CS 106
Introduction to Computer Science I

06 / 02 / 2015

Instructor: Michael Eckmann
Today’s Topics

• Operators
• continue if/else statements
• User input
Operators

- + when used with numeric types (e.g. int, float etc.) acts as an add, however, + when used with Strings is a concatenation operator.
  - example: `text_string = “Hello ” + first_name;`
- = is the assignment operator.
- * is multiplication, / is division, % is modulus and - is subtraction.
- ( ) left and right parentheses are considered operators as well.
The modulus operator (Integer division & remainder)

• % is the modulus operator
  ```
  int val1=16, val2 = 5, remain;
  remain = val1 % val2;
  ```

• The modulus operator returns the remainder after `val1` is divided by `val2`.

• Suppose `val1 = 16` and `val2 = 5`, `remain` would have the value 1 because 16/5 is 3 with a remainder of 1.
Arithmetic Operator precedence levels

- ( ) parentheses are evaluated first
- if parens are nested then the innermost pair is evaluated first.
- *, /, % multiplication, division and modulus are next --- if several of these, they are evaluated left to right.
- +, - addition and subtraction are last --- if several of these, they are evaluated left to right
- It’s useful to memorize this precedence list
Arithmetic Operator precedence example

\[
\text{some\_result} = ( 5 + 4 - 1 ) * 6 + ( 2 - 10 );
\]

- the order in which operations are performed could change the computation's answer.
- *(wrong way)* suppose we did 2-10 = -8 and then 6-8 = -2 then 5+4-1 = 8, 8*(-2) = -16
- *(correct way)* 5+4-1=8, 2-10=-8, 8*6 = 48, 48-8 = 40
String concatenation / integer addition

• Now that we know that operators which are at the same precedence level get evaluated from left to right, we can figure out how and why the following line of code works as it does:
• `System.out.println("10 plus 5 is " + 10 + 5);`
• will print
  10 plus 5 is 105

• the first `+` acts like concatenation and joins the string and the 10 which gets converted to a String automatically. Then the second `+` acts like concatenation of the String to 5 which is converted to a String automatically.
Equality and Relational operators

==  is equal to operator
!=  is not equal to operator
<   is less than operator
>   is greater than operator
<=  is less than or equal to operator
>=  is greater than or equal to operator
if / else structure

if (condition)
{
  statement_to_do_if_condition_is_true;
  another_statement_to_do_if_condition_is_true;
}
else
{
  statement_to_do_if_condition_is_false;
  another_statement_to_do_if_condition_is_false;
  yet_another_statement_to_do_if_condition_is_false;
}
if / else structure

• condition is something that will evaluate to true or false (usually using relational or equality operators.)

• if you only want to do one statement (in the if or else portion) curly braces are not required

• if you want to do more than one statement the curly braces are necessary

• also, the else portion of the if / else structure is optional
Exercise

• Write an application given two integers, displays the larger number followed by the words “is larger”. If the numbers are equal, print the message “These numbers are equal.”
compound statements

- A single statement is a line of code ending in a semicolon.
- A compound statement is a set of statements within a pair of curly braces.
- A compound statement may be used anywhere in a program that a single statement can be used.
- example of a single statement
  
  amount = 45;

- example of a compound statement

  
  {
      area = 3.14159 * radius * radius;
      circumference = 2 * 3.14159 * radius;
  
  }
if / else structure revisited

• The **if** part of the **if / else** structure can contain a single statement or a compound statement. Same for the **else** portion of the **if / else**.

```java
if (degrees_fahr <= 45)
    System.out.println("It is pretty cold.");
else
    System.out.println("It is not too cold.");
    System.out.println("You don't need a jacket.");
```

• What would print out if **degrees_fahr = 30** before the above **if / else**?
if / else and compound statements

• The code on the preceding slide would print “You don’t need a jacket.” regardless of the value of degrees_fahr.

• The following code, using a compound statement, only prints that line if degrees_fahr > 45.

```java
if (degrees_fahr <= 45)
    System.out.println("It is pretty cold.");
else
{
    System.out.println("It is not too cold.");
    System.out.println("You don't need a jacket.");
}
```
another subtlety of the if/else

- Java always associates an else with the preceding if unless curly braces are used to force it not to do this.

- example:
  ```java
  if (i == 0)
      if (j == 0)
          System.out.println("both i and j are 0");
  else
      System.out.println("i is not 0");
  ```

- The above code will not execute like the programmer thought, regardless of the indentation.
- The else will be paired with the if (j==0)  NOT  the if (i == 0).
another subtlety of the if / else

- An `else` can be forced by the programmer to pair up with a different (not the immediately preceding) `if` by using curly braces.

```java
if (i == 0)
{
    if (j == 0)
        System.out.println("both i and j are 0");
}
else
    System.out.println("i is not 0");
```

- The above code will now execute like the programmer intended.
Empty statement

;

The above is a valid statement in Java called the empty statement.
Semicolon after if condition is (almost) always a logic error

```java
if (condition) {
    System.out.println(“Hey”);
}
```

Semicolons are at the end of statements (ifs are not statements)
The first ; is an “empty statement” and that is what gets executed if the condition is true. The next line will always execute, regardless of whether condition is true or not!
Let's put these situations in code now

• We'll try the following situations
  – An if with semicolon after condition (mistake)
    • with and without an else clause
  – An if with multiple statements trying to be inside it but without curly braces (mistake)
    • with and without an else clause
  – An else clause with multiple statements trying to be inside it but without curly braces (mistake)
  – The if inside an if example from a few slides ago with the question of which if the else matched up with.
import statement

• Normally we need to tell the compiler where to look for the class or method we are using in the Java API. This is done by the use of the import statement at the beginning of your source code file.

