ITI 1120
Lab #2
Translating Algorithms to Java

Romelia Plesa, Alan Williams, Sylvia Boyd,
Daniel Amyot, Diana Inkpen, Gilbert Arbez,
Mohamad Eid

1

Assignment 1

- Assignment 1 is due soon. Check Virtual Campus or the Course Web site for the exact date and time.
- Questions?

For today's lab

- To get to the point that you can write Java programs to:
  - Declare variables
  - Read values from the keyboard
  - Do some straight-line calculations (no branches or calls to other algorithms)
  - Print the results
- Learn how to code mathematical expressions in Java
- Learn how to translate programs composed of 2 algorithms (main and problem solving) into a Java program with one class that contains 2 methods

3

Numeric data types

- You must declare the type of each variable you use
  - Integers: `int`
  - Real numbers: `double`
    - Why "`double` ?" Real numbers were originally referred to as "floating point values" and there is still a type called `float`.
    - It was discovered that a `float` wasn't able to store enough decimal places for many uses, and an improved version was introduced called "double-precision" values, hence: `double`

4

Ranges of Data Types

- Integers
  - `byte`, 1 byte, -128 to 127
  - `short`, 2 bytes, -32768 to 32767
  - `int`, 4 bytes,
    - -2 147 483 648 to -2 147 483 647
  - `long`, 8 bytes, -9×10^{18} to $9 * 10^{18}$
- Real
  - `float`, 4 bytes, +/- 10^{-45} to 10^{38}
  - `double`, 8 bytes, +/- 10^{-324} to 10^{308}

Operators

- There is a version of each operator for types `int` and `double`, with one exception:

- Addition: `+`
- Subtraction: `-` (also used for negative values)
- Multiplication: `*`
- Division: `/`
- Modulus: `%` (remainder after division, integers only)

- WARNING: Be careful if you mix integers and real values in the same program statement.

Integer division

- When an integer denominator does not divide evenly into an integer numerator, the division operator `/` drops the fraction from the result, producing another integer.
- The modulus operator `%` produces the integer remainder

```
5 / 3    //divide, and drop fraction, resulting in 1
5 % 3    // remainder when 5 is divided by 3,
         // resulting in 2
```

- How can you use modulus to tell if a number is even or odd?

Common errors in arithmetic expressions

- Precedence of operators: `*` before `+`

  `6 + 3 * 2` Result: `12`
  - Equivalent to `6 + (3 * 2)`
  - Use parentheses to change the order of evaluation

    `(6 + 3) * 2` Result: `18`

- Integer division, versus division of real numbers

  `5 / 4` Result: `1`

  `5.0 / 4.0` Result: `1.25`

Getting ready to write programs…..

- Copy **`Template.java`** into your working directory.
  - You should have a copy on your **`H:`** drive; if you don't, save one there from the course Web Site.
- Copy **`ITI1120.java`** into your working directory.
- Start up Dr. Java
- Click on "open" and select **`Template.java`** and **`ITI1120.java`**
- Start a "new" file.
- Copy and paste the contents of the **`Template.java`** file into the (empty) unnamed file.
- Close the file **`Template.java`**

9

Java program structure (for now)

1. Consists of one class with two methods
   - The "main" that is used to interact with the user
   - The second in the implementation of an algorithm model to solve some problem.
2. Translate the "main" algorithm model to the "main" method, for
   - Setting up input from the keyboard.
   - Asking the user to enter the values of each GIVEN of the problem solving algorithm/method.
   - Call the problem solving method.
   - Print the results returned by the problem solving method to the console
3. Translate the problem solving algorithm to the problem solving method
   - Results should be returned

10

Classes Reading from the keyboard

- The Scanner Class

nextInt( ): Returns an integer of type **int**.

nextDouble( ): Returns a "real" number of type **double**

nextBoolean( ): Returns a value of **true** or **false** as a value of type **boolean**

nextLine( ): Returns a **String** with the entire remaining contents of the line.

