**Laboratorio 2:**

**Programación Orientada a Objetos**

Paradigmas 2025 - FAMAF

Revisión 2025: Juan Costamagna

## **En este laboratorio vamos a implementar un lector automático de feeds, como aplicación de consola, bajo el paradigma orientado a objetos, utilizando como lenguaje de programación Java.**

Instalación

Instalar un java development kit (JDK):

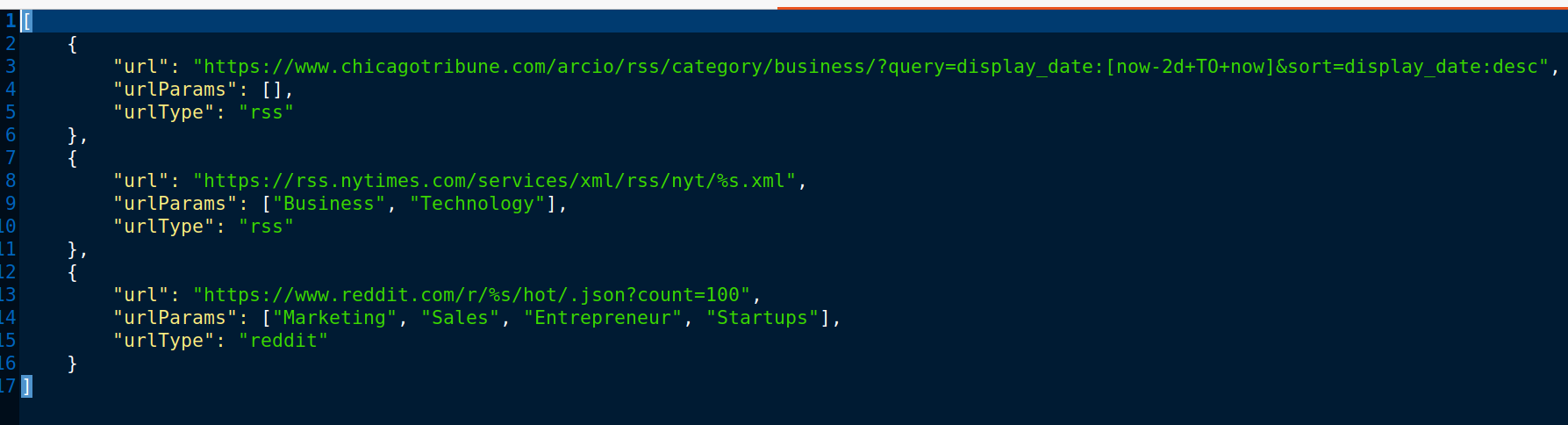
$sudo apt install openjdk-24-jdk.

