Laboratorio - Recuperatorio Parcial 2
Programación Orientada a Objetos en Java

Generales

- Para codear el parcial solo podrán utilizar algún editor de texto plano o IDE (sin asistencia). No utilizar teléfonos durante el examen. Por favor, ser honestos intelectualmente y no poner al docente en una situación incómoda. 🙏
- Pueden utilizar sus propias laptops o alguna máquina del laboratorio que tenga instalado todo lo necesario para el lab2 (openjdk, ant, etc).
- Pueden tener el código fuente del laboratorio 2 para consulta pero NO deben entregarlo, solo el código fuente del examen deben submitear.
- NO compila => Reprobado

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.

Instrucciones para Compilar y Correr

Para compilar y ejecutar el código del examen, debemos tener instalado una herramienta llamada “Ant” la cual facilita la gestión de proyectos en java con muchos archivos. Para instalarlo, ejecutamos el siguiente comando en la terminal de linux:

```
$ sudo apt install ant.
```

Y luego, verificamos si la instalación fue exitosa:

```
$ ant -version
```

```
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial2/skeleton$ ant -version
Apache Ant(TM) version 1.10.14 compiled on September 25 2023
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial2/skeleton$
```

Ant mediante el seteo de un archivo de configuración (build.xml) se encarga de compilar y linkear todo el proyecto llamando “por detrás” al comando “javac” y “java” que deben estar ya instalados en el sistema cuando instalaron una JDK.

La cátedra provee en la raíz del skeleton el archivo “build.xml” ya configurado para compilar y correr todo el parcial (o sea, no deben preocuparse por su configuración).

Para chequear que esta todo OK para empezar a codear el parcial nos posicionamos en el directorio donde se encuentra “build.xml” y ejecutamos en la terminal:

```
$ ant run_ok
```

Y si vemos al final de la terminal el mensaje **“TODO OK PARA EL PARCIAL, gracias.”** entonces tenemos todo listo para empezar a trabajar.

Más tarde, para compilar y correr el código fuente del ejercicio X nos posicionamos en el directorio donde se encuentra “build.xml” y ejecutamos en la terminal:

```
$ ant run_ejX
```

Para eliminar los binarios generados por una compilación ejecutamos en la terminal:

```
$ ant clean
```

Ejercicio 1. Impresión de Feeds con condiciones

En el fichero “config/*subscriptions_ej1.json”* se encuentra un tercer “feature” en cada suscripción llamado “content”. En caso de ser **True**, la aplicación debe realizar el pretty print del feed completo. En caso de ser **False**, imprimir los artículos del feed sin el campo *description*. Por ejemplo, si tenemos:

```
[
    {
        "url": "https://rss.nytimes.com/services/xml/rss/nyt/Business.xml",
        "urlType": "rss",
        "content": "True"
    },
    {
        "url": "https://rss.nytimes.com/services/xml/rss/nyt/Technology.xml",
        "urlType": "rss",
        "content": "False"
    }

]
```

Entonces, los artículos del feed “Business” deben ser mostrados completos. Mientras que los artículos del feed “Technology” deben ser mostrados de forma tal que no impriman el campo *description*.

Para compilar y ejecutar el ejercicio 1, hacemos:

```
$ ant run_ej1
```

Ejercicio 2. Nueva Heurística para detectar Artículos

Implementar una nueva heurística para detectar artículos en el texto, llamada “ArticleHeuristic” que clasifica una palabra si esta es “a”, “an”, “the” y sus variantes con mayúscula en la primera letra.

Para compilar y ejecutar el ejercicio 2, hacemos:

```
$ ant run_ej2
```

Veremos por pantalla los artículos que se encuentran en los textos del feed “Business” . La salida es como la mostrada a continuación:

```
[java]
[java] The --> 16
[java] the --> 63
[java] a --> 39
[java] A --> 4
[java] an --> 4
```

Ejercicio 3. Parseo local de Feeds atom y conteo de entidades

Implementaremos un nuevo tipo de servidor de feed en el directorio **localhost** para feeds de tipo **atom**:

```
[
    {
        "url": "localhost/book_feed.xml",
        "urlType": "atom"
    },
    {
        "url": "localhost/cs.xml",
        "urlType": "atom"
    }
]
```

cuya respuesta es un xml con el formato que se muestra a continuación :

