

Basic Multi-Level Modeling in RDF and SHACL

— Tutorial and Task Assignment —

Sebastian Gruber, Bernd Neumayr

Contents

- Introduction
- DDO-Playground
- Use Case
- Multi-level Modeling
- Task
- Evaluation
- Contact
- References

Introduction

Use Case: Just-in-Time Adaptive Intervention (JITAI)

- The just-in-time adaptive intervention (JITAI) is an intervention design to support health behavior changes for, e.g., physical inactivity, ...

(Nahum-Shani et al., 2018)

- This intervention design aims to
 - “provide the right type/amount of support”,
 - “at the right time”,
 - “by adapting to an individual’s changing internal and contextual state”.

(Nahum-Shani et al., 2018)

“At 2021-07-15 12:10, Jane is available for intervention. She has been sedentary for the last 30 minutes and the weather is fine. The system suggests to take a walk to the nearby lake to have lunch at the lakeside restaurant. Jane follows the suggestion and makes 3000 steps in the next hour.” (Gruber et al., 2021)

Introduction

Basic Multi-Level Modeling in RDF/SHACL

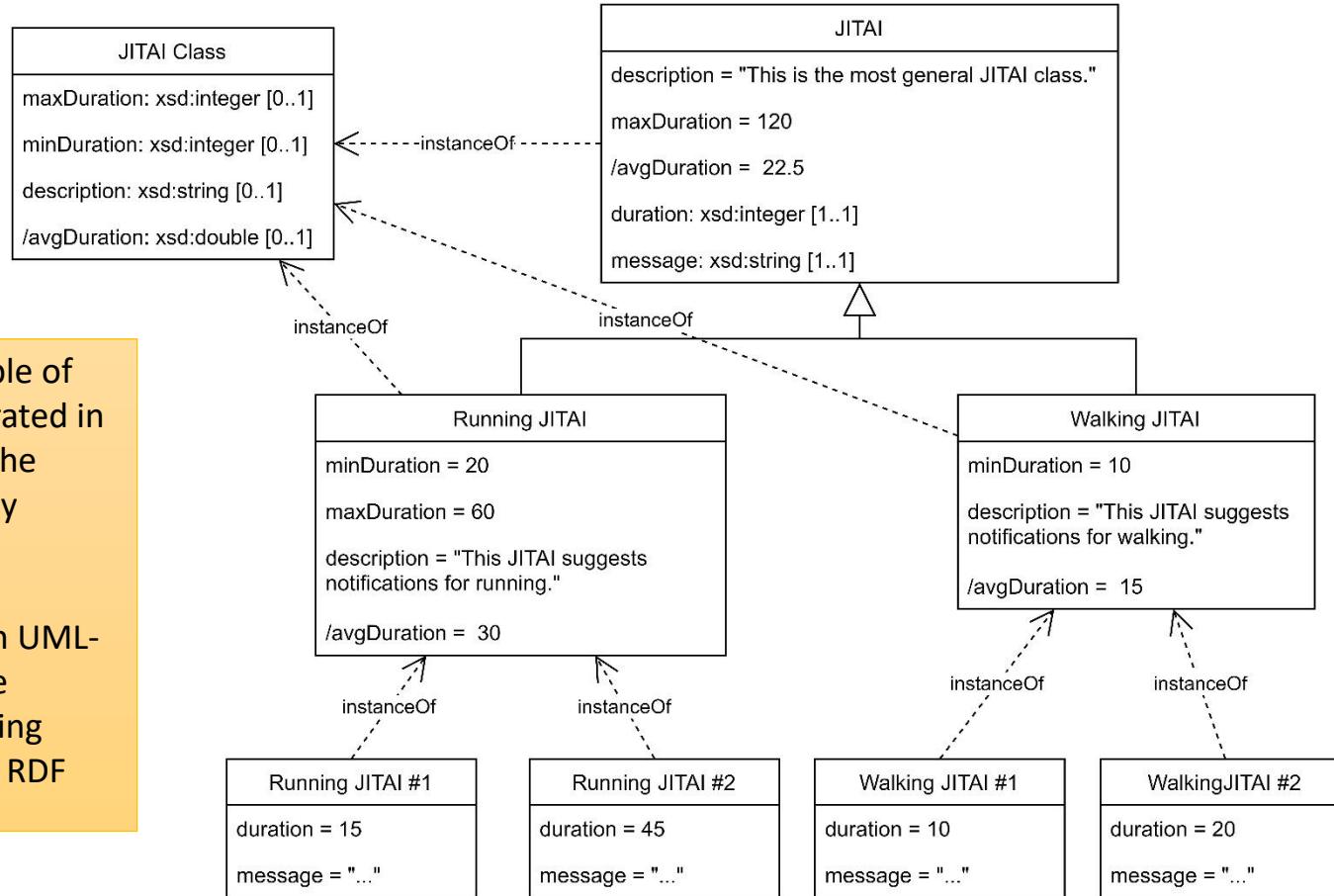
- The purpose of this tutorial is to provide students a first contact with basics of multi-level modeling (MLM) in RDF and SHACL.
- We assume that students have a solid knowledge of RDF, SPARQL and SHACL.
- We only consider 3 classification levels (individuals, classes, meta classes) and focus on the use of powertypes for metamodeling in SHACL.

Introduction

Basic Multi-Level Modeling in RDF/SHACL

This is a part of our running example of MLM of JITAIs which will be elaborated in the rest of the tutorial and forms the basis of the tasks to be prepared by students.

For illustration purposes we use an UML-like representation here, but in the remainder of the tutorial the running example will be represented using RDF and SHACL.



