Estructuras de Datos. 2022/23
Grado en Ingeniería Informática, del Software y Computadores
ETSI Informática
Universidad de Málaga

Estructuras de Datos

@ José E. Gallardo, Francisco Gutiérrez, Pablo López, Laura Panizo
Dpto. Lenguajes y Ciencias de la Computación
Universidad de Málaga

Profesorado

Pablo López (Teoría y prácticas)

Despacho: 3.2.50

Contacto: mensaje o correo interno del **campus virtual**
(por favor, no uses mi correo de la uma)

Tutorías: **exclusivamente** mediante **cita previa** en
https://calendly.com/plopez-uma-es/tutoria

Otros profesores de práctica por asignar

Horario y Aulas

Lunes	10:45 – 12:15 (recuperación)
Martes	12:45 – 14:30 (teoría / prácticas)
Jueves	10:45 – 12:30 (teoría)

Aula: 3.0.8

Laboratorios: 3.1.2, 3.1.3, 3.1.4

Campus Virtual

Dos páginas:

- **Sala común asignatura Estructuras de Datos**
  - transparencias, enunciados de prácticas, foro común, …
- **Ingeniería de Computadores, Grupo A**
  Ingeniería Informática/Software, Grupo D,
  Doble Grado Informática + Matemáticas
  - entrega de prácticas y exámenes, foro particular, …

Objetivos de la asignatura

1. Estudiar Estructuras de Datos
2. Introducir la Programación Funcional

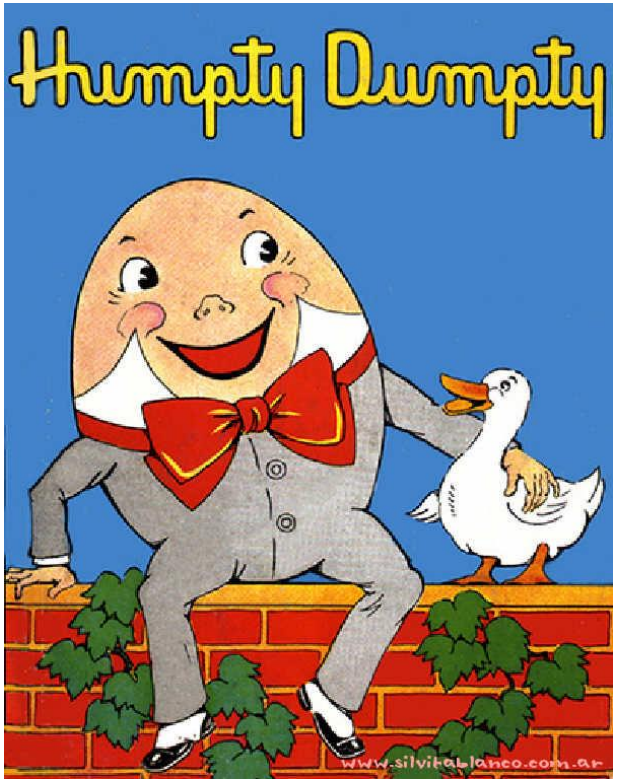
Objetivos de la Asignatura (I)

- Hasta ahora habéis **usado** estructuras de datos ya implementadas:
  - **`List (ArrayList, LinkedList)`**
  - **`Set (HashSet, TreeSet)`**
  - **`Map (HashMap, TreeMap), …`**

- Esta asignatura se centra en el **diseño**, **análisis**, **implementación eficiente** y **uso** de estructuras de datos

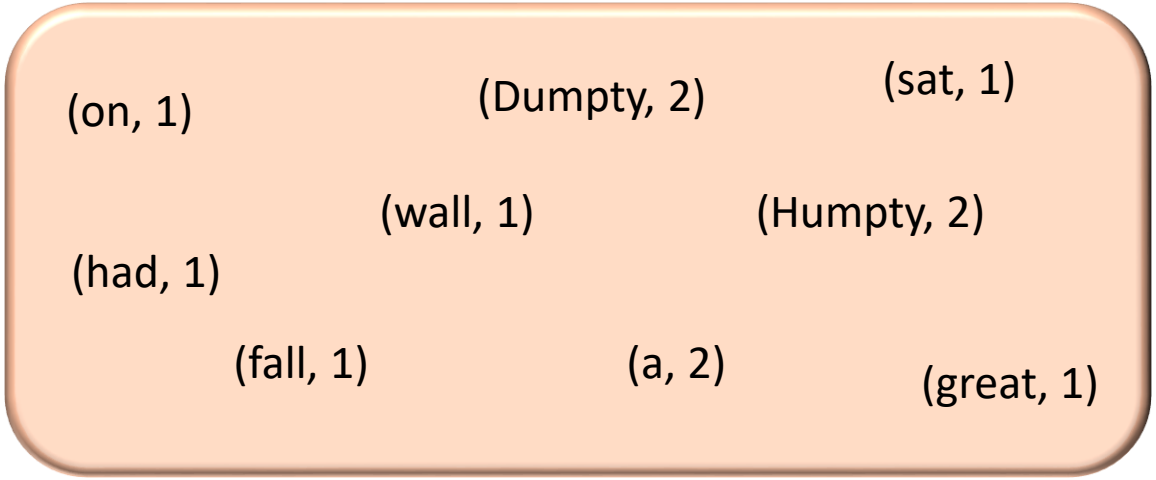
¿Cuál es la palabra más frecuente?

