DSCM031Semantic Technologies

Coursework 2: Ontological Modeling

Student: Zornitsa Hristova Hristova
FN: 120168

I've created an OWL Ontology for my microworld **"My Music".** This ontology models a **music microworld**: people, artists, songs, genres, playlists, and how they are related. It is written in **Turtle syntax** for OWL and is based on the following pre-defined

Tasks:

- *Model the vocabulary for your ontology with at least 12 individual objects, 6 classes, 8 class properties and 12 data properties.*
- *Organize the classes into at least two taxonomies and build ate least one taxonomy of properties*

```
:Entity a owl:Class .
:Agent a owl:Class ; rdfs:subClassOf :Entity .
:CreativeWork a owl:Class ; rdfs:subClassOf
:Entity .
```

- Entity is the most general concept.
- Agent = something that can act (people, artists).
- CreativeWork = things that are created (songs, albums, playlists).

This already forms **two taxonomies**:

- **Entity → Agent → Person → Artist**
- **Entity → CreativeWork → Song / Album / Playlist**

People and artists

```
:Person a owl:Class ; rdfs:subClassOf :Agent .

:Artist a owl:Class ; rdfs:subClassOf :Person .
```

- **Every artist is a person.**
- **Every person is an agent.**

Music-related classes

```
:Song a owl:Class ; rdfs:subClassOf :CreativeWork .
:Album a owl:Class ; rdfs:subClassOf :CreativeWork .
:Playlist a owl:Class ; rdfs:subClassOf :CreativeWork .
:Genre a owl:Class .
```

- **Songs, albums, and playlists are creative works.**
- **Genre is a separate classification concept.**

Defined (logical) classes

```
:ShortSong a owl:Class ;
    owl:equivalentClass [
        owl:onProperty :duration ;
        owl:someValuesFrom [ xsd:integer < 200 ]
    ] .
```

- `ShortSong` is **not manually assigned**.
- Any song with `duration < 200` seconds **will be inferred** as a `ShortSong`.
- This is a **necessary and sufficient condition** (`equivalentClass`).

A **property taxonomy** is defined through: likesSong, likesArtist, likesGenre ⊑ likes

This supports reasoning at multiple abstraction levels.

- *Build a logical theory of your microworld with at least 16 axioms.*

The ontology contains **more than 16 axioms**, including:

- Subclass axioms
- Domain and range restrictions
- Functional properties (hasArtist, genreOf)
- Inverse properties

```
:genreOf a owl:FunctionalProperty .
:hasArtist a owl:FunctionalProperty .

:artistOf owl:inverseOf :hasArtist .
:likesArtist owl:inverseOf :isLikedByArtist .
:isLikedByArtist a owl:ObjectProperty ; rdfs:domain :Artist ;
rdfs:range :Person .
```

- Equivalent class definitions (ShortSong)

```
:ShortSong a owl:Class ;
    owl:equivalentClass [ a owl:Restriction ;
        owl:onProperty :duration ;
        owl:someValuesFrom [ a rdfs:Datatype ;
            owl:onDatatype xsd:integer ;
            owl:withRestrictions ( [ xsd:maxExclusive
"200"^^xsd:integer ] )
        ]
    ] .
```

- Property chain axioms

```
# H1: If person likesArtist and artistOf -> likes song
:likes owl:propertyChainAxiom ( :likesArtist :artistOf ) .

# H2: If person likesGenre and genreOf(song,genre) -> likes(song)
:likes owl:propertyChainAxiom ( :likesGenre [ owl:inverseOf :genreOf ]
) .

# H3: If playlistGenre and genreOf(song,genre) ->
inPlaylist(song,playlist)
:inPlaylist owl:propertyChainAxiom ( [ owl:inverseOf :genreOf ]
:playlistGenre ) .
```

- *Formulate at least 6 different rules using SWRL for modeling some of the constraints and policies of your microworld*

  At least **6 SWRL rules** model policies and constraints:

  1. Liking an artist implies liking their songs

```
# Rule 1 (H1): If person likesArtist and artistOf -> likes(song)
[] a owl:Axiom ; rdfs:comment "Rule1: likesArtist(?p, ?a) ^
artistOf(?a, ?s) -> likes(?p, ?s)" .
```

  2. Liking a genre implies liking songs of that genre

```
# Rule 2 (H2): If person likesGenre and genreOf -> likes(song)
[] a owl:Axiom ; rdfs:comment "Rule2: likesGenre(?p, ?g) ^ genreOf(?s,
?g) -> likes(?p, ?s)" .
```

3. Genre-based playlist assignment

```
# Rule 3 (H3): If playlistGenre and genreOf -> inPlaylist
[] a owl:Axiom ; rdfs:comment "Rule3: playlistGenre(?pl, ?g) ^
genreOf(?s, ?g) -> inPlaylist(?s, ?pl)" .
```

4. Short songs are added to chill playlists

```
# Rule 4 (H4): short duration -> in Chill playlist
[] a owl:Axiom ; rdfs:comment "Rule4: duration(?s, ?d) ^
swrlb:lessThan(?d, 200) -> inPlaylist(?s, ChillAtmospheric)" .
```

5. Popularity inferred from play count or rating

```
# Rule 5: high playCount or high rating -> PopularSong
[] a owl:Axiom ; rdfs:comment "Rule5: playCount(?s, ?c) ^
swrlb:greaterThan(?c, 10000) -> PopularSong(?s)" .
[] a owl:Axiom ; rdfs:comment "Rule5b: rating(?s, ?r) ^
swrlb:greaterThan(?r, 4.5) -> PopularSong(?s)" .
```

6. Songs by awarded artists become award-winning songs

```
# Rule 6: artist has award -> award-winning song
[] a owl:Axiom ; rdfs:comment "Rule6: hasAward(?a, ?w) ^ artistOf(?a,
?s) -> AwardWinningSong(?s)" .
```