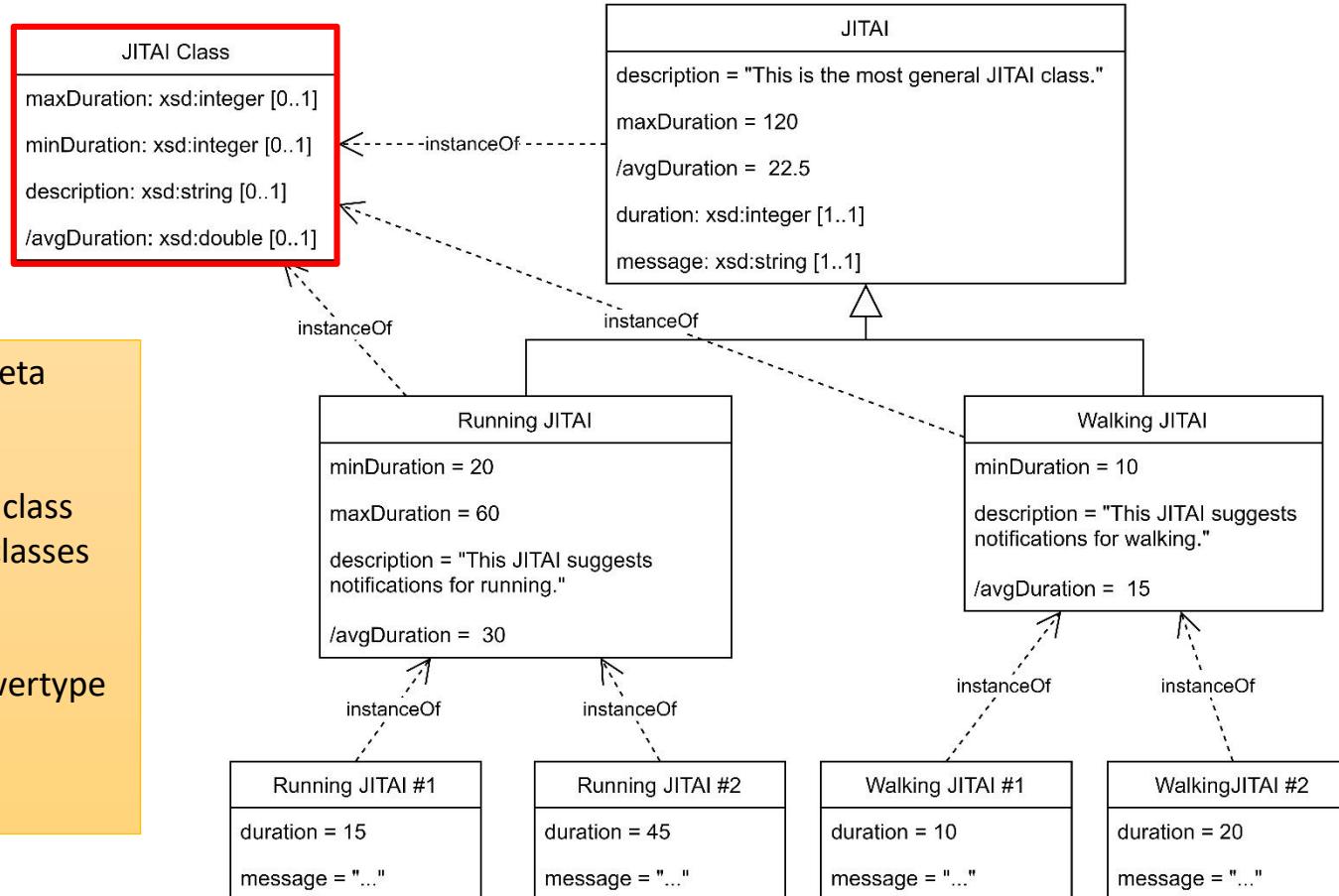
Introduction

Basic Multi-Level Modeling in RDF/SHACL

In MLM classes are instances of meta classes.

The **powertype** of a class has that class and all its direct and indirect sub classes as instances. (Carvalho et al., 2018)

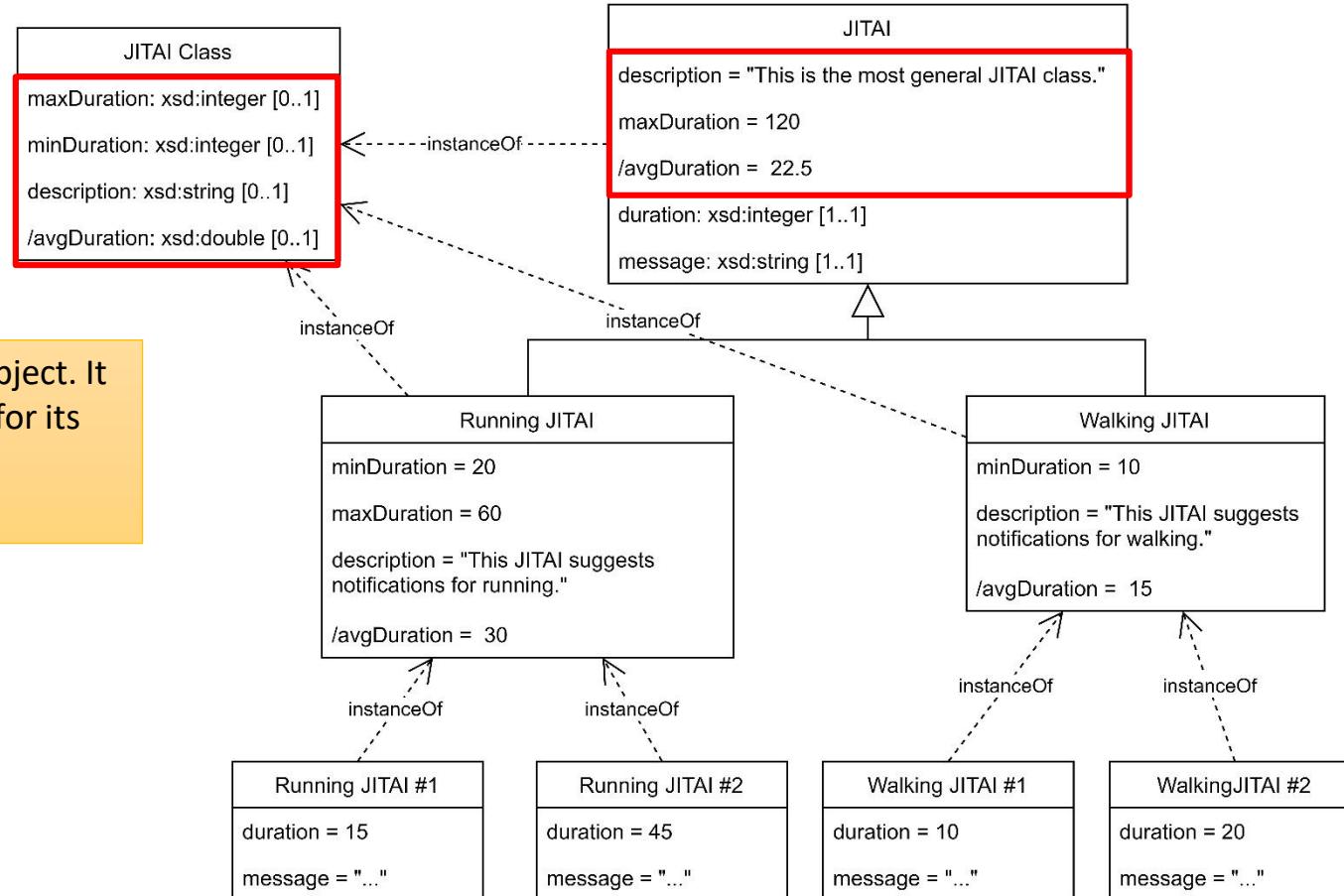
For example, **JITAI Class** is the powertype of class JITAI and has JITAI and its subclasses as instances.



Introduction

Basic Multi-Level Modeling in RDF/SHACL

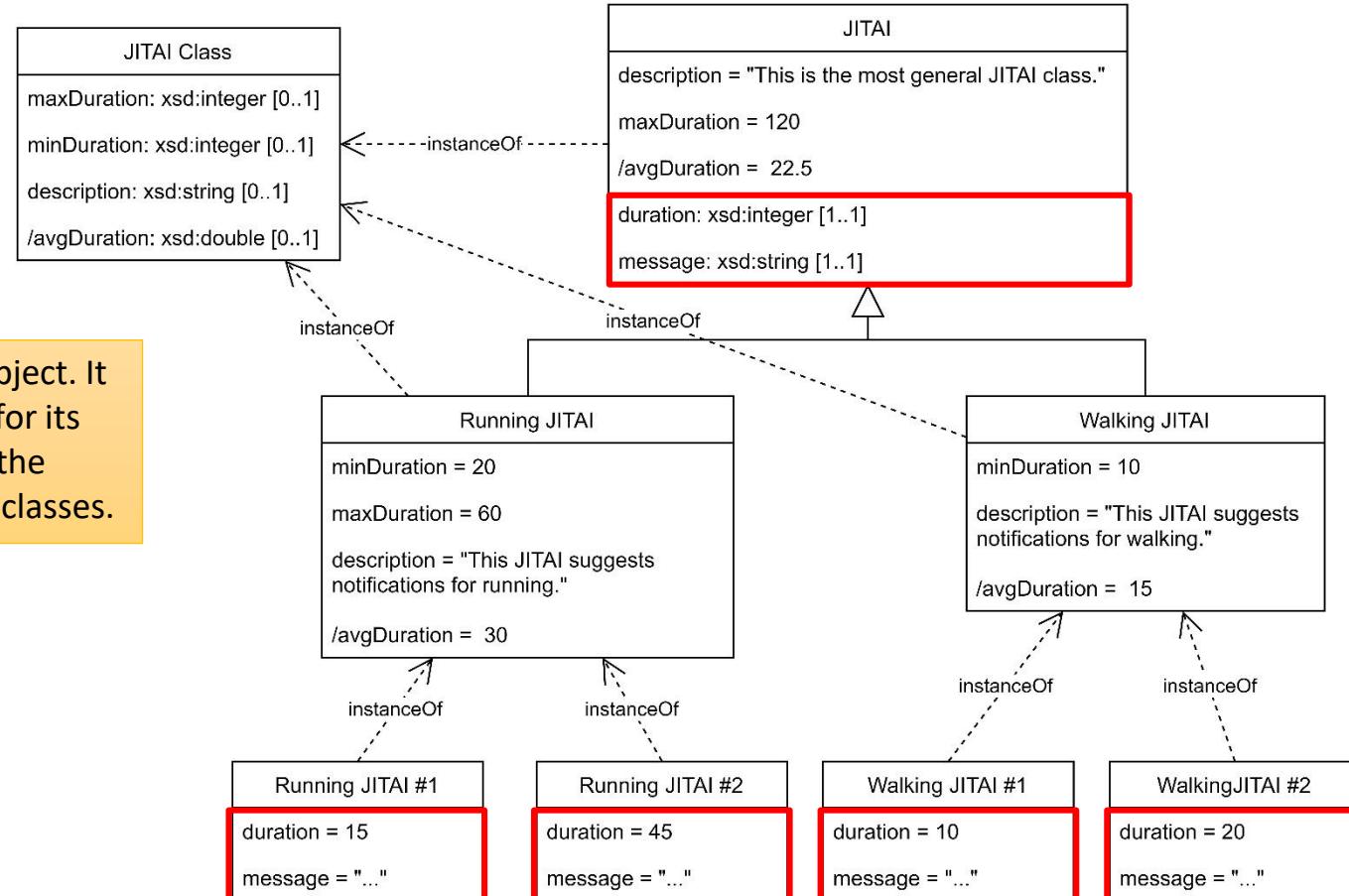
A **clabout** is both a class and an object. It defines the schema of properties for its instances ...