Humpty Dumpty sat on a wall,
Humpty Dumpty had a great fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.



¿Cómo calcular la palabra más frecuente?

- Utilizaremos una **estructura** que almacene pares:
  `(palabra, contador de apariciones)`



*Contenido tras procesar los **dos primeros versos***

- La palabra más frecuente será la que tenga el mayor contador asociado en la **estructura**

Diseño de la estructura (I)

Utilizaremos una estructura **KeyValueStore** genérica que almacene pares:

(Key, Value)

donde la clave (**Key**) es única.

Resolveremos el problema tomando:

Key = palabras,
Value = contadores

Diseño de la estructura (II)

La estructura soporta las siguientes **operaciones**:

```
public interface KeyValueStore<Key, Value> {

// add a key-value pair, replacing it if key present
void put(Key key, Value val);

// return value associated with key, null if not present
Value get(Key key);

// does this collection contain the given key?
boolean contains(Key key);

// return all keys as an Iterable
Iterable<Key> keys();

}
```

Uso de la estructura (I)

```
KeyValueStore<String, Integer> pairs = new ...;

...

for (String word : text) {
      // feed data structure
}

...

for (String word : pairs.keys()) {
      // compute most frequent word
}
```

Uso de la estructura (II)

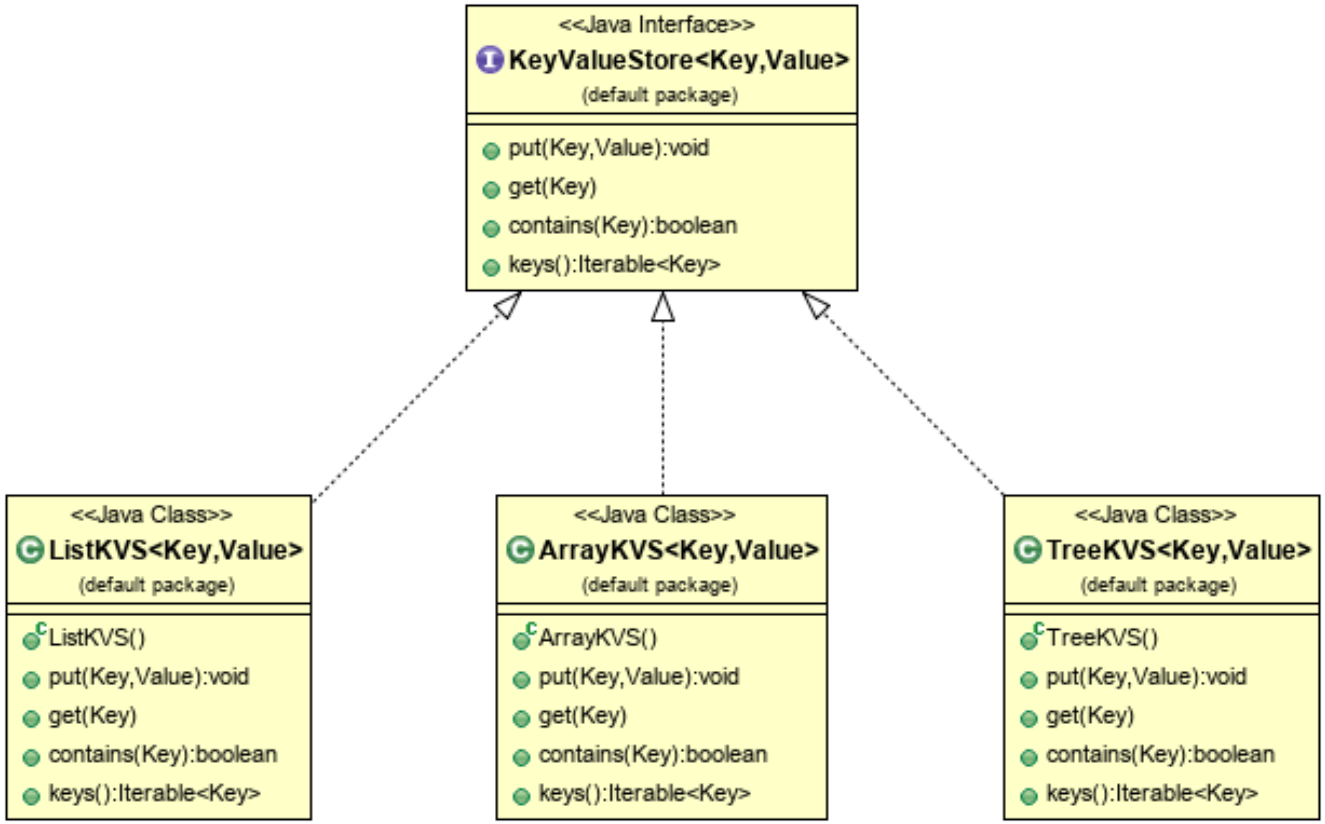
```
KeyValueStore<String, Integer> pairs = new ...;

for (String word : text) {
      Integer n = pairs.get(word);
      if (n == null)
            pairs.put(word, 1); // first occurrence
      else
            pairs.put(word, n + 1);
}

int maxOccurrences = 0;
String mostFrequentWord = null;
for (String word : pairs.keys()) {
      int n = pairs.get(word);
      if (n > maxOccurrences) { // n >= maxOccurrences ?
            maxOccurrences = n;
            mostFrequentWord = word;
      }
}
```