Funcionalidad

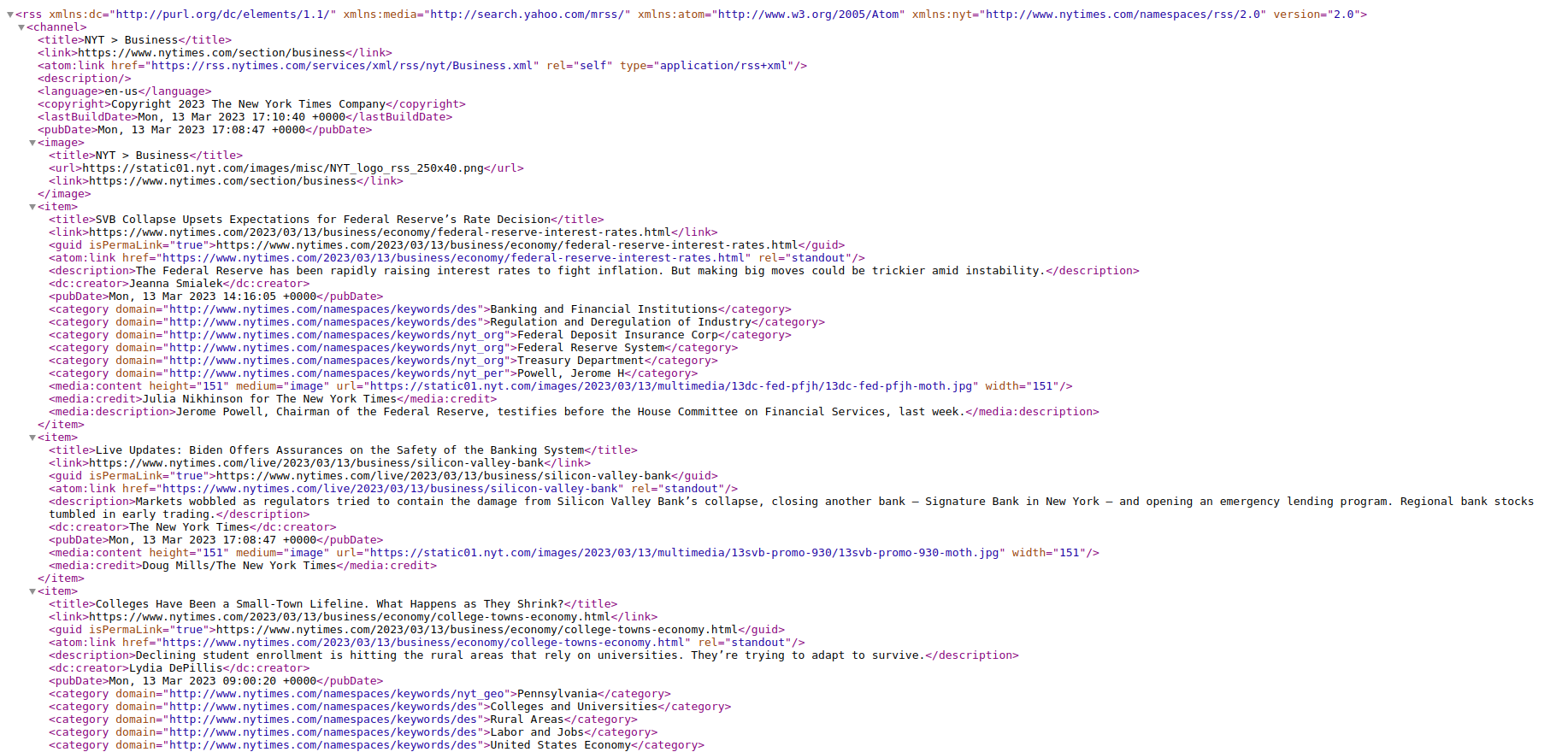
El usuario de nuestro lector establece, mediante un archivo en formato .json, los distinto sitios (ej: new york times, etc) y sus respectivos tópicos (sports, business, etc) de los cuales obtener los diferentes feeds a mostrar por pantalla en forma legible y amigable para el usuario. Además, se agrega una funcionalidad a nuestro lector para computar heurísticamente las entidades nombradas más mencionadas en la lista de feeds.

Pipeline de Trabajo de la Aplicación (Tentativo)

1. Obtener las url’s concretos de los distintos feeds del archivo de suscripción (o sea, obtener qué feeds tenemos que consumir). Para ello van a tener que parsear un archivo en formato json, utilizando alguna libreria ya disponible para ello: <https://www.w3docs.com/snippets/java/how-to-parse-json-in-java.html>



1. Luego realizar la consulta “http request” al servidor del feed. El servidor en caso de éxito, devolverá un feed RSS, que es simplemente una interfaz (en formato xml) para obtener el contenido de los diferentes artículos del feed. Por ejemplo, si queremos obtener el feed de “business” del “sitio New York Times”, en el navegador ponemos la siguiente url: <https://rss.nytimes.com/services/xml/rss/nyt/Business.xml>



1. Una vez conseguidos la lista de artículos de un feed, van a extraer sólo aquellos atributos del artículo (item) que nos interesa (title, description,pubDate,link ) que luego serán mostrados por pantalla en forma legible para el usuario. Para ello van a tener que parsear formato xml con alguna libreria para ello: <https://www.tutorialspoint.com/java_xml/java_dom_parse_document.htm>

Notar que deben parsear archivos json y xml, por lo tanto, deben tratar de generalizar la noción de parser en la aplicación (hint: pensar en herencia).

1. Por defecto, Imprimir por pantalla los artículos del feed en forma legible y agradable para el usuario y terminar la aplicación. El ejecutable del lector de feed tendrá una funcionalidad extra (via parámetro por consola, por ejemplo, FeedReader **-ne** ), que computa mediante una heurística, un conjunto de entidades nombradas (*named entities*) de los feeds y su cantidad de ocurrencias, global, por clase y por subclase, para su posterior impresión por pantalla en forma de tabla.   
   La cátedra ofrecerá dos heurísticas para ello, “QuickHeuristic” y “RandomHeuristic,” muy simples, para [detectar entidades nombradas](https://es.wikipedia.org/wiki/Reconocimiento_de_entidades_nombradas). Ustedes deben programar la aplicación de tal manera que cambiar de una a otra no implique un gran cambio en el código de la aplicación. (hint: pensar en herencia).