Introduction

Basic Multi-Level Modeling in RDF/SHACL

A **clobject** is both a class and an object. It defines the schema of properties for its instances and specifies values for the properties introduced by its meta classes.

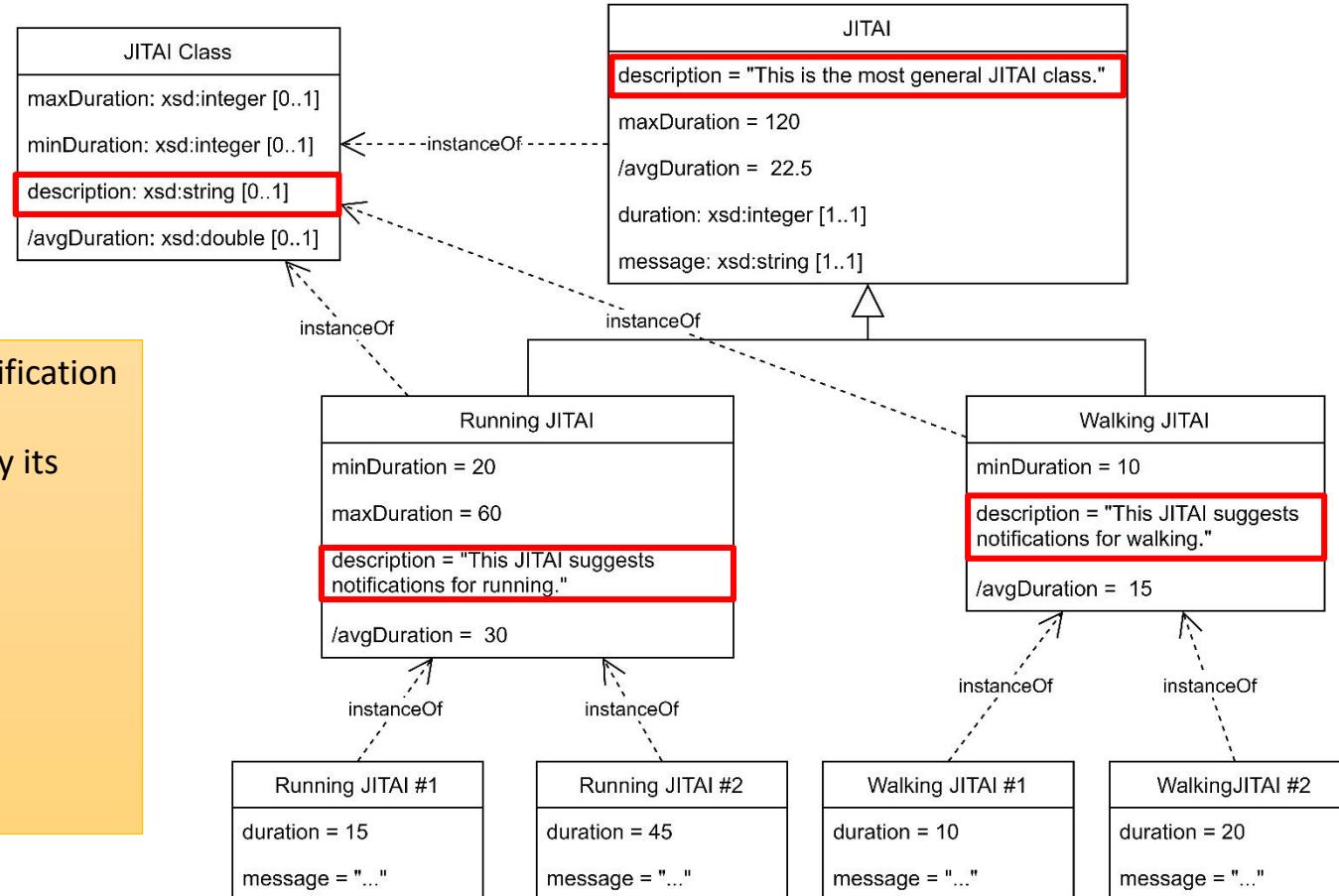


Introduction

Basic Multi-Level Modeling in RDF/SHACL

Meta classes can be used for specification of

- **direct properties** instantiated by its member classes,

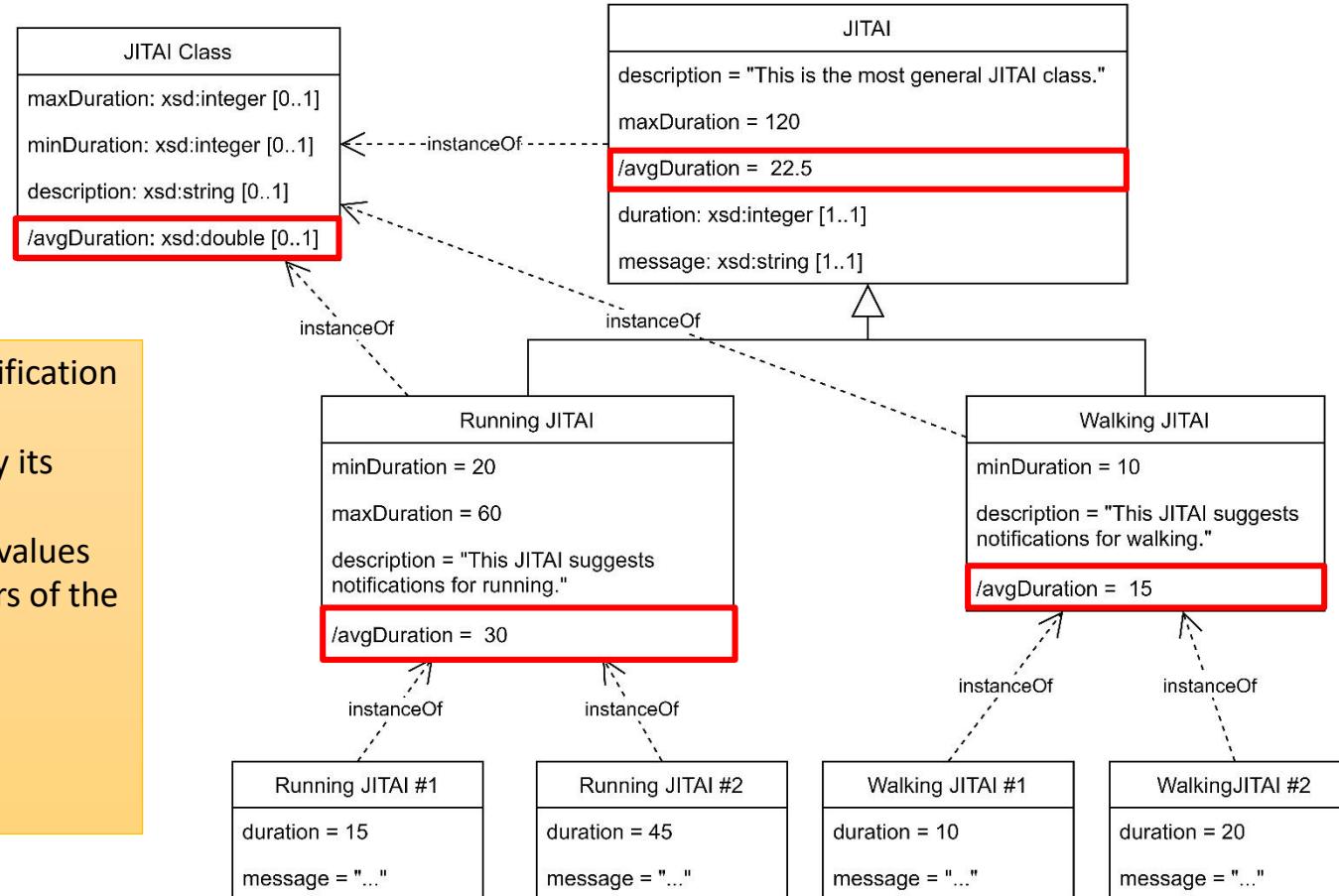


Introduction

Basic Multi-Level Modeling in RDF/SHACL

Meta classes can be used for specification of

- direct properties instantiated by its member classes,
- **resultant properties** with their values being derived from the members of the member classes



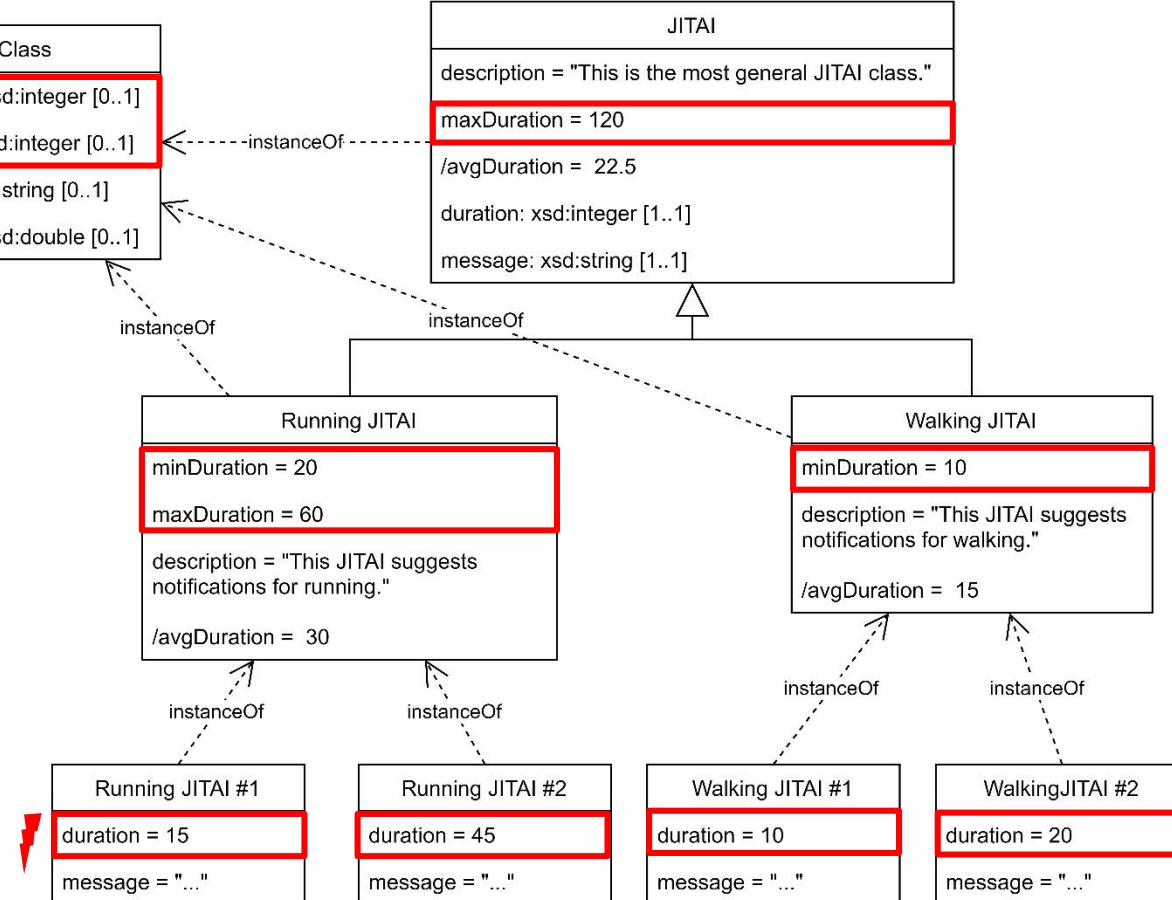
Introduction

Basic Multi-Level Modeling in RDF/SHACL

Meta classes can be used for specification of

- direct properties instantiated by its member classes,
- resultant properties with their values being derived from the members of the member classes,
- and **regularity properties** regulating property values of the members of the member classes.

(Almeida et al., 2021)



