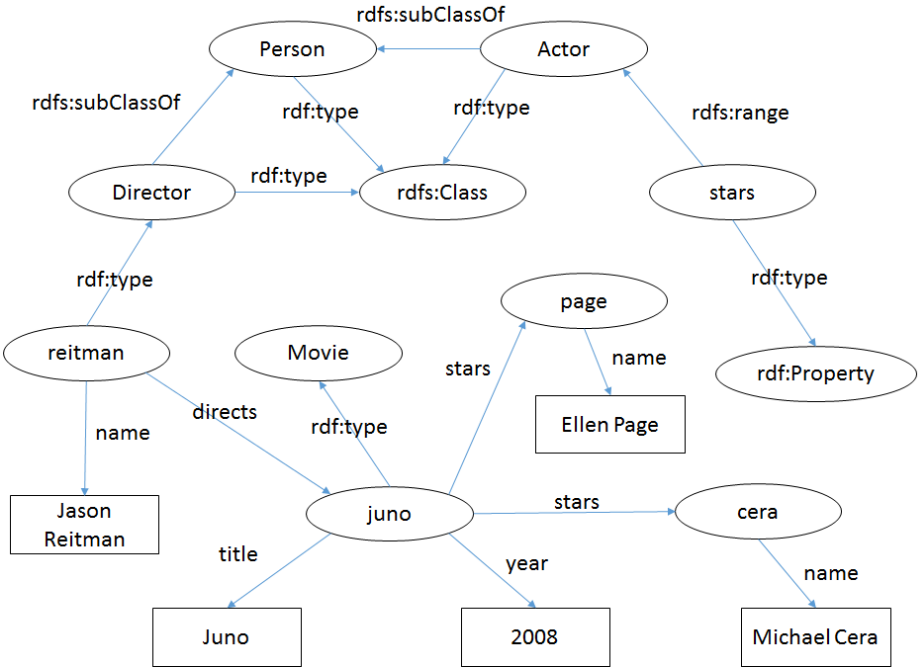
OPEN DATA EXAM

19th of June 2015. *The exam will take 2 hours. Answer each question in the provided space.* Answers out of such space will not be considered.

Name: ……

Question 1. [5p]

a) Write an RDF model in Turtle notation representing the following RDF Graph. When writing the triples separate them in two blocks: one block for terminological triples (i.e., defining concepts) and one block for asserted facts (i.e., defining instances). **(1,5p)**



Note: Squared concepts represent literals. For the other concepts without an explicit URI assume the http://www.example.org/ namespace. Properly define any other prefix than rdf and rdfs you may want to use.

Prefixes:

Terminological triples (*concepts*)

```
:Person rdf:type rdfs:Class .
:Actor rdf:type rdfs:Class .
:Actor rdfs:subClassOf :Person .
:stars rdf:type rdf:Property .
:stars rdfs:range :Actor .
:Director rdf:type rdfs:Class .
:Director rdfs:subClassOf :Person .
```

Asserted facts (*instances*)

```
:reitman rdf:type :Director .
:reitman :name "Jason Reitman" .
:reitman :directs :juno .
:juno rdf:type :Movie .
:juno :title "Juno" .
:juno :year 2008 .
:juno :stars :cera .
:juno :stars :page
:cera :name "Michael", "Cera" .
:page :name "Ellen Page" .
```

b) Assume a RDFS entailment regime for the RDF graph you just created. Represent the inferred knowledge in the form of triples **and** for each of them justify from what RDFS semantic construct it is inferred. **Circle** those inferred triples that were not already represented in the graph (i.e., clearly identify the new knowledge inferred) **(1,5p)**

```
:Movie rdf:type rdfs:Class .
:page rdf:type :Actor .
:cera rdf:type :Actor .
```

c) Write a SPARQL query to retrieve the following query: "*Retrieve the number of movies starred by Ellen Page per year*"

The query must compile and include a proper definition of the prefixes used. **(1p)**

```
SELECT ?year (COUNT(?movie) AS ?movieCount)
WHERE {
  ?movie :stars ?actor .
  ?actor :name "Ellen Page" .
  ?movie :year ?year .
}
GROUP BY ?year
ORDER BY ?year
```

d) **Just considering this query**, what would be the best option to physically store this RDF graph: either in a relational-based or graph-based triplestore? **Justify** your answer. **(0,5p)**

* Relational DBs are optimized for structured queries and can efficiently handle SPARQL queries, especially those involving counts, joins, groupings.
While relational databases can store RDF data, they might not be as flexible in handling the schema-less nature of RDF and might require additional overhead for schema management. Handling highly interconnected data can become complex and less efficient compared to graph databases.
* Graph DBs are designed to store and query graph data like RDF naturally. They handle the schema-less, highly interconnected nature of RDF data more efficiently. Queries that involve traversing relationships are typically faster in graph databases. While graph DBs excel in traversals, complex aggregations (like counting and grouping) might not be as optimized as in relational DBs.
* The given query involves counting and grouping, which relational databases handle well. However, modern graph DBs are also

e) What are the differences between a triplestore and a graph database? **(0,5p)**

* Triplestores store data in the form of triples. RDF data is schema-less. : The primary query language is SPARQL, supporting querying based on semantics and relationships between triples, including complex pattern matching.
* Graph DBs store data as nodes, edges, and properties. The query languages include Cypher (used by Neo4j), Gremlin, optimized for graph traversals, allowing for efficient queries on paths, neighbors, and subgraphs.

Question 2. [5p]

a) Model in RDF(S) the following statements:

"*A football player plays in a team but could have played in many teams before. Thus, we want to store the period he / she played for a team. For each team, though, he / she had a unique number assigned. Each team currently has a single manager, but obviously it might have had different managers along time. Again, we want to store the period each manager was at each team. Finally, we want to consider the figure of player-manager, who is someone that for a team played the role of manager and player at the same time*".

Draw a RDF graph modelling as many statements as you can. Clearly identify the statements modelled in your graph. Clearly identify in the graph the RDFS constructs used and the concepts created. Create your own namespace for the URIs you need to create. **(1,5p)**

b) Use Description Logics to represent those statements that you could not represent using RDF(S). Clearly identify the statement and the DL construct used. Maintain the consistency with your RDF(S) graph by using the same names (i.e., URIs) whenever possible. **(1,5p)**

c) Is there any statement that you could not represent yet? Why? **(0,5p)**

- We need to ensure that the periods during which a player plays for a team do not overlap with the same player managed the same team unless the player is a player-manager.
- Ensuring that a player-manager role implies that the person simultaneously holds both roles for a team.

d) Now, assert the following instance to your RDF graph (draw it in the above RDF graph sketched and clearly identify it by drawing the instances within rectangles): *Messi is a player and plays for FC Barcelona from 2004 until now with the number 10*. **(0,5p)**

e) Assume now the following query over the RDF graph: "*Retrieve all managers*". Do not consider SPARQL though and strictly **apply the open-world assumption** (answer the query from a theoretical point of view). What would be the result retrieved? Why? **(1p)**

Under the open-world assumption, if we query the RDF graph to retrieve all managers, we can only assert that certain individuals are managers if they are explicitly stated to be so in the graph. We cannot infer that an individual is not a manager simply because there is no information about them being a manager.