ATENCIÓN

Las entidades nombradas están organizadas en una jerarquía de clases, como sigue:

entidad nombrada → persona → apellido

→ nombre

→ título

→ lugar → país

→ ciudad

→ dirección

→ otro

→ organización

→ producto

→ evento

→ fecha

→ otro

los atributos específicos **mínimos** (pueden añadir otros que los ayuden en el conteo) para cada una de estas clases son:

* entidad nombrada: string, nombre canónico
* persona: ID de la persona
* apellido: forma canónica, origen
* nombre: forma canónica, origen, formas alternativas
* título: forma canónica, profesional
* país: población, lengua oficial
* ciudad: país, capital, población
* dirección: ciudad
* organización: forma canónica, número de miembros, tipo de organización
* producto: comercial, productor
* evento: forma canónica, fecha, recurrente
* fecha: precisa, forma canónica
* otro: comentarios

Adicionalmente, existe otra jerarquía que también se aplica a las entidades nombradas, de forma ortogonal a la primera, es decir, conceptualmente como si fuera herencia múltiple:

tema → deportes → fútbol

→ básquet

→ tenis

→ fórmula 1

→ otros

→ cultura → cine

→ música

→ otros

→ política → nacional

→ internacional

→ otros

→ otros

### Recomendaciones

* Leer un tutorial o libro de Java antes de comenzar.
* Identificar en su aplicación cada mini-funcionalidad y mapearla a una clase que la abstraiga y la encapsule. “Mientras más clases, mejor”. No importa si una clase tiene un solo atributo y/o método, ya que, ésta inicialmente podría ser simple pero en el futuro podría crecer producto de nuevas funcionalidades.
* Testear cada mini-funcionalidad o clase por separada y a medida que las van implementado, es decir, construir una solución entera, conlleva construir pequeñas soluciones paso a paso y no pretender que todo funcione correctamente al último sin haber testeado cada parte individual de la aplicación.
* Identificar qué mini-funcionalidades son del mismo tipo o naturaleza para poder generalizarlas en una superclase utilizando herencia. En una superclase se modela aquellos atributos y métodos comunes a todas las subclases, quedando en cada subclase exclusivamente aquellos atributos y métodos propios o particulares de una subclase.
* No dar por sentado que el código del skeleton provisto por la cátedra está libre de bugs. Además pueden modificarlo libremente si ustedes lo consideran necesario.

### Servidores de Feeds

A continuación presentamos una tabla de distintos feeds para utilizar durante el desarrollo de la aplicación:

| Template | Parámetro | URL final |
| --- | --- | --- |
| https://rss.nytimes.com/services/xml/rss/nyt/**%s**.xml ([Más info](https://archive.nytimes.com/www.nytimes.com/services/xml/rss/index.html?mcubz=0)) | Business | https://rss.nytimes.com/services/xml/rss/nyt/Business.xml |
| Technology | https://rss.nytimes.com/services/xml/rss/nyt/Technology.xml |
| https://www.chicagotribune.com/arcio/rss/category/**%s**/?query=display\_date:[now-2d+TO+now]&sort=display\_date:desc | sports | https://www.chicagotribune.com/arcio/rss/category/sports/?query=display\_date:[now-2d+TO+now]&sort=display\_date:desc |
| business | https://www.chicagotribune.com/arcio/rss/category/business/?query=display\_date:[now-2d+TO+now]&sort=display\_date:desc |
| https://www.reddit.com/r/**%s**/hot/.json?count=10 | Android | https://www.reddit.com/r/Android/hot/.json?count=10 |
|  | OddlySatisfaying | https://www.reddit.com/r/OddlySatisfaying/hot/.json? |

Puntos Extras

* Implementar el parser para leer feeds reddit.
* Proponer una nueva heurística para detectar entidades nombradas y comparar su efectividad contra las ofrecidas por la cátedra.
* Agregar al lector de feeds alguna funcionalidad que a ustedes se les ocurra.

Entrega

Via repositorio github del grupo con fecha límite de entrega el dia Viernes 16 de Mayo de 2025.