Implementaciones de la estructura (I)

Aplicaremos el mismo algoritmo utilizando 3 implementaciones distintas de la estructura:



Implementaciones de la estructura (II)

- **`ListKVS<Key, Value>`**
  Lista enlazada (búsqueda secuencial)
- **`ArrayKVS<Key, Value>`**
  Array ordenado (búsqueda binaria)
- **`TreeKVS<Key, Value>`**
  Árbol roji-negro (ordenado y equilibrado)

Análisis de la estructura

Eficiencias de las distintas operaciones para cada estructura:

	Lista Enlazada	Array Ordenado	Árbol Roji-Negro
put	O(n)	O(n)	O(log n)
get	O(n)	O(log n)	O(log n)
contains	O(n)	O(log n)	O(log n)
keys	O(n)	O(n)	O(n)

- n es el número de pares almacenados en la estructura
- O(n) significa del orden del nº de pares
- O(log n) significa del orden del logaritmo del nº de pares

Demo…

Tiempos de ejecución

- Hardware: i7-3770@3.40GHz, 8Gb
- Software: Fedora 20, Eclipse Luna JDT 4.1.1, OpenJDK 1.8.0_11

tale.txt (135 635 palabras)		
ListKVS	11.5 segundos	
ArrayKVS	0.30 segundos	38 veces más rápida
TreeKVS	0.20 segundos	57 veces más rápida
leipzig100K.txt (2 121 054 palabras)		
ListKVS	3757.5 segundos	
ArrayKVS	24.5 segundos	153 veces más rápida
TreeKVS	2.2 segundos	1700 veces más rápida
leipzig1M.txt (21 191 455 palabras)		
ArrayKVS	1040 segundos	
TreeKVS	20.9 segundos	50 veces más rápida

¿Por qué estudiar Estructuras de Datos?

"The difference between a bad programmer and a good one is whether he considers code or data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships."

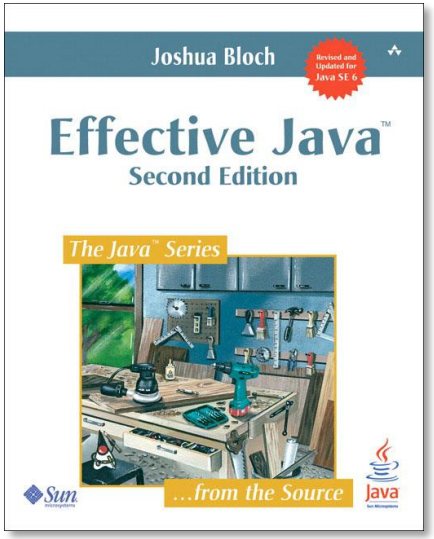
— Linus Torvalds

Objetivos de la Asignatura (II)