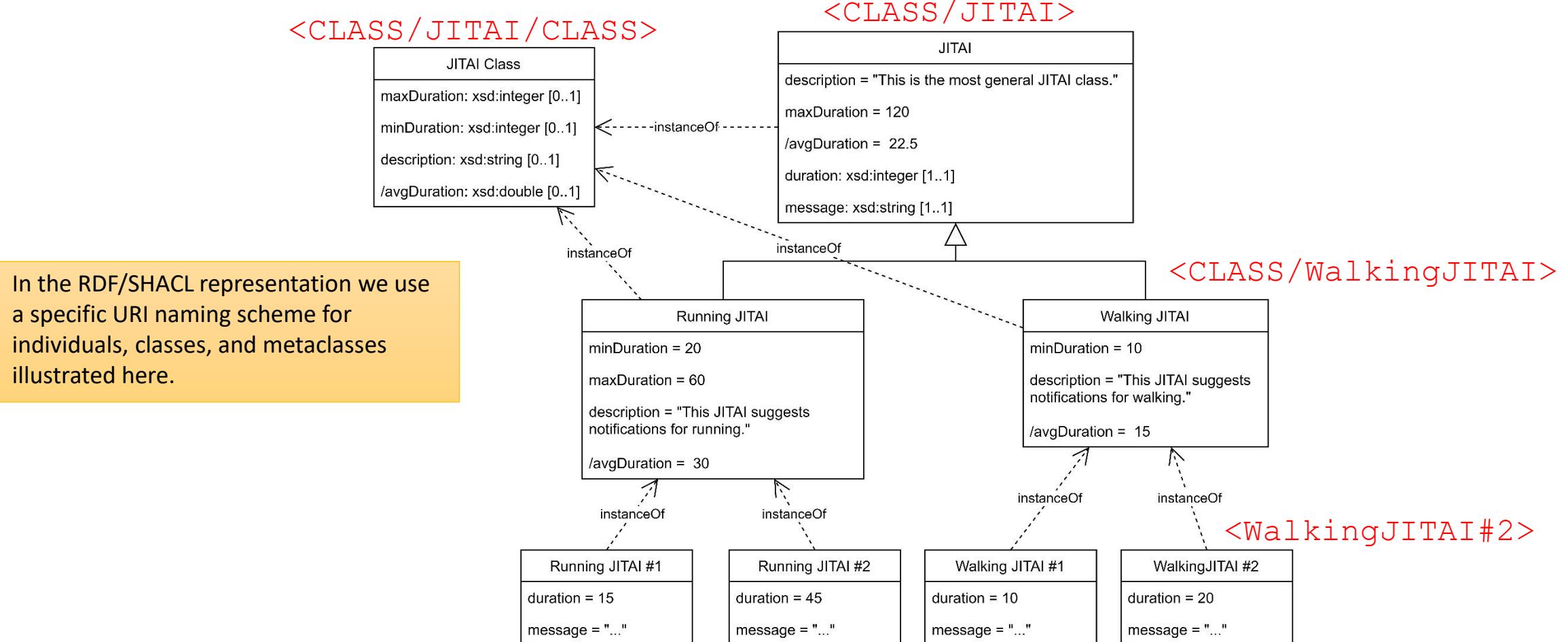
Introduction

Basic Multi-Level Modeling in RDF/SHACL

- The Resource Description Framework (RDF) is a W3C standard for graph-based description of resources.
 - <https://www.w3.org/TR/rdf11-primer/>
- The Shapes Constraint Language (SHACL) is a W3C standard for validation of RDF graphs and beyond.
 - <https://www.w3.org/TR/shacl/>
 - <https://www.w3.org/TR/shacl-af/>
- SHACL builds on SPARQL, the query language for RDF, which is also a W3C standard
 - <https://www.w3.org/TR/sparql11-query/>

