Elementary Programming

Number Types & Conversions

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

LASSONDE SCHOOL OF ENGINEERING

- 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

Numerical Type Conversion: Coercion

- *Implicit* and *automatic* type conversion
- Java *automatically* converts an integer value to a real number when necessary (which adds a fractional part).

```
double value1 = 3 * 4.5;  /* 3 coerced to 3.0 */
double value2 = 7 + 2;  /* result of + coerced to 9.0 */
```

- However, does the following work?

```
int value1 = 3 * 4.5;
```

  - RHS evaluates to `13.5` due to coercion.
  - LHS declares a variable for storing integers (with no fractional parts).

∴ Not compatible [***compile-time error***]

⇒ Need a way to "truncate" the fractional part!

Numerical Type Conversion: Casting

- *Explicit* and *manual* type conversion
- **Usage 1**: To assign a real number to an integer variable, you need to use explicit *casting* (which throws off the fractional part).

```
int value3 = (int) 3.1415926;
```

- **Usage 2:** You may also use explicit *casting* to force precision.
  - ```
    System.out.println(1 / 2);   /* 0 */
    ```
    ∵ When both operands are integers, division evaluates to quotient.
  - ```
    System.out.println( ((double) 1) / 2 ); /* 0.5 */
    System.out.println( 1 / ((double) 2) ); /* 0.5 */
    System.out.println( ((double) 1) / ((double) 2) ); /* 0.5 */
    ```
    ∵ Either or both of the integers operands are cast to `double` type
  - ```
    System.out.println((double) 1 / 2);   /* 0.5 */
    ```
    ∵ Casting has *higher precedence* than arithmetic operation.
  - ```
    System.out.println((double) (1 / 2));   /* 0.0 */
    ```
    ∵ Order of evaluating division is forced, via parentheses, to occur first.

Numerical Type Conversion: Exercise

Consider the following Java code:

```
1  double d1 = 3.1415926;
2  System.out.println("d1 is " + d1);
3  double d2 = d1;
4  System.out.println("d2 is " + d2);
5  int i1 = (int) d1;
6  System.out.println("i1 is " + i1);
7  d2 = i1 * 5;
8  System.out.println("d2 is " + d2);
```

Write the **exact** output to the console.

```
d1 is 3.1415926
d2 is 3.1415926
i1 is 3
d2 is 15.0
```

Expressions (2.1)

Consider the following Java code, is each line type-correct? Why and Why Not?

```
1  double d1 = 23;
2  int i1 = 23.6;
3  String s1 = ' ';
4  char c1 = " ";
```

- **L1**: YES [coercion]
- **L2**: NO [cast assignment source, i.e., `(int) 23.6`]
- **L3**: NO [cannot assign char to string]
- **L4**: NO [cannot assign string to char]

Expressions (2.2)

Consider the following Java code, is each line type-correct? Why and Why Not?

```
1  int i1 = (int) 23.6;
2  double d1 = i1 * 3;
3  String s1 = "La ";
4  String s2 = s1 + "La Land";
5  i1 = (s2 * d1) + (i1 + d1);
```

- **L1**: YES [proper cast]
- **L2**: YES [coercion]
- **L3**: YES [string literal assigned to string var.]
- **L4**: YES [type-correct string concat. assigned to string var.]
- **L5**: NO [string × number is undefined]