Elementary Programming

More I/O (+ C.S. 2)

EECS1021:
Object Oriented Programming:
from Sensors to Actuators
Winter 2020

Original slides: DR. CHEN-WEI "JACKIE" WANG
Updates: DR. JAMES ANDREW SMITH

Topics in “Elementary Program’g” series

- Intro to Java (General)
- Operations and Data
- Input & Output (+ Case Study 1)
- *More I/O (+ Case Study 2)*
- Numbers Types & Conversions
- Software Development High Level Process

Case Study 2: Display Time

Problem: prompt the user for an integer value of seconds, divide that value into minutes and remaining seconds, and print the results. For example, given an input 200, output “200 seconds is 3 minutes and 20 seconds”.

```
import java.util.Scanner;
public class DisplayTime {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    /* Prompt the user for input */
    System.out.print("Enter an integer for seconds: ");
    int seconds = input.nextInt();
    int minutes = seconds / 60; /* minutes */
    int remainingSeconds = seconds % 60; /* seconds */
    System.out.print(seconds + " seconds is ");
    System.out.print(" minutes and ");
    System.out.println(remainingSeconds + " seconds");
  }
}
```

Where May Assignment Sources Come From?

In `target = source`, the *assignment source* `source` may come from:

- A literal

```
int i = 23;
```

- A variable

```
int i = 23;
int j = i;
```

- An expression involving literals and variables

```
int i = 23;
int j = i * 2;
```

- An input from the user

```
Scanner input = new Scanner(System.in);
int i = input.nextInt();
int j = i * 2;
```

Escape Sequences

An *escape sequence* denotes a single character.

- Specified as a *backslash* (\) followed by a *single character*
  - e.g., `\t`, `\n`, `\'`, `\"`, `\\`
- *Does not mean literally*, but means specially to Java compiler
  - `\t` means a “tab”
  - `\n` means a “new line”
  - `\\` means a “back slash”
  - `\'` means a “single quote”
  - `\"` means a double quote
- May use an *escape sequence* in a character or string literal:
  - `'''` [INVALID; need to *escape* the { ']
  - `'\''` [VALID]
  - `'"'` [VALID; no need to *escape* the "]
  - `"""` [INVALID; need to *escape* the "]
  - `"\""` [VALID]
  - `"'"` [VALID; no need to *escape* the ']
  - `"\n\t""` [VALID]

Identifiers & Naming Conventions

- Identifiers are *names* for identifying Java elements: *classes*, *methods*, *constants*, and *variables*.
- An identifier:
  - Is an arbitrarily long sequence of characters: letters, digits, underscores (_), and dollar signs (`$`).
  - Must start with a letter, an underscore, or a dollar sign.
  - Must not start with a digit.
  - Cannot clash with reserved words (e.g., `class`, `if`, `for`, `int`).
- *Valid* ids: `$2`, `Welcome`, `name`, `_name`, `YORK_University`
- *Invalid* ids: `2name`, `+YORK`, `Toronto@Canada`
- More conventions:
  - Class names are compound words & mixed capitalization: e.g., `Tester`, `HelloWorld`, `TicTacToe`, `MagicCardGame`
  - Variable and method names are like class names, except 1st word is all lower case: e.g, `main`, `firstName`, `averageOfClass`
  - Constant names are underscore-separated upper cases: e.g., `PI`, `USD_IN_WON`