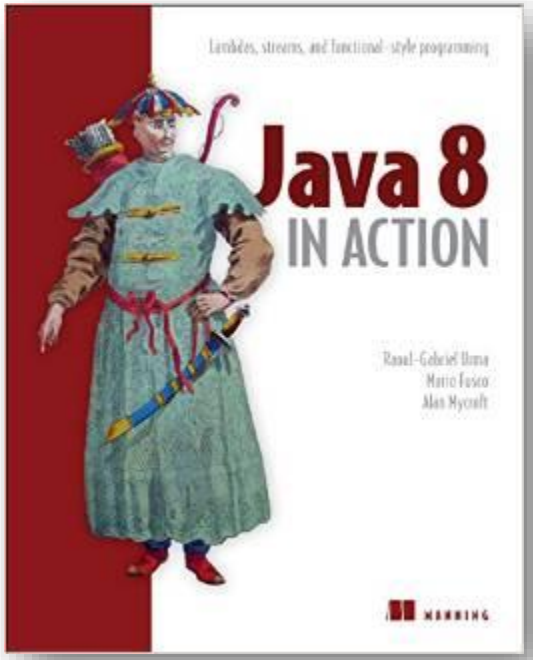
- Hasta ahora habéis usado el estilo **imperativo**
  - C y Java - **datos mutables**
- Esta asignatura introduce el estilo **funcional**
  - Haskell - **datos inmutables**

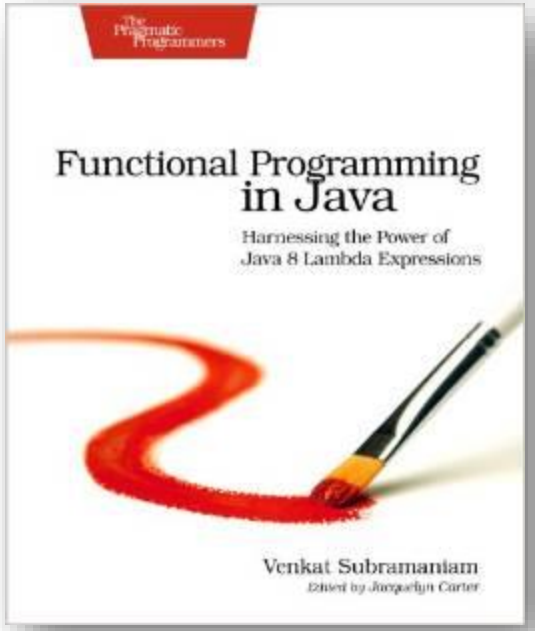
"Classes should be made immutable unless there is a very good reason to make them mutable… If a class cannot be made immutable, you should still limit its mutability as much as possible."

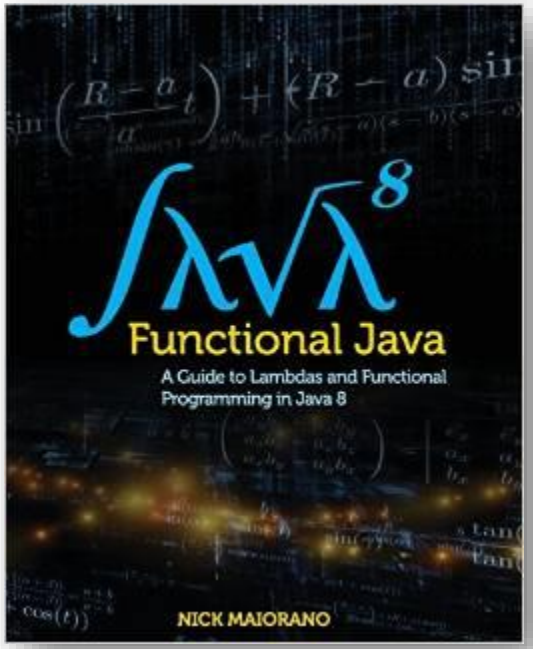
— Joshua Bloch



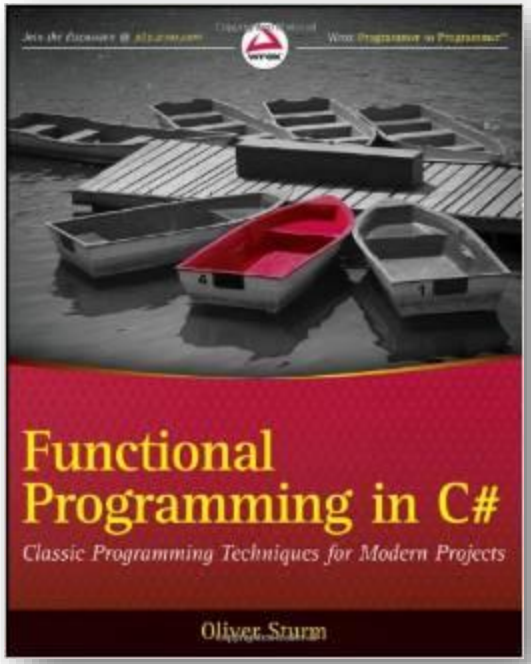
La programación funcional ya está en Java…

