or global data.

Lecture Contents:

- Java language Basics
  - Memory, variables, and Data types
  - Casting
  - Input
  - constants
- Case studys
- Program Style

Java Basics: Memory & variables

- Variables
  - A memory location to store data for a program
  - Every variable has a name, a type, a size (in bytes) and a value.
  - The name can be any valid identifier.
  - A variable's type specifies what kind of information is stored at that location in memory.
  - Must declare all data before use in program
  - When a new value is placed into a variable, the new value replaces the previous value (if any) → The previous value is lost.

Fig. 2.8 | Memory location showing the name and value of variable `number1`.

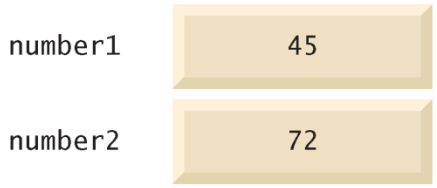

Fig. 2.9 | Memory locations after storing values for `number1` and `number2`.

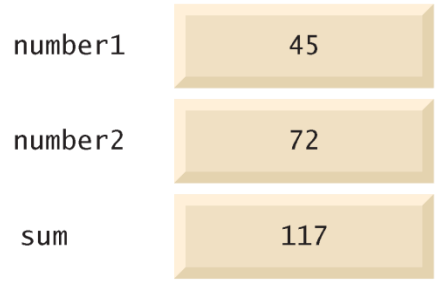

Fig. 2.10 | Memory locations after storing the sum of `number1` and `number2`.

Declaring Variables

- Rule:

```
<dataType> identifier;
```

- Examples:

```
int myValue;
char response;
float price;
double yCoord;
```

Data Types

Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED	SIZE RANGE	PRECISION
`short` (also called `short int`)	2 bytes	−32,768 to 32,767	Not applicable
`int`	4 bytes	−2,147,483,648 to 2,147,483,647	Not applicable
`long` (also called `long int`)	4 bytes	−2,147,483,648 to 2,147,483,647	Not applicable
`float`	4 bytes	approximately 10^{-38} to 10^{38}	7 digits
`double`	8 bytes	approximately 10^{-308} to 10^{308}	15 digits

Data Types:

`long double`	10 bytes	approximately 10^{-4932} to 10^{4932}	19 digits
`char`	1 byte	All ASCII characters (Can also be used as an integer type, although we do not recommend doing so.)	Not applicable
`bool`	1 byte	`true`, `false`	Not applicable

The values listed here are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. *Precision* refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types `float`, `double`, and `long double` are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.

Java operation	Operator	Algebraic expression	Java expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Fig. 2.11 | Arithmetic operators.

The arithmetic operators are binary operators because they each operate on two operands.

Java basics : expressions

The asterisk (*) indicates multiplication

The percent sign (%) is the remainder operator

The remainder operator, %, yields the remainder after division.

Java basics : Assigning Data

- Initializing data in declaration statement
  - Results "undefined" if you don't!

```
int myValue = 0;
```

- Assigning data during execution
  - Lvalues (left-side) & Rvalues (right-side)
    - Lvalues must be variables
    - Rvalues can be any expression
      - Example:

```
distance = rate * time;
Lvalue:  "distance"
Rvalue: "rate * time"
```

Values of the Variables	Variables	Statement/Explanation
Before Statement 1	num1: ? num2: ? num3: ?	
After Statement 1	num1: 18 num2: ? num3: ?	`num1 = 18;` Store 18 into num1.
After Statement 2	num1: 45 num2: ? num3: ?	`num1 = num1 + 27;` num1 + 27 = 18 + 27 = 45. This value is assigned to num1, which replaces the old value of num1.
After Statement 3	num1: 45 num2: 45 num3: ?	`num2 = num1;` Copy the value of num1 into num2.
After Statement 4	num1: 45 num2: 45 num3: 9	`num3 = num2 / 5;` num2 / 5 = 45 / 5 = 9. This value is assigned to num3. So num3 = 9.
After Statement 5	num1: 45 num2: 45 num3: 2	`num3 = num3 / 4;` num3 / 4 = 9 / 4 = 2. This value is assigned to num3, which replaces the old value of num3.

