CSSE2002/7023

Semester 2, 2021

Programming in the Large

Week 1.2: Intro to Java

In this Session

- What is Java?
- First Java program
- Some differences between Java and Python
- Variables, control statements, loops, and recursion in Java

What is Java?

"The Java programming language is a general-purpose, concurrent, class-based, object-oriented language."

— Gosling et al, 2015

Gosling, J., Joy, B., Steele, G., Bracha, G. and Buckley, A. (2015). The Java language specification: Java® SE 8 edition, 2015. URL

`https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf`

What is Java?

"The Java programming language is a general-purpose, concurrent, class-based, object-oriented language."

— Gosling et al, 2015

- General purpose – wide variety of application domains.

Gosling, J., Joy, B., Steele, G., Bracha, G. and Buckley, A. (2015). The Java language specification: Java® SE 8 edition, 2015. URL

`https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf`

What is Java?

"The Java programming language is a general-purpose, concurrent, class-based, object-oriented language."

— Gosling et al, 2015

- General purpose – wide variety of application domains.
- Concurrent – can be used to create applications that use multiple computers, or multiple cores on the same computer.

Gosling, J., Joy, B., Steele, G., Bracha, G. and Buckley, A. (2015). The Java language specification: Java® SE 8 edition, 2015. URL

`https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf`

What is Java?

"The Java programming language is a general-purpose, concurrent, class-based, object-oriented language."

— Gosling et al, 2015

- General purpose – wide variety of application domains.
- Concurrent – can be used to create applications that use multiple computers, or multiple cores on the same computer.
- Class-based – everything in Java is within a "class" (more on these next week).

Gosling, J., Joy, B., Steele, G., Bracha, G. and Buckley, A. (2015). The Java language specification: Java® SE 8 edition, 2015. URL `https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf`

What is Java?

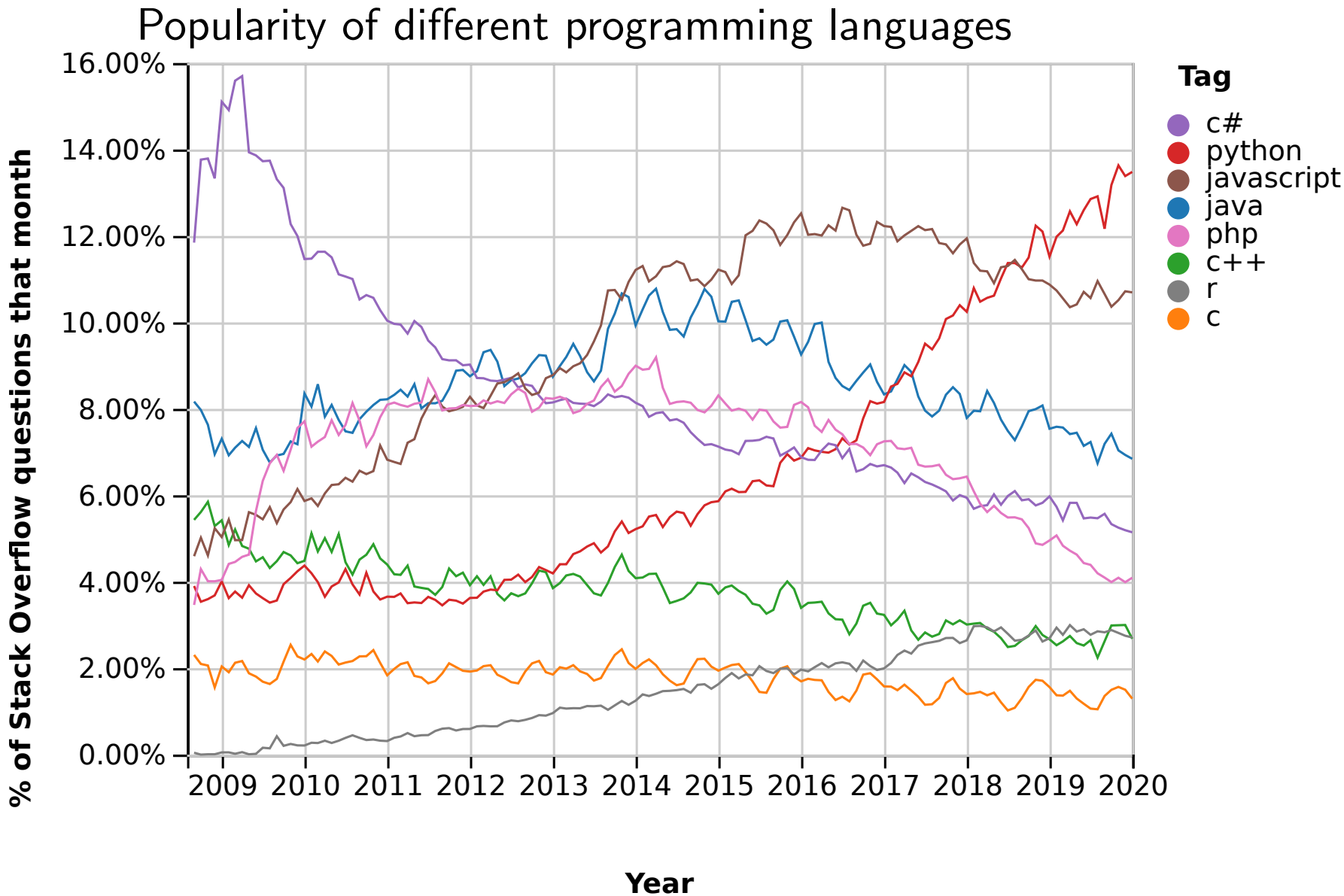
"The Java programming language is a general-purpose, concurrent, class-based, object-oriented language."