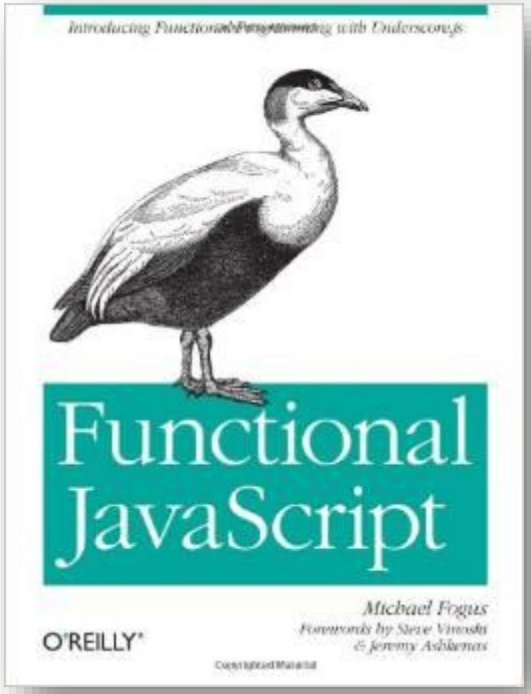


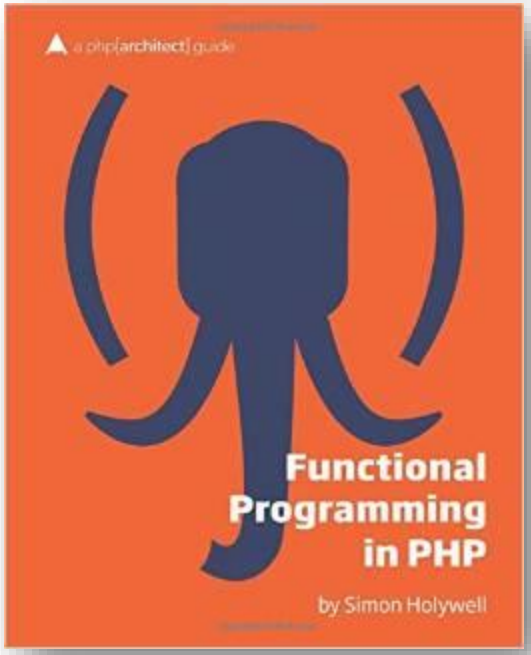




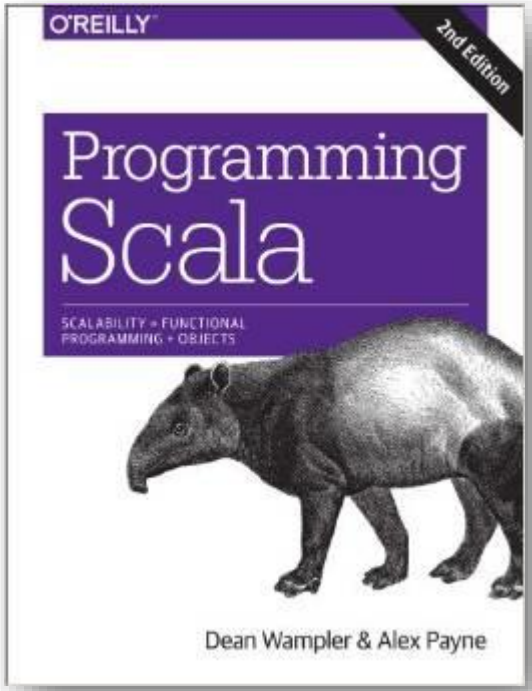
… y en otros lenguajes extendidos

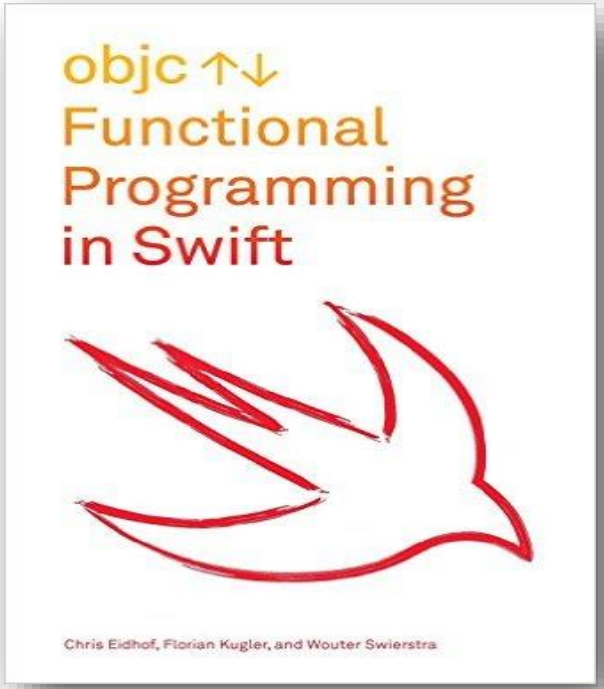


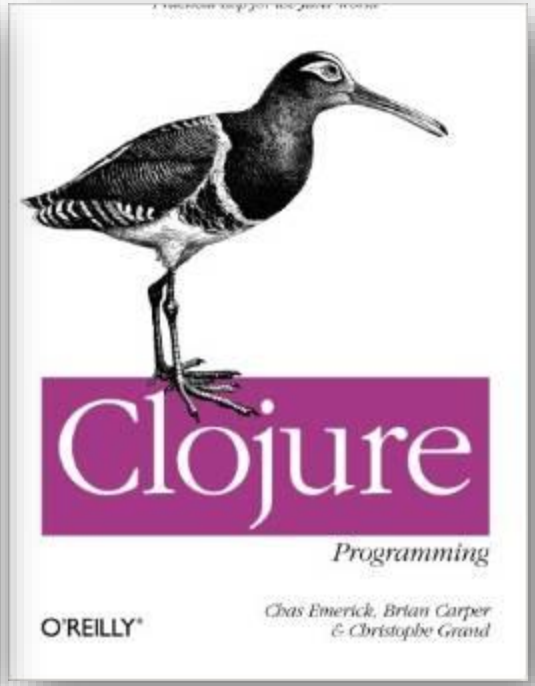




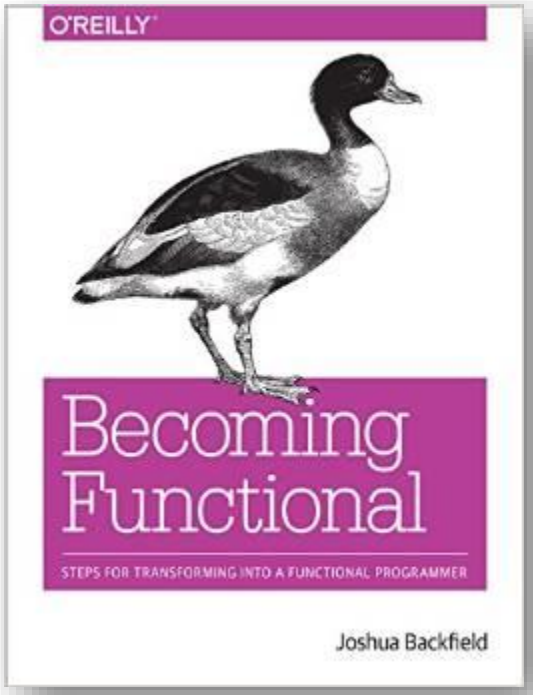
… y en los lenguajes más recientes