Data Assignment Rules

- Compatibility of Data Assignments
  - `intVar = 2.99;  // 2 is assigned to intVar!`
    - Only integer part "fits", so that's all that goes
    - Called "implicit" or "automatic type conversion"
  - Literals
    - 2, 5.75, "Z", "Hello World"
    - Considered "constants": can't change in program
  - Type mismatches
    - General Rule: Cannot place value of one type into variable of another type

Shorthand Operators

- Increment & Decrement Operators
  - Increment operator, ++
    `intVar++;` → `intVar = intVar + 1;`
  - Decrement operator, --
    `intVar--;` → `intVar = intVar - 1;`
- Post-Increment : intVar++
  - Uses current value of variable, THEN increments it
- Pre-Increment : ++intVar
  - Increments variable first, THEN uses new value
- No difference if "alone" in statement:
  intVar++; and ++intVar; → identical result

Example

Post-Increment

```
int n=2,res;
res = 2 * (n++);
/*what is the
value of res and
n?*/
```

Pre-Increment

```
int n = 2, res;
res = 2 * (++n);
 /*what is the
value of res and
n?*/
```

Arithmetic Assignment Operators

EXAMPLE	EQUIVALENT TO
`count += 2;`	`count = count + 2;`
`total -= discount;`	`total = total - discount;`
`bonus *= 2;`	`bonus = bonus * 2;`
`time /= rushFactor;`	`time = time/rushFactor;`
`change %= 100;`	`change = change % 100;`
`amount *= cnt1 + cnt2;`	`amount = amount * (cnt1 + cnt2);`

```
int i = 10;
int newNum = 10 * i++;
System.out.print("i is " + I + ",
newNum is " + newNum);
```

```
int i = 10;
int newNum = 10 * (++i);
System.out.print("i is " + I + ",
newNum is " + newNum);
```


Program output

Arithmetic Precision

- Precision of Calculations
  - "Highest-order operand" determines type of arithmetic "precision" performed
- Examples:
  - 17 / 5 evaluates to ??
    - Both operands are integers
    - Integer division is performed!
  - 17.0 / 5 equals ??
    - Highest-order operand is "double type"
    - Double "precision" division is performed!
  - ```
    int intVar1 =1, intVar2=2;
    intVar1 / intVar2;
    ```
    - Result: ??


    ```

Type Casting

- Casting for Variables
  - Can add ".0" to literals to force precision arithmetic, but what about variables?
    - We can't use "myInt.0"!
- Two types
  - Implicit—also called "Automatic"
    - Done FOR you, automatically
      17 / 5.5
  - Explicit type conversion
    - Programmer specifies conversion with cast operator

```
(double)17 / 5.5
(double)myInt / myDouble
```

    - Explicitly "casts" or "converts" intVar to double type
    - Result of conversion is then used

Java basics : input


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

- `Scanner`
  - Enables a program to read data for use in a program.
  - Data can come from many sources, such as the user at the keyboard or a file on disk.
  - Before using a `Scanner`, you must create it and specify the source of the data.
  - Standard input object, `System.in`, enables applications to read bytes of information typed by the user.
  - translates these key strokes into types that can be used in a program.

Java basics : input

- use : `import java.util.Scanner;`
- Define an object of the Scanner class:

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

- input values:

```
num1 = input.nextInt();
```

- Display after calculation:

```
System.out.printf("the square is : %d
  \n", num1*num1);
```

Printf Conversion-Characters

type	*code*	*typical literal*	*sample format strings*	*converted string values for output*
int	d	512	"%14d" "%-14d"	" 512" "512 "
double	f	1595.1680010754388	"%14.2f" "%.7f"	" 1595.17" "1595.1680011"
	e		"%14.4e"	" 1.5952e+03"
String	s	"Hello, World"	"%14s" "%-14s" "%-14.5s"	" Hello, World" "Hello, World " "Hello "

Case study: Adding numbers

- Write a program that computes and displays the sum of two integer numbers.

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

```
1  // Fig. 2.7: Addition.java
2  // Addition program that displays the sum of two numbers.
3  import java.util.Scanner; // program uses class Scanner
4
5  public class Addition
6  {
7     // main method begins execution of Java application
8     public static void main( String[] args )
9     {
10       // create a Scanner to obtain input from the command window
11       Scanner input = new Scanner( System.in );
12
13       int number1; // first number to add
14       int number2; // second number to add
15       int sum; // sum of number1 and number2
16
17       System.out.print( "Enter first integer: " ); // prompt
18       number1 = input.nextInt(); // read first number from user
19
20       System.out.print( "Enter second integer: " ); // prompt
21       number2 = input.nextInt(); // read second number from user
22
23       sum = number1 + number2; // add numbers, then store total in sum
24
25       System.out.printf( "Sum is %d\n", sum ); // display sum
26    } // end method main
27 } // end class Addition
```

Imports class `Scanner` for use in this program

Creates `Scanner` for reading data from the user

Variables that are declared but not initialized

Reads an `int` value from the user

Reads another `int` value from the user

Sums the values of `number1` and `number2`

Fig. 2.7 | Addition program that displays the sum of two numbers. (Part 1 of 2.)

Arithmetic Expressions

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

is translated to

(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)

Constants

- A “constant variable” is an identifier that is similar to a variable except that it holds one value for its entire existence
- Why constants:
  - give names to otherwise unclear literal values
  - facilitate changes to the code
  - prevent inadvertent errors
- In Java:

```
final double PI = 3.14159265;
```

Example

PROGRAMMING EXAMPLE: Convert Length

Write a program that takes as input given lengths expressed in feet and inches. The program should then convert and output the lengths in centimeters. Assume that the lengths given in feet and inches are integers.

Input: Length in feet and inches

Output: Equivalent length in centimeters

Sample Run: (In this sample run, the user input is shaded.)

