
# COMP90041

## Programming and Software Development

## Semester 1, 2021

## Lab 2

Andrew Naughton

andrew.naughton@unimelb.edu.au

THE UNIVERSITY OF MELBOURNE

# Outline

- Pre Increment/Decrement
- Post Increment/Decrement
- printf
- Format Specifiers
- Scanner
- Scanner Methods
- Exercises

THE UNIVERSITY OF MELBOURNE

# Pre Increment/Decrement

- ++x is a special expression that increments x and then returns the new incremented value
  - x = 5 [x is 5]
  - ++x [x is 6]
  - x [x is 6]
- --x is the same, except that it decrements x and then returns the new decremented value

- Called pre **crement because it **crements variables before returning their value

THE UNIVERSITY OF MELBOURNE

# Post Increment/Decrement

- x++ is a special expression that returns x and then increments it
  - x = 5 [x is 5]
  - x++ [x is 5]
  - x [x is 6]
- x-- is the same, except that it returns x and then decrements it

- Called post **crement because it **crements variables after returning their value

THE UNIVERSITY OF MELBOURNE

# Question 1

- What will this code print?

```
int x = 5; int y = 5;

System.out.println(++x);
System.out.println(x);

System.out.println(y++);
System.out.println(y);
```

THE UNIVERSITY OF MELBOURNE

# Question 2

- What will this code print?

```
int x = 10; int y = 5;

System.out.println(x++  –  ++y);
```

THE UNIVERSITY OF MELBOURNE

# printf

- printf is like print, but you can control how data is formatted
- Of the form:
  - System.out.printf(<format string>, <arg 1>, …, <arg n>);
- Format string contains format specifiers, one for each of the arguments
  - These begin with %

THE UNIVERSITY OF MELBOURNE

# Format Specifiers

- Of the form:
  - %<width><.precision><conversion-character>

| Conversion Character | Usage |
|---|---|
| d | To format an integer |
| s | To format a string |
| c | To format a character |
| f | To format a float or double |
| e | To format a float or double in exponential notation |
| g | Java chooses the shorter of %f or %e |

THE UNIVERSITY OF MELBOURNE

# %X.Y{d,s,c,f,e,g}

- X specifies the minimum number of characters to be printed
  - If more than X characters in argument, we print full number of characters
  - If less than X characters in argument, we pad with whitespaces
  - If X is negative, the value will be left-justified, otherwise right-justified
  - Example:
    - “Hello” (%10s) -> “#####Hello” where # denotes whitespace
    - “Hello” (%-10s) ->“Hello#####”

- Y specifies the number of decimal places to keep
- Example:
  - 10.667 (%.2f) -> 10.67

THE UNIVERSITY OF MELBOURNE

# Scanner

- How we read input from the console
- To use, requires importing from java.util.Scanner
- Create an object (which we'll call "keyboard") of the class Scanner with:
  - Scanner keyboard = new Scanner(System.in);

THE UNIVERSITY OF MELBOURNE

# Scanner Methods

- nextLine() reads up to and including newline
- Others (next, nextInt, nextDouble) do not read after the next token

- To read a double on one line followed by the next whole line:

```
double x = keyboard.nextDouble();
keyboard.nextLine(); // throw away rest of line
String line = keyboard.nextLine();
```

THE UNIVERSITY OF MELBOURNE

# Question 3

- How can we read in the following input?

```
2
heads are better than
1 head
```

- Note: read the first line as an integer


THE UNIVERSITY OF
MELBOURNE

# Exercises