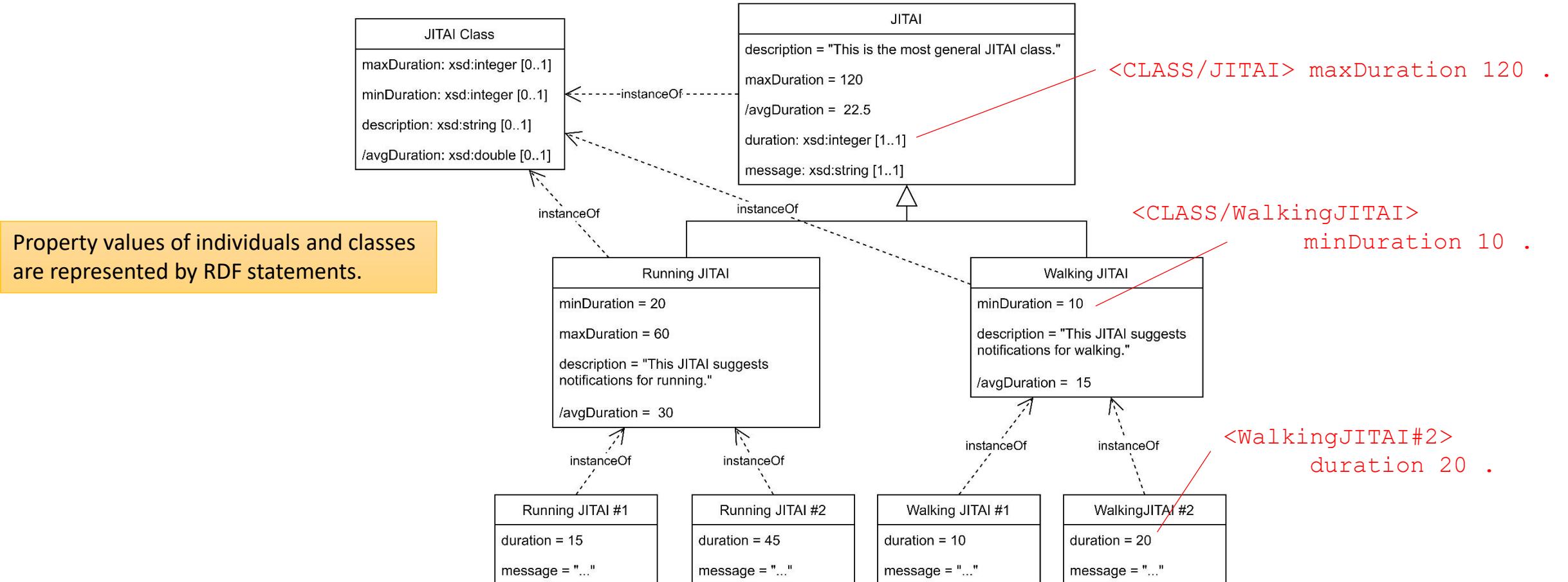
Introduction

Basic Multi-Level Modeling in RDF/SHACL



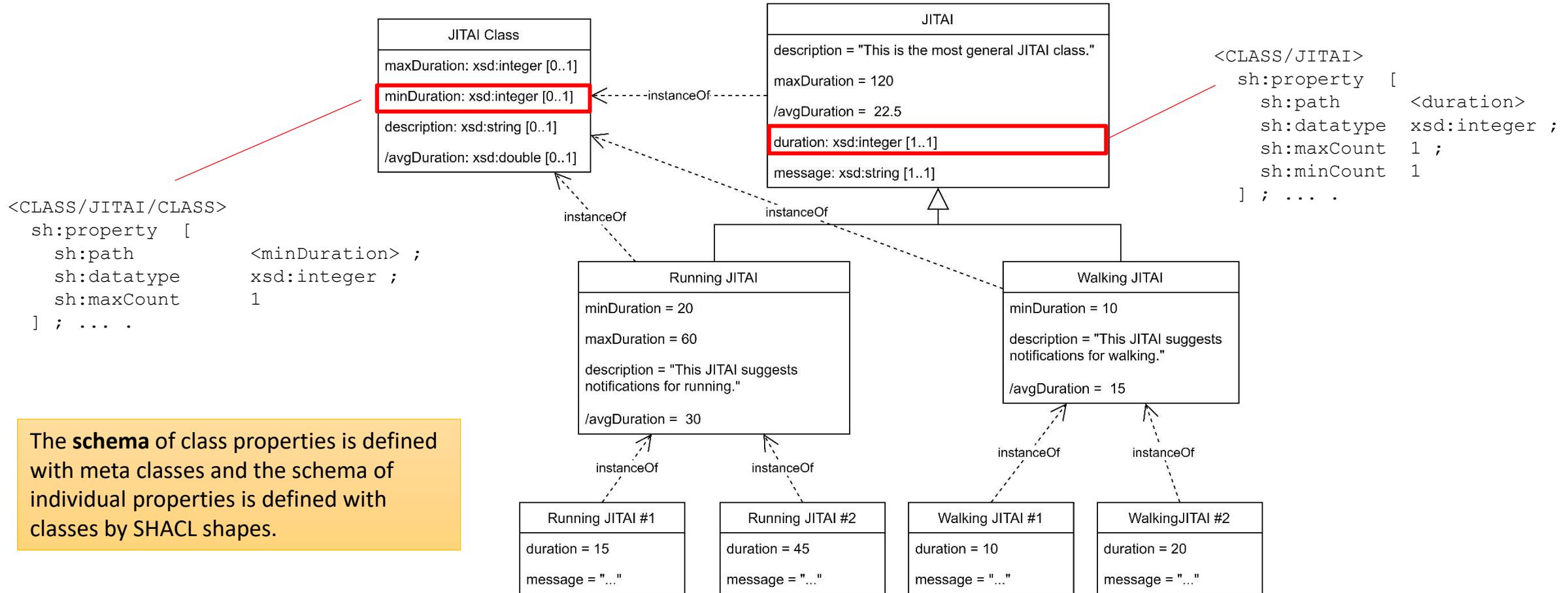
Introduction

Basic Multi-Level Modeling in RDF/SHACL



Introduction

Basic Multi-Level Modeling in RDF/SHACL

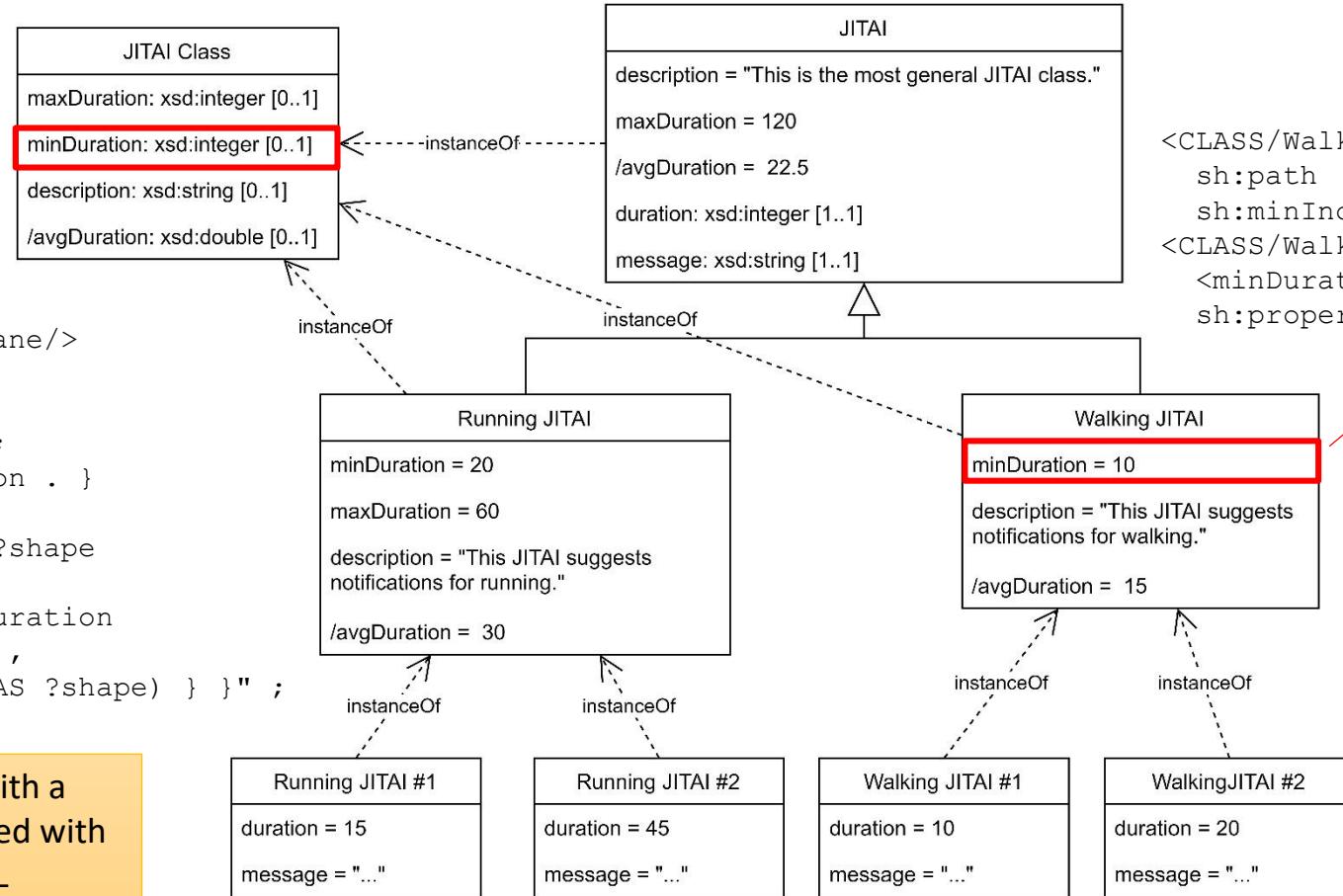


Introduction

Basic Multi-Level Modeling in RDF/SHACL

```
<CLASS/JITAI/CLASS> sh:rule [
  rdf:type sh:SPARQLRule ;
  sh:construct "
    BASE <https://example.org/Jane/>
    CONSTRUCT {
      $this sh:property ?shape .
      ?shape sh:path <duration> ;
      sh:minInclusive ?minDuration .
    }
    WHERE {