```
Enter feet: 15

Enter inches: 7

The numbers you entered are 15 for feet and 7 for inches.

The total number of inches = 187
The number of centimeters = 474.98
```

PROBLEM ANALYSIS AND ALGORITHM DESIGN

The lengths are given in feet and inches, and you need to find the equivalent length in centimeters. One inch is equal to `2.54` centimeters. The first thing the program needs to do is convert the length given in feet and inches to all inches. To convert the length from feet and inches to inches, you multiply the number of feet by `12` (1 foot is equal to `12` inches), and add your answer to the given inches. Then you can use the conversion formula, `1` inch = `2.54` centimeters, to find the equivalent length in centimeters.

Suppose the input is `5` feet and `7` inches. You find the total inches as follows:

```
totalInches = (12 * feet) + inches
            = 12 * 5 + 7
            = 67
```

You can then apply the conversion formula, `1` inch = `2.54` centimeters, to find the length in centimeters.

```
centimeters = totalInches * 2.54
            = 67 * 2.54
            = 170.18
```

Based on this analysis, you can design an algorithm as follows:

1. Get the length in feet and inches.
2. Convert the length into total inches.
3. Convert total inches into centimeters.
4. Output centimeters.

VARIABLES

The input for the program is two numbers: one for feet and one for inches. Thus, you need two variables: one to store feet and the other to store inches. Because the program will first convert the given length into inches, you need a third variable to store the total inches. You need a fourth variable to store the equivalent length in centimeters. In summary, you need the following variables:

```
int feet;            //variable to store feet
int inches;          //variable to store inches
int totalInches;     //variable to store total inches

double centimeters; //variable to store length in centimeters
```

NAMED CONSTANTS

Recall that to calculate the equivalent length in centimeters, you need to multiply the total inches by `2.54`. Instead of using the value `2.54` directly in the program, you will declare this value as a named constant. Similarly, to find the total inches, you need to multiply the feet by `12` and add the inches. Instead of using `12` directly in the program, you will also declare this value as a named constant. Using named constants makes it easier to modify the program later. Because the named constants will be placed before the method `main`, you must use the modifier `static` to declare these named constants (see the earlier section, Creating a Java Application Program).

```
static final double CENTIMETERS_PER_INCH = 2.54;
static final int INCHES_PER_FOOT = 12;
```

COMPLETE PROGRAM LISTING

```
//*************************************************************
// Author: D. S. Malik
//
// Program Convert: This program converts measurements
// in feet and inches into centimeters using the formula
// that 1 inch is equal to 2.54 centimeters.
//*************************************************************

import java.util.*;

public class Conversion
{
   static Scanner console = new Scanner(System.in);

   static final double CENTIMETERS_PER_INCH = 2.54;
   static final int INCHES_PER_FOOT = 12;
   public static void main(String[] args)
   {
         //declare variables
      int feet;
      int inches;
      int totalInches;

      double centimeters;

      System.out.print("Enter feet: ");                    //Step 1
      feet = console.nextInt();                           //Step 2
```

```
        System.out.println();
        System.out.print("Enter inches: ");                  //Step 3
        inches = console.nextInt();                          //Step 4
        System.out.println();
        System.out.println("The numbers you entered are "
                           + feet + " for feet and "
                           + inches + " for inches.");       //Step 5

        totalInches = INCHES_PER_FOOT * feet + inches;       //Step 6

        System.out.println();
        System.out.println("The total number of inches = "
                           + totalInches);                   //Step 7

        centimeters = totalInches * CENTIMETERS_PER_INCH;    //Step 8

        System.out.println("The number of centimeters = "
                           + centimeters);                   //Step 9
    }
}
```

Debugging

- errors are called bugs.
- When you type a program, typos and unintentional syntax errors are likely to occur.
- Therefore, when you compile a program, the compiler will identify the syntax errors.
- Debugging means to identify and fix syntax errors.

```
1.  import java.util.*;
2.
3.  public class ProgramNum1
4.  {
5.     static Scanner console = new Scanner(System.in);
6.
7.     public static void main(String[ ] args)
8.     {
9.        int num
10.
11.       num = 18;
12.
13.       tempNum = 2 * num;
14.
15.       System.out.println("Num = " + num + ", tempNum = " - tempNum);
16.    }
```


Example

Find and fix error!!

Program Style

- Always use comments…
  - Make programs easy to read and modify
  - // Two slashes indicate entire line is to be ignored
  - /*Delimiters indicates everything between is ignored*/
- Identifier naming
  - ALL_CAPS for constants
  - lowerToUpper for variables
  - Most important: MEANINGFUL NAMES!

That's all for today !!

Text Book:

[2] Chapter 2: 2.1 → 2.7