Tomás Dias Almeida

UNICAMP

VIGO JUG

http://about.me/tomasalmeida

Software Engineer @

netcentric


```
$ javac VigoJUG.java

$ java VigoJUG

* Hola Vigo! Soy su Java User Group :-)

[INFO] Charlas
[INFO] Lightning Talks
[INFO] Workshops
[HACK] Hackatons
[READ] Clubs de Lectura
[HELP] Ideas???
```

> Most people talk about Java the language, and this may sound odd coming from me, but I could hardly care less. At the core of the Java ecosystem is the JVM.

James Gosling,

Creator of the Java Programming Language

(2011, TheServerSide)

CC WikiMedia (2008)

JAVA

- ¿Por qué?
- El pasado: 1.0 a 7
- El presente: 8
- El futuro: 9

Por qué Java?

Cuándo: 1991

Java 1.0: 1996

Quién: El señor Gosling

Dónde: Sun Microsystems

It must be

- simple, object-oriented, and familiar
- robust and secure
- architecture-neutral and portable
- high performance
- interpreted, threaded, and dynamic

de OAK (1996) hasta JDK 1.4 (2002)

- 1.0 - 1996
  - Sintaxis muy parecida a C++
  - Gestión automática de la memoria
  - WORA: *write once, run anywhere*

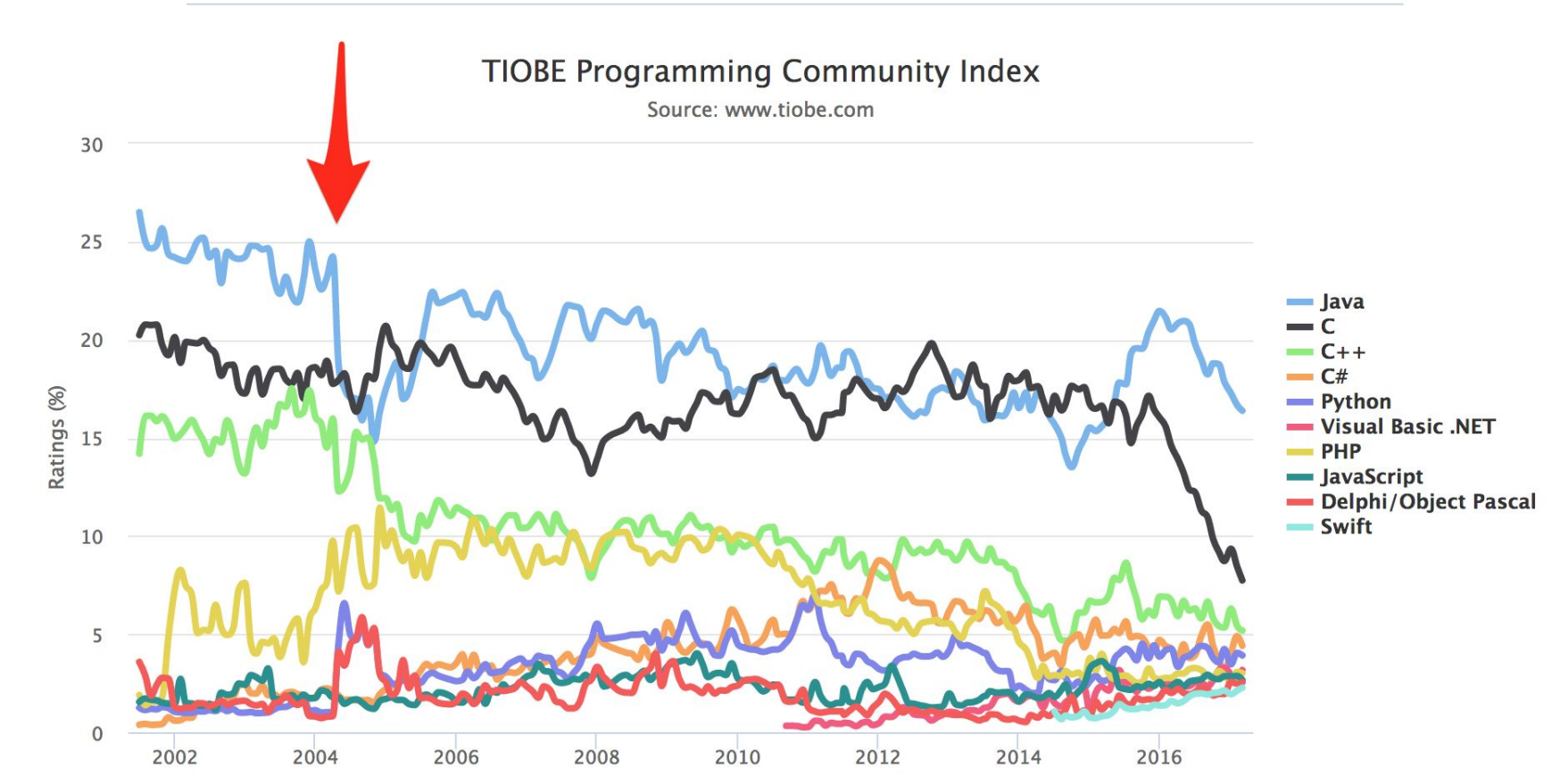
- 1.1 - 1997
  - JIT: just in time compiler

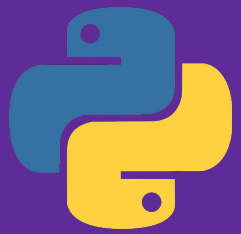
- 1.2 - 1998
  - Collections framework

- 1.3 - 2000
  - JVM Hotspot

- 1.4 - 2002
  - IPv6 Support

El cambio en 2004





Java 5.0, 6 y 7: evolucionamos el lenguaje

Java 5.0 - 2004

- Inclusión de Generics, annotations, Autoboxing/unboxing, Enumerations, Varargs, forEach, Static imports
- Es el "dialecto" de java más conocido y utilizado.