— Gosling et al, 2015

- General purpose – wide variety of application domains.
- Concurrent – can be used to create applications that use multiple computers, or multiple cores on the same computer.
- Class-based – everything in Java is within a "class" (more on these next week).
- Object-oriented – a way of designing code based on the concept of an "object" (more on this next week too).

Gosling, J., Joy, B., Steele, G., Bracha, G. and Buckley, A. (2015). The Java language specification: Java® SE 8 edition, 2015. URL `https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf`

Java Popularity



(Image from `https://insights.stackoverflow.com/trends`)

Java is Widely Taught

Java is taught in many universities worldwide.

- 44 introductory programming courses from Australia were evaluated. 31% used Java.
- 80 introductory programming courses from the United Kingdom were evaluated. 46% used Java.
- 1019 programming courses from 218 universities in 35 European countries were evalauted. 20.7% used Java.

Details:

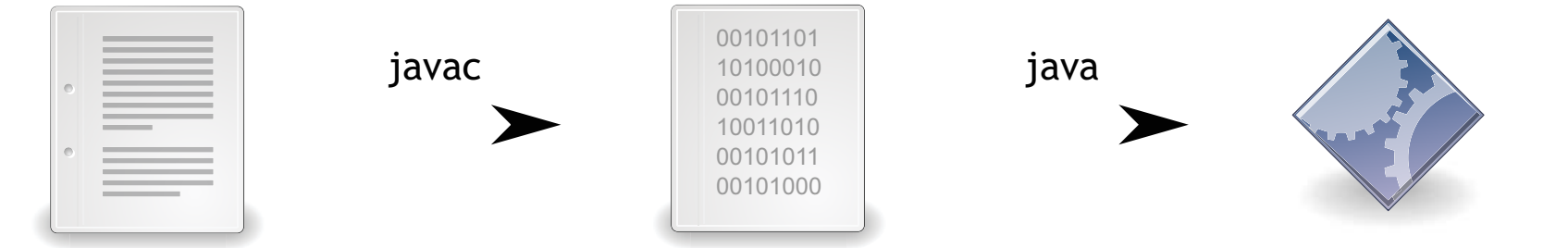
Mason, R., Crick, T., Davenport, J. H., & Murphy, E. (2018, February). Language choice in introductory programming courses at Australasian and UK universities. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (pp. 852-857).

Aleksić, V., & Ivanović, M. (2016). Introductory programming subject in European higher education. Informatics in Education, 15(2), 163-182.

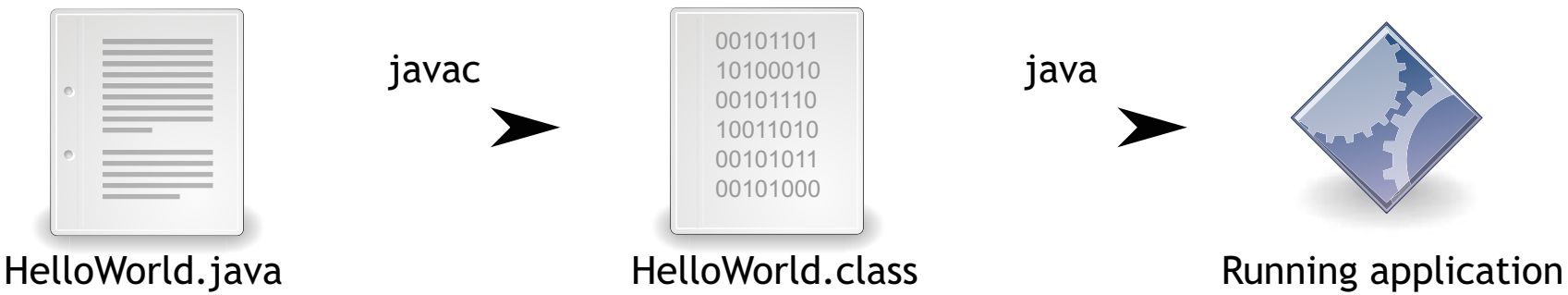
First Java Program

HelloWorld.java

Compiling and Running Java Programs



Compiling and Running Java Programs



However, most of you will use IntelliJ, which will compile and run your code when you press the "Play" button.