      SELECT $this ?minDuration ?shape
      WHERE {
        $this <minDuration> ?minDuration
        BIND(IRI(CONCAT(STR($this),
          \"#\", \"duration\")) AS ?shape) } }" ;
]
```

Each **regularity property** comes with a SPARQL-based SHACL rule (specified with the meta class) that derives SHACL constraints for classes from regularity property values.

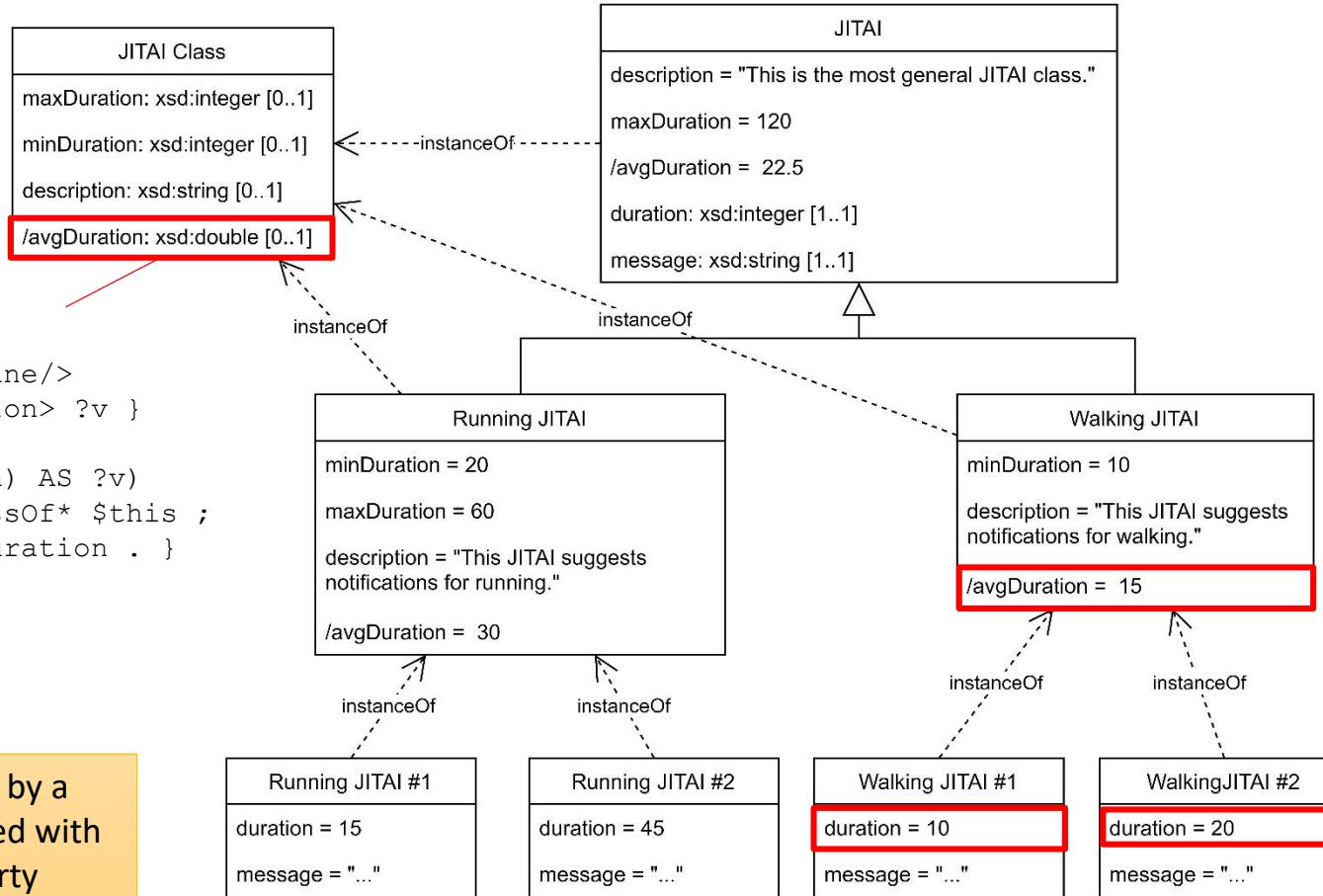


```
<CLASS/WalkingJITAI#duration>
  sh:path              <duration> ;
  sh:minInclusive 10 .
<CLASS/WalkingJITAI>
  <minDuration> 10 ;
  sh:property <CLASS/WalkingJITAI#duration> .
```