Java 6 - 2006

- Support a Scripting Language: tenemos Javascript en Java!!!!
- Nuevos algoritmos y upgrades en el garbage collection de la JVM.

Java 7 - 2011 (Coin Project)

- String in switches (finalmente!)
- Diamond Operator: `Map<String, List<String>> myMap = new HashMap<>();`
- Varias exceptions en un catch: `catch (IOException | SQLException ex) { … }`

Garbage Collection

Demo

Java 8

El presente

Un gran saludo a la programación funcional

Cambios importantes en el lenguaje

- Métodos default en interfaces: se permite la herencia múltiple de comportamiento.

```
public interface Car{

   default boolean isMoving() {
      ...
   }
}
```

Visibility	Public	Protected	Default	Private
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	Yes	Yes	No
From a subclass in the same package	Yes	Yes *(Package, Inheritance)*	Yes *(Package)*	No
From a subclass outside the same package	Yes	Yes *(Inheritance)*	No	No
From any non-subclass class outside the package	Yes	No	No	No

Cambios importantes en el lenguaje

- Lambdas, Stream y sus amigos
  Permite declarar y operar sobre bucles de una manera mucho más simple.

```
List<String> myList =
    Arrays.asList("a1", "a2", "b1", "c3", "c2", "c1");

myList
    .stream()
    .filter(s -> s.startsWith("c"))
    .map(String::toUpperCase)
    .map(word -> word.concat("!"))
    .sorted()
    .forEach(System.out::println);
```

Java 9

El futuro es hoy!

Java 9 REPL - Read Evaluate Print Loop (JShell)

```
G:\>jshell
|  Welcome to JShell -- Version 9-ea
|  For an introduction type: /help intro


jshell> int a = 10
a ==> 10

jshell> System.out.println("a value = " + a )
a value = 10
```

Adiós a tu Test.java!

Métodos privados en interfaces

```
public interface Car{

   boolean isMoving() {
      ...
   }

   private void clearEngine() {
      ...
   }

   private static boolean isStaticPrivate() {
      ...
   }
```

Factory methods de collections inmutables

```
List immutableList = List.of();

List immutableList = List.of("one","two","three");

Map nonemptyImmutableMap = Map.of(1, "one", 2, "two", 3,
"three")
```

HTTP2 Client (aún compatible con http1.1)

```
import java.net.http.*;

import static java.net.http.HttpRequest.*;

import static java.net.http.HttpResponse.*;
```

Mejoras en Stream

- Método takeWhile

```
Stream<Integer> stream = Stream.of(1,2,3,4,5,6,7,8,9,10)
stream.takeWhile(x -> x < 4).forEach(a -> System.out.println(a))
// Imprime 1 2 3
```

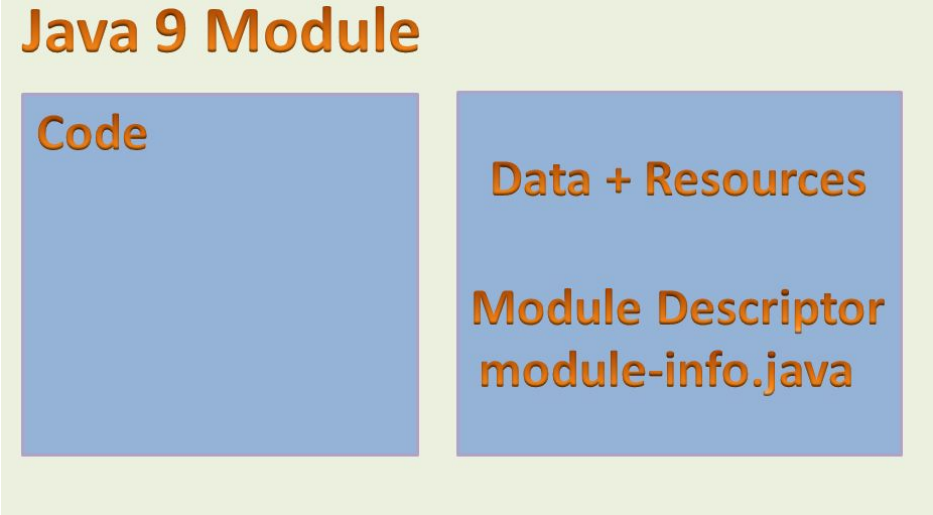
- Método dropWhile

```
Stream<Integer> stream = Stream.of(1,2,3,4,5,6,7,8,9,10)
stream.dropWhile(x -> x < 4).forEach(a -> System.out.println(a))
// imprime 4 5 6 7 8 9 10
```

- Método Iterate

```
IntStream.iterate(2, x -> x < 20, x -> x*x).forEach(System.out::println)
// Imprime 2 4 16
// En JAVA 8
IntStream.iterate(2, x -> x*x).filter(x -> x < 20).forEach(System.out::println)
```

Java 9 Module System (Jigsaw Project)



- 95 módulos y no un tools.jar y rt.jar para todo en el JRE
- Más encapsulation (public is too public)
- Servicios son declarados en metadata (OSGi es programático)
- Start/stop modules in runtime

> **Everyday life is like programming, I guess. If you love something you can put beauty into it.**
>
> **Donald Knuth**

Gracias por la atención :-)

(NO TE OLVIDES DE DECIR EL TÍTULO DE LA CHARLA)


¡Pregunten!
DonaldDraws.com