- The ITI1120 Class

ITI1120.readInt( ) : Returns an integer of type **int**

ITI1120.readDouble( ) : Returns a real number of type **double**

ITI1120.readChar( ) : Returns a character of type **char**

ITI1120.readBoolean( ) : Returns a value of type **boolean**

ITI1120.readDoubleLine( ) : Returns a array of **double**

ITI1120.readIntLine( ) : Returns an array of **int**

ITI1120.readCharLine( ) : Returns an array of **char**

ITI1120.readString( ) : Returns a string of type **String**

11

Exercise 1

- Translate the pseudocode main algorithm to the main method.

```
GIVENS:          (none)
RESULTS:         (none)
INTERMEDIATE:
                 n1, n2 ,n3          (three real numbers)
                 average             (the average of n2, n2, n3)
HEADER:          main()
BODY:
    (get the numbers from the user)
    printLine("Please enter three numbers")
    n1 ← readReal()
    n2 ← readReal()
    n3 ← readReal()
    (Call the problem solving algorithm)
    average ← computeAverage(n1, n2, n3)
    (Print the results)
    printLine("The average is ", average);
```

12

Exercise 1

- Translate the pseudocode algorithm to a problem solving for averaging three numbers.

GIVENS: num1, num2 ,num3 (three numbers)

RESULTS: avg (the average of num1, num2, and num3)

INTERMEDIATE:
sum (the sum of num1, num2, num3)

HEADER: (avg) ← computeAverage(num1, num2 ,num3)

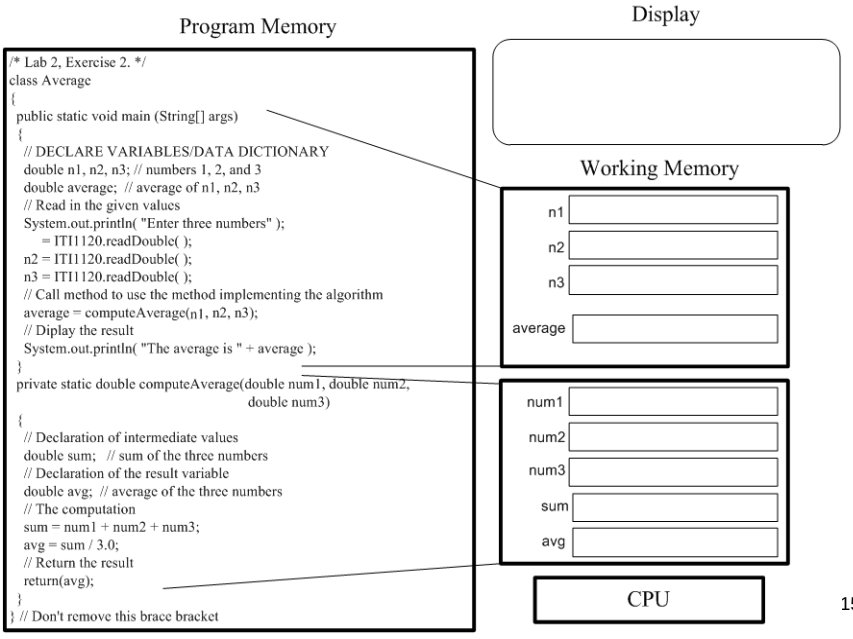
BODY:
sum ← num2 + num2 + num3
avg ← sum / 3

13

Exercice 2

- Programming Model
  - Review the programming model in the next slide (not that it concerns the program from exercise 1).
  - Complete it for the execution of the program given that the user types in the values 2.3, 4.6, and 7.8 when requested.
    - Show how values changed in the variables.
    - Show with arrows how some values (givens/results) are exchanged between methods.
    - Show the dialogue with the user on the screen.
  - If you have printed this presentation, you can complete the model on paper.
  - Otherwise, the Visio template, Lab2Ex2.vsd has been provided so can complete the model electronically.

Exercise 2



15

Using the Debugger

- Using Dr. Java's "debug mode" ("Debugger → Debug Mode"), you can do the equivalent of an algorithm trace for a Java program.
  - You can go through the program one step at a time.
  - You can stop the program at "break points" of your choosing.
  - You can check the values of variables.

- Try this for the program you wrote for Exercise 1, the average of 3 numbers.
- Use the programming mode to follow the execution of the program.

16

Break Points

- Select a line of your program, and under the debug menu, choose "toggle break point on this line".
  - The first `System.out.println` statement is a good choice
  - This will change the colour of the chosen line of code to red.
- You can also right-click on a line and select "Toggle BreakPoint".
  - Many lines can be (de)selected this way.
- When you run the program, the program will stop just before this line is going to be executed.
- In the interactions window, the debugger will tell you where the program is, and the current line of code will be coloured light blue.