Introduction

Basic Multi-Level Modeling in RDF/SHACL

```
<CLASS/JITAI/CLASS>
sh:rule [
  rdf:type sh:SPARQLRule ;
  sh:construct """
    BASE <https://example.org/Jane/>
    CONSTRUCT { $this <avgDuration> ?v }
    WHERE {
      SELECT $this (AVG(?duration) AS ?v)
      WHERE { ?obj a/rdfs:subClassOf* $this ;
               <duration> ?duration . }
      GROUP BY $this
    """
]
```



Each **resultant property** is defined by a SPARQL-based SHACL rule (specified with the meta class) that derives property values for classes from individuals.

Prototype

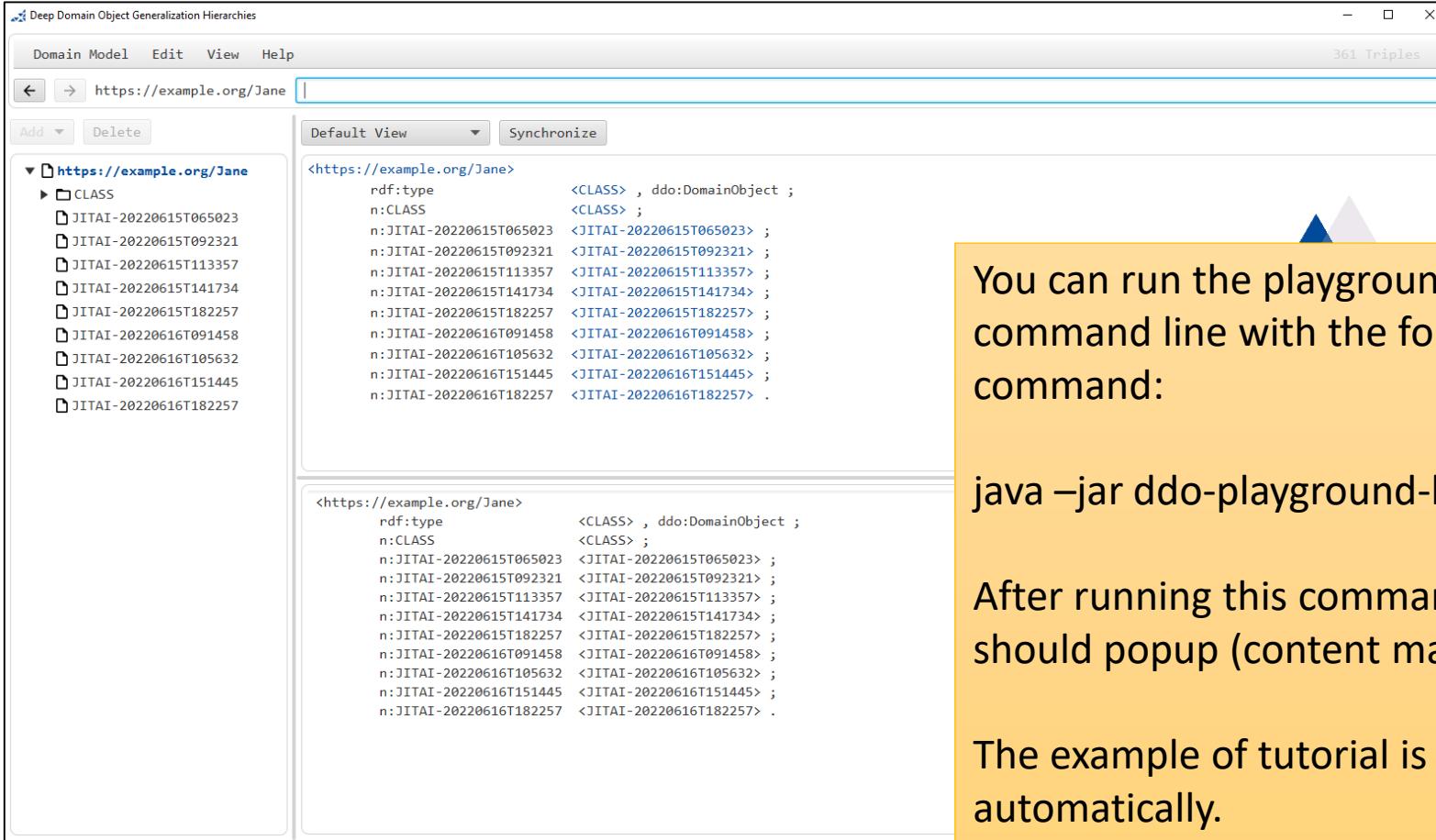
DDO Playground

- The prototype (DDO-Playground) used in the tutorial implements an extension of the Deep Domain Object Hierarchies (DDO) approach
(Neumayr & Schrefl, 2022).
- For the purposes of this tutorial, we use the DDO-Playground, but severely limit its functionality and do not consider DDO-specific language constructs and composition hierarchies.
- You can download the limited version of the DDO-Playground here:
 - <https://resources.lbi-dhp.at/mlm4jtais/ddo-playground-basic.jar>

Note: You may have to install Oracle JDK 18.
(You may run into troubles using Open JDK)

DDO-Playground

Getting Started



You can run the playground from command line with the following command:

