Práctica
Sistemas Basados en Conocimiento

Existen varios frameworks para gestionar bases de conocimiento en RDF y OWL. Posiblemente dos de las más importantes son APACHE JENA y OWL API. En esta práctica se usará Apache Jena, el cual es un framework de código abierto ampliamente utilizado para construir aplicaciones semánticas basadas en RDF (Resource Description Framework) y OWL (Web Ontology Language). Desarrollado en Java, Jena proporciona herramientas robustas para crear, manipular y consultar datos semánticos, facilitando la implementación de tecnologías de la Web Semántica en diversos contextos.

Para la ejecución de la práctica se recomienda seguir los siguientes pasos:

- **Descargar Apache Jena:** Descargar las librerías desde su sitio oficial (**Apache Jena**). En el menú Download puede localizar en la sección **"**Apache Jena Binary Distributions" el archivo "apache-jena-5.2.0.zip" el cual contiene la distribución binaria de bibliotecas (API), el motor SPARQL, la base de datos RDF nativa TDB y una variedad de scripts de línea de comandos y herramientas para trabajar con estos sistemas.
- **Configurar tu editor de código**: Usar cualquier IDE compatible con Java, como IntelliJ IDEA, o Eclipse.
- **Crear un proyecto en Java**: Asegúrate de incluir las librerías de Jena en el classpath de tu proyecto.

NOTA IMPORTANTE
Es importante mencionar que APACHE JENA maneja dos bibliotecas diferentes que poseen capacidades para trabajar con ontologías OWL/RDF:

Los paquetes de origen en donde se encuentran ambas bibliotecas son:

- JENA CORE: org.apache.jena.ontology.*
- JENA ONTAPI: org.apache.jena.ontapi.*

Las diferencias principales entre los paquetes se resumen en la tabla adjunta:

Aspecto	org.apache.jena.ontology.*	org.apache.jena.ontapi.*
Propósito	API clásica para trabajar con RDF y OWL de forma básica.	API moderna para manejo avanzado de OWL2 y axiomas complejos.
Soporte de OWL	Limitado (OWL básico, clases simples, propiedades básicas).	Completo (OWL2, expresiones complejas, clases anónimas, restricciones avanzadas).
Nivel de abstracción	General y orientado a RDF con compatibilidad OWL básica.	Más detallado y alineado con estándares OWL2.
Modelos manejados	RDF y OWL con capacidades simples.	RDF, OWL2 y modelos con axiomas complejos.
Expresiones avanzadas	Soporte limitado para intersecciones, uniones, o disyunciones.	Soporte completo para operaciones como unión, intersección, restricciones.
Razonamiento	Limitado a funcionalidades básicas de inferencia.	Optimizado para trabajar con razonadores OWL2 avanzados.
Flexibilidad	Adecuado para proyectos simples o básicos.	Ideal para proyectos avanzados que requieren manejo preciso de OWL.
Compatibilidad	Más común en proyectos antiguos.	Recomendado para nuevos proyectos que usen OWL2.
Facilidad de uso	Más sencillo, pero menos funcional para tareas complejas.	Más detallado y poderoso, pero requiere más configuración.

En **conclusión**, se puede usar:

- **`org.apache.jena.ontology.*`** para proyectos que requieren acceder y manipular ontologías representados en RDF y que no necesiten un manejo exhaustivo de OWL2.
- **`org.apache.jena.ontapi.*`** para proyectos avanzados con OWL2 que requieran expresiones complejas, axiomas, y razonamiento detallado.

Para los fines de esta práctica el paquete **org.apache.jena.ontology.*** es más que suficiente.