17

Watches

- To keep track of the values of variables as they change, use a "watch"
  - Double-click on an empty area in the "name" column, then type in the name of a variable, and hit 'enter'.
  - If the variable already has a value, it will be shown. If the variable does not yet have a value, the value will say <not found>.
- Try this for all of the variables you use in your program for example 1.
- As the program executes, each time the program stops in the debugger, the current values of the variables will be shown.

18

Controlling Execution

- With the debugger, there are four ways to advance through a program
- Resume
  - The program will run up to the next break point, or the end of the program if there are no more break points.
- Step into
  - Use this for the most detailed debugging
  - The program will move to the next statement – even if that statement is in another method.
  - This will not go into methods in the Java software development kit, but the methods in the class `ITI1120` shall be visited..

19

Controlling Execution

- Step over
  - Most commonly used
  - Use this to move to the next statement in the current method.
  - If the current line of the program calls one or more methods, all of those methods will be invoked, and returned from.
- Step out
  - Often used when you have stepped into a method and you want to go back quickly to the previous method.
  - Use this to run as far as the end of the current method.
- Try using "Step over" to go through your Exercise 1 program one statement at a time.
- But use "Step into" when you arrive at the call of the problem solving method (computeAverage).

20

Exercise 3

- Develop a Java program converting temperature expressed in Fahrenheit to Celsius.
- Formula: c ← (f - 32) * 5 / 9
  - Develop your algorithms with Word – start with the file Lab2Ex3.doc.
  - Translate the algorithms to Java. Start with Template.java.
  - The Main algorithm (main() method) requests from the user a temperature value in degrees Fahrenheit and displays the temperature value in Celsius.
  - The problem solving algorithm/method is given the Fahrenheit temperature and produces the Celsius temperature as a result.

21

Exercise 4

- Develop a Java program that takes a two digit positive integer and reverse its digits.
  - For example: The program will transform the two digit integer 12 into 21.
- Hint:
  - The first digit is the result of dividing the integer by 10 (integer division)
  - The second digit is the remainder of the division by 10
    - e.g.:original integer: 12
      - first digit is 12 / 10 = 1
      - second digit is 12 % 10 = 2

22

Exercise 4 - continued

- Develop your algorithms with Word – start with the file Lab2Ex4.doc.
- Translate the algorithms to Java. Start with Template.java.
- The Main algorithm (main() method) requests from the user a number with 2 digits and displays the number with the digits reversed.
- The problem solving algorithm/method is given a 2 digit number and produces the number with the digits reversed.

Built-in math functions

- The `Math` class
  - Automatically loaded: no import required.
- `Math.abs()` - absolute value | x |
- `Math.pow()` - exponentiation
- `Math.sqrt()` - square root $\sqrt{x}$
- `Math.round(n)` - n rounded
- `Math.PI` = 3.14159…
- `Math.E` = 2.71828…
- Examples
  - `Math.abs(-3)` Result: `3` | -3 | = 3
  - `Math.pow(2,5)` Result: `32.0` $2^5 = 32$
  - `Math.sqrt(49)` Result: `7.0` $\sqrt{49} = 7$
- See other math functions in Section 5.9 of the textbook
- On line description at http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Math.html

Exercise 5

- Given 2 points in the X-Y plane (XA,YA) and (XB, YB) compute the distance between the two points, according to the following formula:

$$\sqrt{(XA-XB)^2+(YA-YB)^2}$$

- Develop your algorithms with Word – start with the file Lab2Ex5.doc.
- Translate the algorithms to Java. Start with Template.java.
- The Main algorithm (main() method) requests from the user the coordinates of 2 points and displays the distance between the 2 points.
- The problem solving algorithm/method is given two coordinates and computes the distance as the result.

25

Attention!

- Start your programs with a personalized version of the Template.java provided (insert your name, student number, etc.)
- According to standard convention, the class names in Java start with Upper-case and names for variables and methods start with lower case.
- Use indentation to make your programs readable
  - HINT: in Dr Java, if you type Cntrl-A (all your code will be selected) and then type Tab, Dr Java will organize your code using standard indentation convention.

26

For Super-Users ☺

- In Dr. Java, click on Tools, and select the menu "Javadoc"
  - With comments that have specific format, you can generate Web pages that serve as documentation for your program
  - This feature may be useful later in the course.