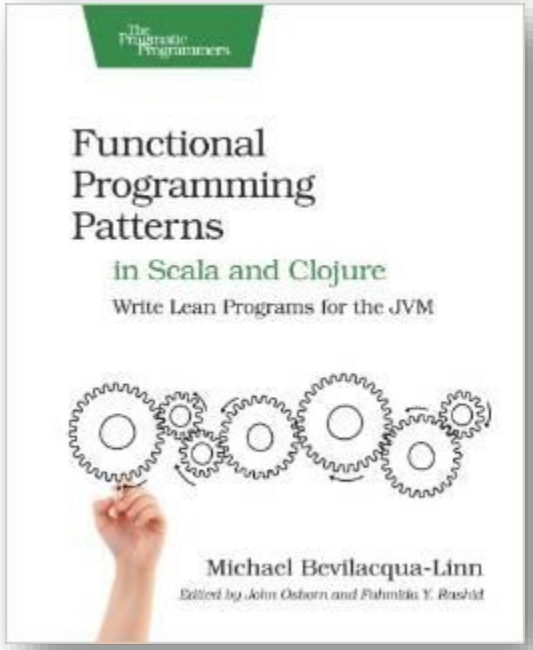


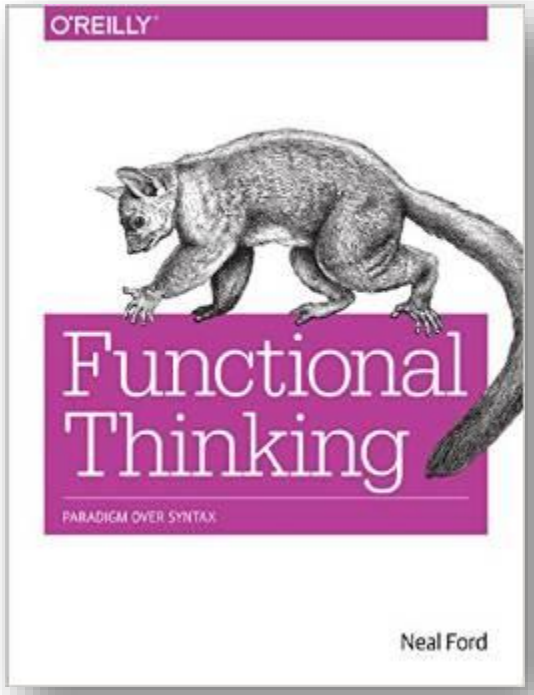




La programación funcional está de moda







Contenido

1. Introducción al estilo Funcional.
2. Características de la Programación Funcional.
3. Introducción a los Tipos Abstractos de Datos. Implementaciones lineales.
4. Árboles.
5. Tablas Hash.
6. Grafos.