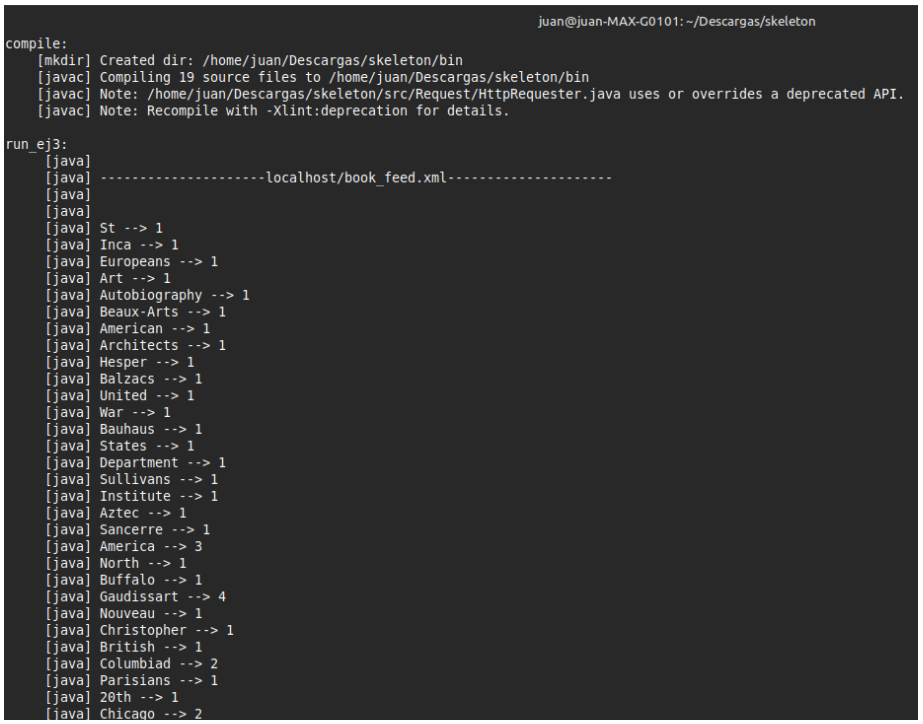
```
<feed>
  <entry>
    <id>oai:arXiv.org:2506.06282v1</id>
    <title>Understanding Financial Reasoning in AI: A Multimodal Benchmark and Error Learning Approach</title>
    <updated>2025-06-11T01:34:17.392181+00:00</updated>
    <link>https://arxiv.org/abs/2506.06282</link>
    <summary>Abstract: Effective financial reasoning demands not only textual understanding but also the ability to interpret complex visual data such as charts, tables, and trend graphs. This paper introduces a new benchmark designed to evaluate how well AI models -- especially large language and multimodal models -- reason in finance-specific contexts. Covering 3,200 expert-level question-answer pairs across 15 core financial topics, the benchmark integrates both textual and visual modalities to reflect authentic analytical challenges in finance. To address limitations in current reasoning approaches, we propose an error-aware learning framework that leverages historical model mistakes and feedback to guide inference, without requiring fine-tuning. Our experiments across state-of-the-art models show that multimodal inputs significantly enhance performance and that incorporating error feedback leads to consistent and measurable improvements. The results highlight persistent challenges in visual understanding and mathematical logic, while also demonstrating the promise of self-reflective reasoning in financial AI systems. Our code and data can be found at https://anonymous/FinMR/CodeData.</summary>
    <published>2025-06-10T00:00:00-04:00</published>
  </entry>
<entry>
    <id>oai:arXiv.org:2506.06283v1</id>
    <title>Facial Foundational Model Advances Early Warning of Coronary Artery Disease from Live Videos with DigitalShadow</title>
    <updated>2025-03-11T01:34:17.392013+00:00</updated>
    <link> https://arxiv.org/abs/2506.06283</link>
    <summary>Abstract: Global population aging presents increasing challenges to healthcare systems, with coronary artery disease (CAD) responsible for approximately 17.8 million deaths annually, making it a leading cause of global mortality. As CAD is largely preventable, early detection and proactive management are essential. In this work, we introduce DigitalShadow, an advanced early warning system for CAD, powered by a fine-tuned facial foundation model. The system is pre-trained on 21 million facial images and subsequently fine-tuned into LiveCAD, a specialized CAD risk assessment model trained on 7,004 facial images from 1,751 subjects across four hospitals in China. DigitalShadow functions passively and contactlessly, extracting facial features from live video streams without requiring active user engagement. Integrated with a personalized database, it generates natural language risk reports and individualized health recommendations. With privacy as a core design principle, DigitalShadow supports local deployment to ensure secure handling of user data.</summary>
<published>2025-02-10T00:00:00-04:00</published>
  </entry>
```

Realizar todo lo necesario para que este nuevo tipo de feed pueda ser consumido por nuestra aplicación y después con cada suscripción contar las entidades nombradas y los artículos y mostrarlos por pantalla.

Para compilar y ejecutar el ejercicio 3, hacemos:

```
$ ant run_ej3
```

Veremos por pantalla la cantidad de artículos y entidades nombradas de los feed locales de libros y de publicaciones de arxiv. El formato de salida debe ser como se muestra en la siguiente imagen:



Debe encabezar con el nombre del feed y luego debe imprimir las entidades nombradas, además de los artículos con sus correspondientes cantidades. Esto debe ser por cada uno de los feeds presentes en el archivo *subscriptions_ej3.json* presente en el directorio **config**.

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:

```
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.