# Examen Final: Laboratorio 2

Paradigmas de la Programación - 17/12/2025

## Generales

- Para codear el parcial solo podrán utilizar algún editor de texto plano o IDE sin asistencia de IA. No utilizar teléfonos durante el examen. No acceder a sitios que permitan comunicarse con otras personas como whatsapp, discord, zulip, etc. **No respetar esta regla implica desaprobación inmediata del final y será calificada con 0 en el acta del examen.**
- Pueden utilizar sus propias laptops o alguna máquina del laboratorio que tenga instalado todo lo necesario para el lab2 (openjdk v21, etc).
- Pueden tener el código fuente del laboratorio 2 para consulta pero **NO** deben entregarlo. Solo entregar el código con la solución del examen. Resta puntos entregar el código de los laboratorios.
- **Verificar que se haya recibido el correo de confirmación** luego de la entrega**.** Si no llega, deben avisar antes de retirarse del aula. **No se aceptarán reclamos posteriores por entregas incorrectas o incompletas.**

El examen consiste en ir agregando con cada ejercicio, una nueva funcionalidad a nuestro FeedReader. Pero al ser un examen de POO, no solo es importante lograr la funcionalidad que se requiere (condición necesaria pero no suficiente) sino como modelan su solución a través de los conceptos de POO (clases, herencia, interfaces, modificadores de scope, etc), es decir, es importante la calidad de su solución.

## Esqueleto

Descargar del siguiente link:

Si al descomprimir se encuentran con archivos que comienzan con "._" deben borrarlos a todos, incluso en sub-directorio. Piensen y busquen un script que lo haga por ustedes

17-12-25-lab2-skeleton.tar.xz

Otras librerías que pueden ser útiles

json-20231013.jar

jsoup-1.21.1.jar

## Instrucciones para Compilar y Correr

Para compilar y ejecutar el código del examen, solo es necesario utilizar el Makefile provisto desde la carpeta del esqueleto:

```
None
$ make run ARGS="-h quick"
```

El resultado es similar a:

```
→  skeleton make run ARGS="-h quick"
java -cp "bin:lib/jsoup-1.21.1.jar:lib/json-20231013.jar:"
************* FeedReader version 2.0 *************
Named Entities Found:
Top 20 Named Entities by Frequency:
Engineering (OTHER): 101
Daily</a></p> (OTHER): 100
AI (OTHER): 34
GPIO (OTHER): 23
TI (OTHER): 18
GitLab (OTHER): 16
Software (OTHER): 16
However (OTHER): 15
Reolink (OTHER): 14
Allwinner (OTHER): 13
USB (OTHER): 13
Wi-Fi (OTHER): 13
Pi&rsquos (OTHER): 12
Linux (OTHER): 11
Programmer</a></p> (OTHER): 10
Contents</label> (OTHER): 10
Crazy (OTHER): 10
Gregor (OTHER): 10
Best (OTHER): 10
RAM (OTHER): 10
```

Notar que el esqueleto del examen difiere del código del laboratorio en varios puntos. En particular:

- Para asignar a cada palabra una categoría de entidad nombrada se utiliza el método Heuristic.getCategory. Si la palabra no es una entidad nombrada, entonces devuelve null.
- Para obtener las entidades nombradas de un artículo, se utiliza el método Heuristic.getNamedEntities

# Tech Radar / Technology Classifier

El objetivo del examen es convertir el FeedReader en una herramienta que escanee blogs técnicos y genere un "Radar Tecnológico", diciéndonos de qué tecnologías se está hablando más (Lenguajes, Infraestructura, Bases de Datos).

## Ejercicio 1. Heurística de Clasificación Tecnológica (Map Lookup)

Crear una nueva heurística llamada TechnologyHeuristic. La detección debe hacerse mediante comparación exacta de cadenas (ignorando mayúsculas/minúsculas) contra una base de conocimientos en memoria.

**Requerimientos:**

- La heurística se debe seleccionar con la opción `-h technology` al ejecutar el programa.
- Deben eliminarse las **stopwords** de la misma forma que en las otras heurísticas (no deben clasificarse ni contarse si aparecen en el texto).
- Utilizar un `Map<String, String>` donde la **Clave** es la palabra (ej: "python") y el **Valor** es la categoría (ej: "LANGUAGE").
- Al instanciar la clase, se deben cargar (hardcodeado por ahora) las siguientes asociaciones:
    - **LANGUAGES:** "java", "c++", "python", "rust", "javascript", "kotlin".
    - **INFRA:** "docker", "kubernetes", "aws", "linux", "jenkins".
    - **DATABASE:** "mysql", "postgres", "mongodb", "redis", "sql".
    - **CONCEPTS:** "algorithm", "recursion", "api", "framework", "compiler".

**Lógica esperada:** El método getCategory(word) debe normalizar la palabra (a minúsculas) y buscarla en el Mapa. Si existe, devuelve la categoría. Si no, devuelve null.