• To use a class called JOptionPane to create dialog boxes, we need to add the following line to the beginning of our code:

  – import javax.swing.JOptionPane;
import statement

• The import statement is of the form
  – **import** package.class;

• **javax.swing** is a package and **JOptionPane** is the class we want to use in it.

• **import javax.swing.** * would allow us to use any class in the **javax.swing** package.
import statement

• We didn’t need to `import` anything to use `System.out` in the HelloWorld program because `System` is a class that is part of the `java.lang` package and `java.lang` gets implicitly imported in all Java programs.

• Java API Specification is on the web and is where you’ll find all the classes available to you in the Java API. There is a link to it on our course web page.
javax.swing.JOptionPane

• Contains methods to show dialog boxes.
• JOptionPane.showMessageDialog displays messages in a dialog box.
• JOptionPane.showInputDialog allows for user input in a dialog box.
• Nicer to use than reading from / printing to the console screen.
JoptionPane class for I/O

• `showMessageDialog` takes two parameters, the first of which should always be null for now.
• The second is a String that should be outputted.

• `showInputDialog` takes one parameter which is a String that displays --- it should tell the user what to enter. The user's input is returned as a String.
JOptionPane class for I/O

• Example usage:
  
  ```java
  import javax.swing.JOptionPane; // need this line above your class

  String input_string; // variable to store the user's input

  JOptionPane.showMessageDialog(null, "Hey");
  input_string = JOptionPane.showInputDialog("Enter something.");
  ```
javax.swing.JOptionPane

• Let’s write Hello World using a message dialog box instead of System.out.println.
java.util.Scanner

• Scanner input_scan = new Scanner(System.in);
• methods that you can call on your scanner object include:
  – nextInt() --- reads an int from keyboard
  – nextDouble() --- reads a double from keyboard
  – nextLine() --- reads a line (as a String) from keyboard
  – next() --- reads a “word” from the keyboard --- which is a string of nonwhitespace chars delimited by whitespace. whitespace is \n, blank space character, \t, \r
java.util.Scanner

- `nextLine()` --- reads a line (as a String) from keyboard
  - this method “consumes” the `\n` but does not make it part of the String that is returned.

  ```java
  String s1, s2;
  Scanner my_scan = new Scanner(System.in);
  s1 = my_scan.nextLine();
  s2 = my_scan.nextLine();
  ```

  - if input is:
    
    ```
    CS106
    Introduction to Computer Science I
    ```
    
    - there is a `\n` after CS106 in the input (e.g. user hit enter key), but s1 will be “CS106” and s2 will be “Introduction to Computer Science I” neither will have `\n` as a character in its String.
java.util.Scanner

• There's an unfortunate problem with Scanner when using one to get both numeric and String input.

• If you need to get both numeric and String input from user input from the keyboard I recommend creating two Scanners, one that only gets numeric input (nextInt, nextDouble) and one that only gets the String input (nextLine, next).
Pseudocode

- pseudocode is an informal use of English to describe what a program is to do and in what order
- pseudocode is not an actual computer programming language
- it is used prior to writing actual code to help the programmer in the planning stages of a program
Example Application Exercise

• write a program to compute the number of projected home runs a baseball player will hit for the season based on how many homers he’s hit so far.

• Output should look like:

player’s name is projected to hit number home runs in 162 games.

• Any ideas?

Michael Eckmann - Skidmore College - CS 106 - Summer 2015
Pseudocode for our example

get player’s name from the user
get the number of homeruns so far
get the number of games played so far

compute the number of projected homeruns for this player based on a season of 162 games by using the following calculation

\[
\frac{\text{projected homeruns}}{162} = \frac{\text{homeruns so far}}{\text{games played so far}}
\]
Pseudocode for our example (continued)

from this equation,

\[
\frac{\text{projected homers}}{162} = \frac{\text{homers so far}}{\text{games played so far}}
\]

we can multiply both sides of the equation by 162 and get

\[
\text{projected homers} = \frac{\text{homers so far} \times 162}{\text{games played so far}}
\]
Pseudocode for our example
(continued)

Print out the following with actual values for player’s name and number

player’s name is projected to hit number home runs in 162 games.

Pseudocode could be more fleshed out than what we have done here --- use as much or as little detail in pseudocode as you prefer.
Now we can write the program based on our pseudocode.