Parte-1
Usando la API de JENA indicada, se pide crear el modelo sobre las relaciones de un grupo de personas. Implemente una clase en Jena que permita crear el modelo de conocimiento adjunto, suponiendo que el namespace de la ontología se encuentre en **http://example.org/**

1. **Persona** es la clase principal del modelo
2. Agregue que el concepto **Hombre** es un subconcepto de **Persona**
3. Agregue que el concepto **Mujer** es un subconcepto de **Persona**
4. Mejore la definición del concepto Persona indicando que
    - Persona ≡ Hombre ⊔ Mujer
5. Indique que los conceptos **Hombre** y **Mujer** son disjuntos. Esto quiere decir que las instancias que pertenecen a **Hombre** no pueden pertenecer a **Mujer**
6. Agregue las siguientes propiedades objeto, las cuales permitirán modelar las relaciones entre personas.
    - **esCasadoCon** con dominio **Persona** y rango **Persona**
    - **esHermanoDe** con dominio **Persona** y rango **Persona**
    - **esHijoDe** con dominio **Persona** y rango **Persona**
    - **esPadreDe** con dominio **Persona** y rango **Persona**

7. Agregue las siguientes propiedades tipo dato, las cuales permitirán modelar las propiedades del concepto **Persona**
    - **tieneNombre** con dominio **Persona** y rango **String**
    - **tieneEdad** con dominio **Persona** y rango **Integer**

Nota: **String** indica que almacenará datos de tipo texto (Ej. Mauricio Espinoza)
Integer indica que almacenará datos numéricos enteros (Ej. 25)

8. Incorpore las siguientes instancias al concepto **Hombre.** Lo cual indica que **Persona1**, **Persona2** y **Persona3** son **Hombres**
    - Persona1
    - Persona2
    - Persona3

9. Incorpore las siguientes instancias al concepto **Mujer**
    - Persona4

10. Ahora agregue la siguiente información de cada instancia
    - Persona1 tieneNombre “Juan Perez”
    - Persona1 tieneEdad 75
    - Persona2 tieneNombre “Angel Perez”
    - Persona2 tieneEdad 46
    - Persona3 tieneNombre “Vinicio Perez”
    - Persona3 tieneEdad 44
    - Persona4 tieneNombre “Rosario Castillo”

11. Recuerde que, en las bases de conocimiento si se conoce que los individuos son diferentes hay que informarlo al modelo. En este caso sabemos que Persona1, Persona2, Persona3 y Persona4 son personas diferentes. Agregue esta información.

12. Ahora incorpore el siguiente conocimiento al modelo

- Persona1 esPadreDe Persona2
- Persona1 esPadreDe Persona3
- Persona2 esHermanoDe Persona3
- Persona4 estaCasadoCon Persona2

13. Guarde el modelo en formato Turtle, usando como nombre **Modelo1.ttl**.

14. Revise que el código generado se puede abrir en la herramienta PROTEGE

Parte-2

1. Cree una nueva clase en Jena que permita leer el archivo **Modelo1.ttl**

2. Recuerde que en una práctica previa se estableció que para que el modelo pueda inferir que si Persona2 esHermanoDe Persona3, entonces Persona3 esHermanoDe Persona2, se debe agregar la característica simétrica a la propiedad esHermanoDe. Incorpore está característica a las siguientes propiedades:
   - esHermanoDe
   - estaCasadoCon

3. Agregue además al código que la propiedad **esPadreDe** es inversa de **esHijoDe**.

4. Guarde el modelo actualizado en formato **Turtle**, usando como nombre **Modelo2.tll**.

5. Revise que el código generado se puede abrir en la herramienta PROTEGE

Parte-3

1. Cree una nueva clase en Jena que permita leer el archivo **Modelo2.ttl**

2. Incorpore el código en JENA que permita ejecutar la siguiente consulta

   - Mostrar los nombres de los hombres cuya edad sea mayor a 45 años

Parte-4

1. Cree una nueva clase en Jena que permita leer el archivo **Modelo2.ttl**

2. Incorpore el código en JENA que permita ejecutar la siguiente consulta

   - Mostrar los nombres de los hijos de la Persona1

NOTA: La consulta en SPARQL requiere usar obligatoriamente la propiedad **esHijoDe** para lo cual necesite activar un proceso de inferencia en el modelo.