Evaluación en la Primera Convocatoria (I)

- Evaluación Continua
  - obligatoria para estudiantes a tiempo completo
  - opcional para el resto de estudiantes
  - **PE**: dos prácticas de laboratorio evaluables, **3 puntos**
    - dos días de clase, la fechas se anunciarán previamente
  - **EF**: examen final **7 puntos**
    - el examen final consta de **2 partes** (Haskell y Java)
    - hay que obtener **al menos un 25%** en cada parte
  - **PE + EF >= 5** para aprobar

Evaluación en la Primera Convocatoria (II)

- Evaluación por Examen Final
  - solo para estudiantes a tiempo parcial, deportistas de alto rendimiento
  - debe solicitarse por **escrito** y con **antelación**
  - examen final exhaustivo con 2 partes, **al menos un 2,5 sobre 10** en cada parte
  - **al menos un 5 de media** para aprobar

Evaluación en la Segunda Convocatoria

- Evaluación Continua
  - si el estudiante ha seguido evaluación continua y quiere **conservar** la calificación **PE**, se aplica el esquema de evaluación continua de la primera convocatoria

- Evaluación por Examen Final
  - estudiantes que **renuncien** a la evaluación continua, a tiempo parcial, deportistas de alto rendimiento
  - examen final exhaustivo con 2 partes, **al menos un 2,5 sobre 10** en cada parte; **al menos un 5 de media** para aprobar

Evaluación en la Convocatoria Extraordinaria

- Evaluación por Examen Final
  - todos los estudiantes (no se contempla evaluación continua)
  - examen final exhaustivo con 2 partes, **al menos un 2,5 sobre 10** en cada parte
  - **al menos un 5 de media** para aprobar

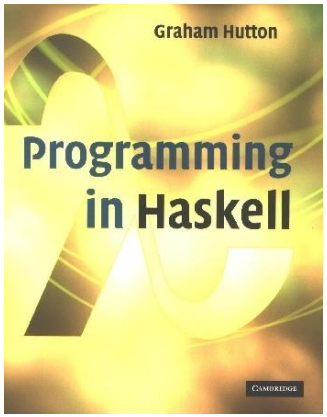
Software

- Campus Virtual (sala común)

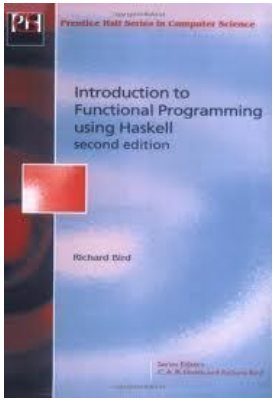
Bibliografía sobre PF y Haskell

- Graham Hutton.
  Programming in Haskell.
  Cambridge University Press, 2007

  http://www.cs.nott.ac.uk/~gmh/book.html

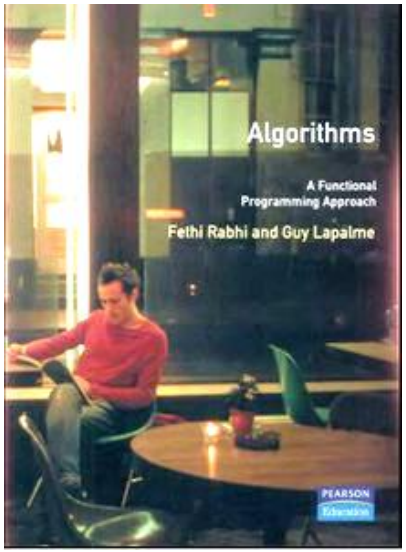


- Richard Bird.
  Introduction to Functional Programming using Haskell.
  Prentice Hall, 1998