Variables (Starting with Python)

variable = a box with a name...

```
# x does not exist
x = 1
print(x)
x = 1.7
print(x)
x = "Hello"
print(x)
```

Variables (Starting with Python)

variable = a box with a name...

```
# x does not exist
x = 1
print(x)
x = 1.7
print(x)
x = "Hello"
print(x)
```

Var name is ...

left of =	*put something into the box*
elsewhere	*get something out of the box*

Variables (Starting with Python)

variable = a box with a name...

```
# x does not exist
x = 1
print(x)
x = 1.7
print(x)
x = "Hello"
print(x)
```

Var name is ...

left of =	*put something into the box*
elsewhere	*get something out of the box*

In Python

- Assigning a value into a name is enough to make it exist.
- Different sorts of things can go in the same variable.

Dynamic Typing (Python)

```
x = 1
print(x, type(x))          1 <class 'int'>
x = 1.7
print(x, type(x))          1.7 <class 'float'>
x = "Hello"
print(x, type(x))          "Hello" <class 'str'>
```

Python knows that 1 and 1.7 are different sorts of values (different "dynamic" / "runtime" types) but doesn't restrict variables on that basis.

Java can do this too, but in a more restricted way.

Java — Static Types

Variables have types as well as names so they need to be "declared" before they can be used.

```
int x;          // x will store whole/integer numbers
x = 1;          // legal
x = -5000;      // legal
x = 1.0;        // illegal: not a whole number
x = "15";       // illegal: cannot convert from a string
                // to a number
```

Primitive Types

Java distinguishes two sorts of types:

- "primitive types":

`boolean`		`true` or `false`
`byte`		-100, 120
`char`	one unicode character	'A', '6'
`short`	small whole number	
`int`	whole numbers	1, -500
`long`	larger range of whole numbers	
`float`	floating point number	27.8, -1.9e200
`double`	higher precision floating point	

- "classes" (reference types): everything else

For example, Strings and arrays are classes.
Because literals and variables have types, expressions using them do too. (Watch out — division between ints gives an int answer).

Scope

A variable's scope describes where your program can use that name to refer to that variable. In Java, a variable's scope ends at the end of the block in which it is declared*.

This works in Python:

```
# z does not exist
if 5 > 4:
    z = 2
else:
    z = 3
print(z)
```

This does **not** work in Java:

```
if (5 > 4) {
    int z = 2;
} else {
    int z = 3;
}
System.out.println(z);
```

Uninitialised Variables

Q: What is the value of y after the following?

```
int x, y; // declaring multiple variables

...       // nothing affecting x or y in here

y = x+1;
```

Uninitialised Variables

Q: What is the value of `y` after the following?

```
int x, y; // declaring multiple variables

...       // nothing affecting x or y in here

y = x+1;
```

A: It depends.
`x` and `y` may be set to something which looks like zero, or the code won't compile (depending on where the variable is declared).

It is better to explicitly initialise your variables.

Control Flow — `if` and `switch`

```
if (cond1) {
    body1
} else if (cond2) {
    body2
} else {
    body3
}
```

```
switch (var) {
    case literal1:
        body1
        break;
    case literal2:
        body2
        break;
    default:
        body3
}
```

Example: Switch.java

while and do-while

```
while (condition) {
    body
}
```

```
do {
    body
} while (condition);
```

Example:

```
int i = 0;
while (i < 10) {
    System.out.println(i);
    i++;
}
```

for

```
for (Init; Test; Update) {
  Body
}
```

for

```
for (Init; Test; Update) {
  Body
}
```

Init

Test

for

```
for (Init; Test; Update) {
  Body
}
```

Init	OR	Init
Test		Test
		Body
		Update
		Test

for

```
for (Init; Test; Update) {
  Body
}
```

Init	OR	Init	OR	Init	...
Test		Test		Test	
		Body		Body	
		Update		Update	
		Test		Test	
				Body	
				Update	
				Test	

for

```
for (Init; Test; Update) {
  Body
}
```

Init	OR	Init	OR	Init	...
Test		Test		Test	
		Body		Body	
		Update		Update	
		Test		Test	
				Body	
				Update	
				Test	

Example:

```
for (int i = 0; i < 10; i++) {
  System.out.println(i);
}
```

Java (temporary) "Magic"

Your most basic Java program `XYZ.java` will be:

```
public class XYZ {

    public static void main(String[] args) {
        // your code here
    }

}
```

Examples

Program	Topic
`Loops.java`	`for`, `while`, `do-while`, increment operator
`ForEach.java`	for-each
`Recursion.java`	simple recursion