**Salida esperada:**

```
************* FeedReader version 2.0 *************
Named Entities Found:
Top 14 Named Entities by Frequency:
Linux (INFRASTRUCTURE): 11
Python (LANGUAGE): 8
C (LANGUAGE): 7
JavaScript (LANGUAGE): 7
API (CONCEPT): 6
rest (CONCEPT): 3
SQL (DATABASE): 3
Java (LANGUAGE): 2
Rust (LANGUAGE): 2
Redis (DATABASE): 2
Docker (INFRASTRUCTURE): 2
Kubernetes (INFRASTRUCTURE): 2
docker (INFRASTRUCTURE): 1
REST (CONCEPT): 1
```

## Ejercicio 2. Top3

Modificar el reporte para que se impriman las 3 entidades con mayor frecuencia por categoría. Se pide imprimir la **Categoría**, y seguido las tres entidades con la posición, el nombre y la cantidad.

Por ejemplo, desde la imagen anterior la categoría INFRASTRUCTURE mostrará Linux como primera entidad con una cantidad 11, seguida por Docker con 2 y finalmente con Kubernetes también con 2, tal y como se muestra en la siguiente imagen.

```
************* FeedReader version 2.0 *************
Category: LANGUAGE
(1) C: 7
(2) JavaScript: 7
(3) Python: 7

Category: CONCEPT
(1) API: 6
(2) rest: 3
(3) REST: 1

Category: DATABASE
(1) SQL: 3
(2) Redis: 2

Category: INFRASTRUCTURE
(1) Linux: 11
(2) Docker: 2
(3) Kubernetes: 2
```

## Ejercicio 3. Carga de Base de Conocimiento (Properties/Map)

En lugar de hardcodear el mapa, leer el archivo config/tech_stack.properties. Formato del archivo:

Properties

```
None
java=LANGUAGE
docker=INFRA
react=FRONTEND
...
```

Dicho archivo debe contener todas las entidades antes mencionadas. El alumno debe leer el archivo, hacer el split por el = y llenar el mapa. Nuevamente recuerden que es importante el lugar del código donde deciden poner esta funcionalidad y cómo la modularizan.

Se pide modificar el sistema para que imprima la frecuencia de cada categoría de entidades nombradas. Por ejemplo, en el caso anterior el resultado sería:

```
None
LANGUAGE: 5
INFRASTRUCTURE: 4
CONCEPT: 3
DATABASE: 2
```

Luego de compilar el ejercicio 3, el resultado es similar al siguiente ejemplo:

```
************* FeedReader version 2.0 *************
Named Entities Found:
LANGUAGE: 5
INFRASTRUCTURE: 4
CONCEPT: 3
DATABASE: 2
```

Notar que con cada heurística, el resultado es distinto:

```
************* FeedReader version 2.0 *************
Named Entities Found:
PERSON: 2543
COUNTRY: 2485
OTHER: 2441
```

Nota: no deben eliminar el código existente.

## Ejercicio 4. Limpiar los tags `html` de los artículos (solo para alumnos libres)

La consigna de este ejercicio es limpiar los tags HTML de los artículos antes de procesar su contenido para extraer entidades nombradas.

El ejercicio se resuelve de forma sencilla utilizando las líneas:

```
None
import org.jsoup.Jsoup;

Jsoup.parse(htmlText).text();
```

Esto eliminará todos los tags y dejará solamente el texto plano, evitando problemas de tokens como `<a href="...">` , `<br>`, ` `, etc.

Lo más importante del ejercicio no es la línea de código en sí, sino decidir en qué clase y/o método colocar esta limpieza de HTML de forma consistente con los principios de POO. Heuristic? Parser? Article? Feed? FeedReaderMain? En el formulario de entrega del examen, debe responder la siguiente pregunta:

> **¿Dónde se agregó el código? Utilice los principios de POO para justificar su elección**

Luego de implementar el ejercicio, el resultado (sin la modificación del ejercicio 1) se verá como:

```
************* FeedReader version 2.0 *************
Named Entities Found:
Top 20 Named Entities by Frequency:
Software (OTHER): 105
Engineering (OTHER): 101
Daily (OTHER): 100
AI (OTHER): 34
GPIO (OTHER): 29
TI (OTHER): 19
GitLab (OTHER): 18
However (OTHER): 16
Reolink (OTHER): 14
Allwinner (OTHER): 14
USB (OTHER): 13
Pis (OTHER): 12
RAM (OTHER): 12
Wi-Fi (OTHER): 12
Plus (OTHER): 11
Linux (OTHER): 11
Building (OTHER): 11
RTSP (OTHER): 11
Ghidra (OTHER): 11
APIs (OTHER): 10
```

Nota: recordá usar la heurística quick y el método original de reporte.

## Entrega

Comprimir el directorio raíz de su examen en un archivo “tar.xz” con el siguiente formato “Apellido_Nombre.tar.xz”. Para que quede con el formato adecuado, usar los siguientes comandos:

```shell
mv skeleton Juarez_MatiasAlfonso
tar -czvf Juarez_MatiasAlfonso.tar.xz Juarez_MatiasAlfonso
```

Adjuntar el comprimido en el formulario de entrega, llenar todos los campos del mismo y submitear con su correo institucional. Por último, chequear en su bandeja de entrada si recibieron el mail de confirmación de entrega realizada, gracias.