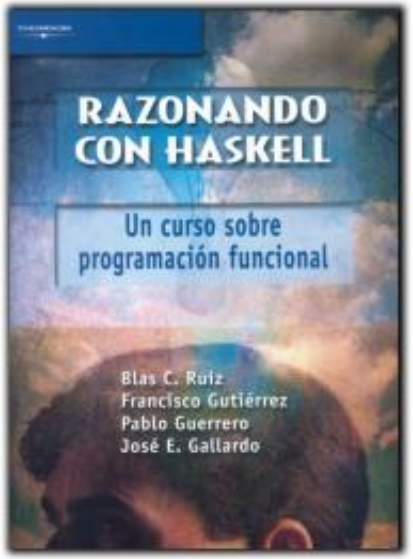
Bibliografía sobre PF y Haskell

- Fethi A. Rabhi & Guy Lapalme. ***Algorithms: a Functional Programming Approach***. Addison Wesley, 1999



http://www.iro.umontreal.ca/~lapalme/Algorithms-functional.html

- Blas Ruiz, Francisco Gutiérrez, Pablo Guerrero Y José Gallardo. ***Razonando con Haskell: un curso sobre Programación Funcional***. Thomson, 2004



http://www.lcc.uma.es/~pepeg/pfHaskell/index.html

Bibliografía sobre PF y Haskell (En la Web)

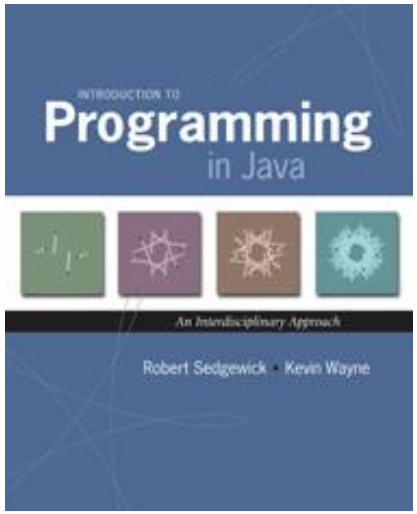
- *Learn You a Haskell* for Great Good!

http://learnyouahaskell.com

http://aprendehaskell.es

Bibliografía sobre POO y Java (En la Web)

- Robert Sedgewick & Kevin Wayne. ***Introduction to Programming in Java: An Interdisciplinary Approach***.
  Addison Wesley, 2007



http://introcs.cs.princeton.edu/java/home/

- Robert Sedgewick & Kevin Wayne. ***Algorithms. 4th Edition***.
  Addison Wesley, 2010

http://algs4.cs.princeton.edu/home/

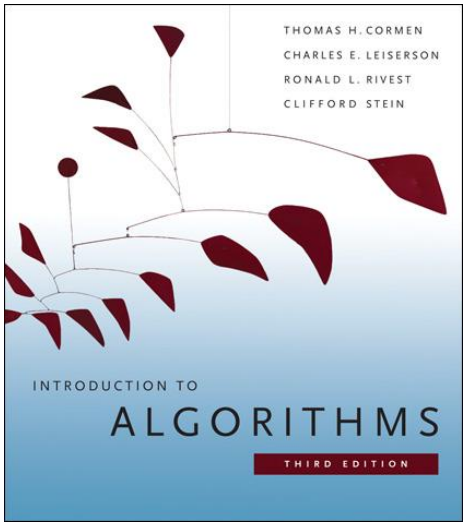
Bibliografía sobre POO y Java

- Francisco Gutiérrez, Francisco Durán Ernesto Pimentel.
  Programación Orientada a Objetos con Java.
  Thomson, 2007

http://dgp.lcc.uma.es

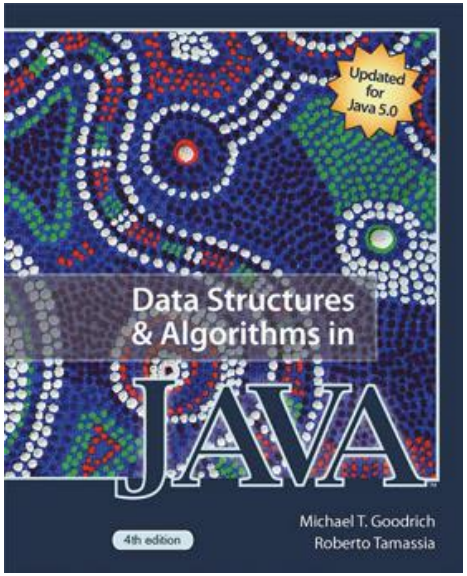
Bibliografía sobre POO y Java

- T.H. Cormen, C.E. Leiserson, R.L. Rivest & C. Stein. ***Introduction to Algorithms*, 3rd Ed.** McGraw-Hill, 2009



https://mitpress.mit.edu/books/introduction-algorithms-third-edition

- Michael T. Goodrich & Roberto Tamassia. **Data Structures & Algorithms in Java, 4th Edition**. John Wiley & Sons, 2006



https://www.wiley.com/en-us/Data+Structures+and+Algorithms+in+Java%2C+6th+Edition-p-9781118771334

Bibliografía sobre POO y Java (En la Web)

- Duane A. Bailey.
  ***Java Structures*. *Data Structures in Java, for the Principled Programmer*.**

http://www.cs.williams.edu/~bailey/JavaStructures