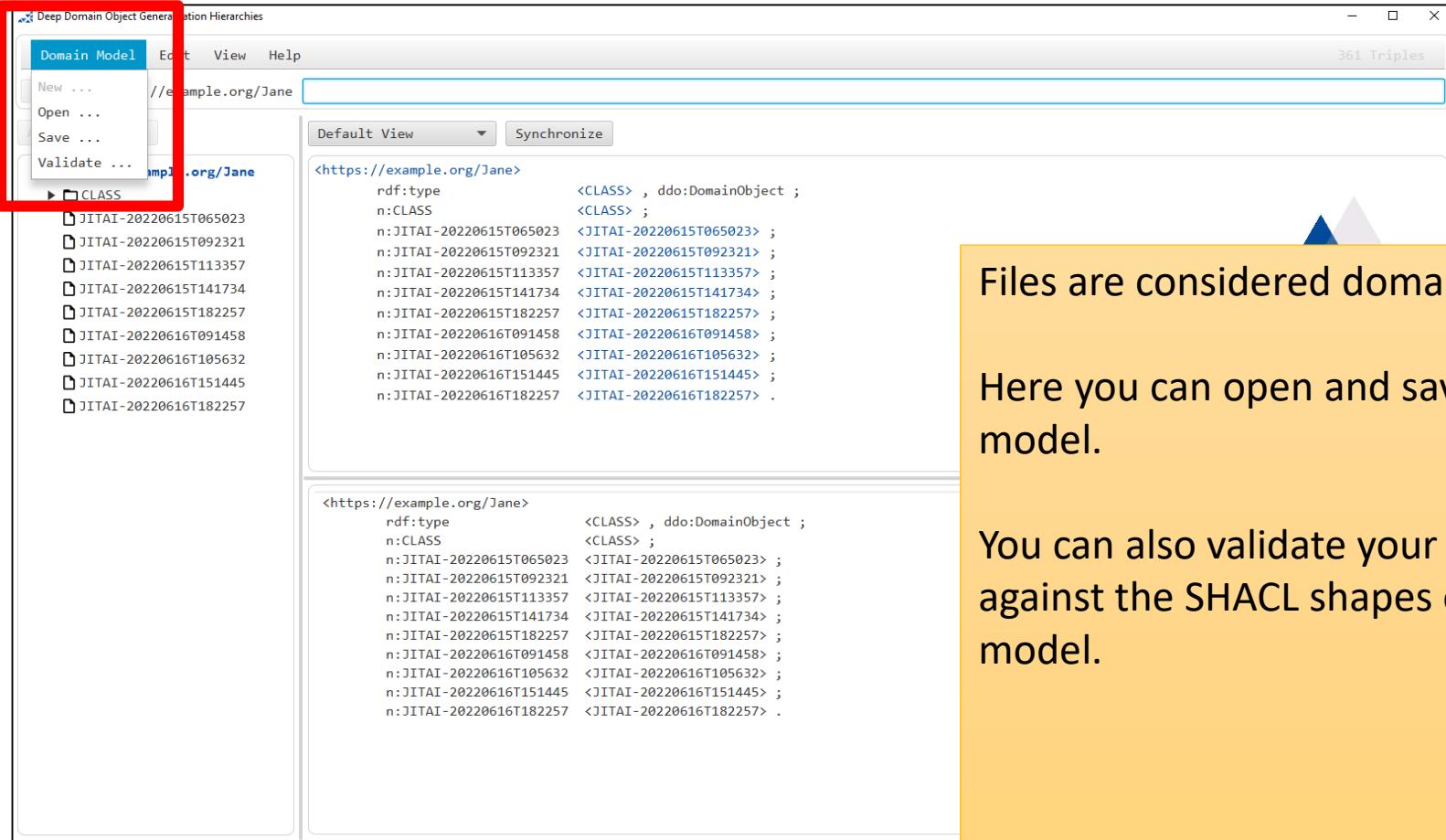
```
java -jar ddo-playground-basic.jar
```

After running this command, this window should popup (content may differ).

The example of tutorial is loaded automatically.

DDO-Playground

Domain Model



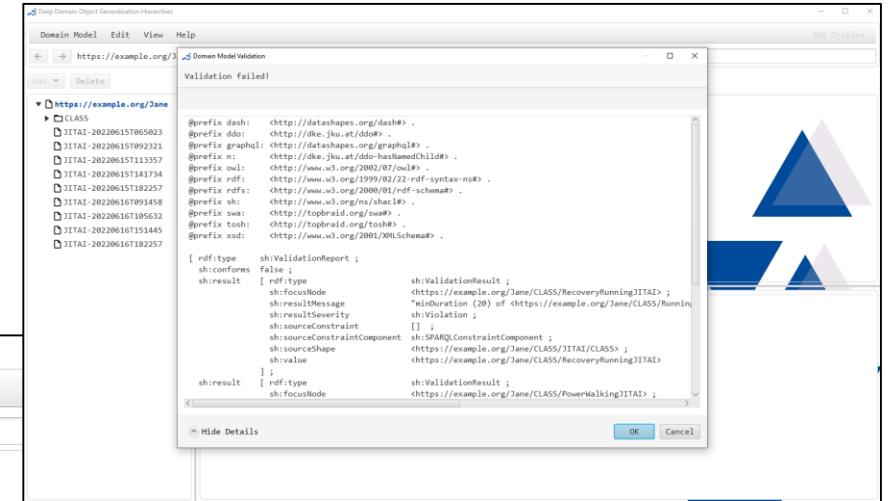
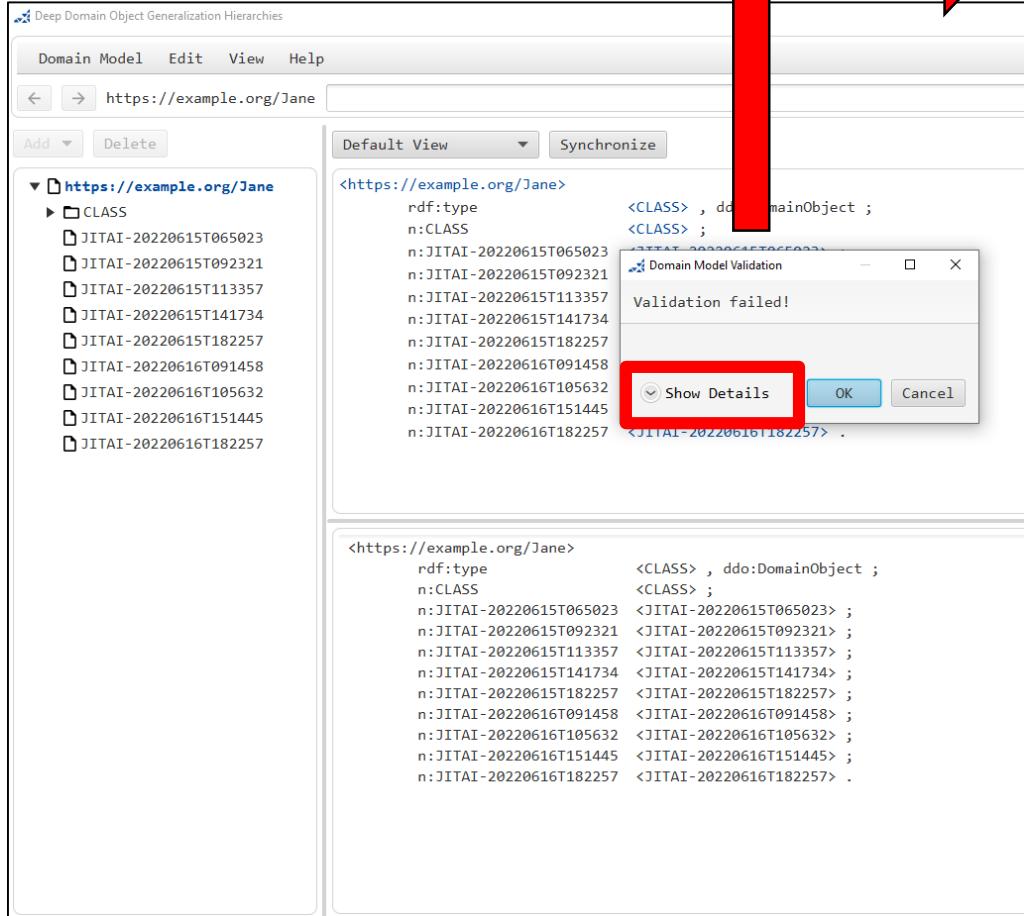
The screenshot shows the DDO-Playground application window titled "Deep Domain Object Generation Hierarchies". The menu bar includes "Domain Model", "Edit", "View", and "Help". A status bar at the bottom right indicates "361 Triples". The left sidebar has a red box highlighting the "Domain Model" button. Below it are "New ...", "Open ...", "Save ...", and "Validate ...". The main area displays two RDF triples for the URL <https://example.org/Jane>. The first triple is:

```
<https://example.org/Jane> rdf:type <CLASS> , ddo:DomainObject ;  
n:CLASS <CLASS> ;  
n:JITAI-20220615T065023 <JITAI-20220615T065023> ;
```

The second triple is identical to the first, showing the same structure for the same URL.

Files are considered domain models.
Here you can open and save a domain model.
You can also validate your domain model against the SHACL shapes of the current model.

DDO-Playground SHACL Validation



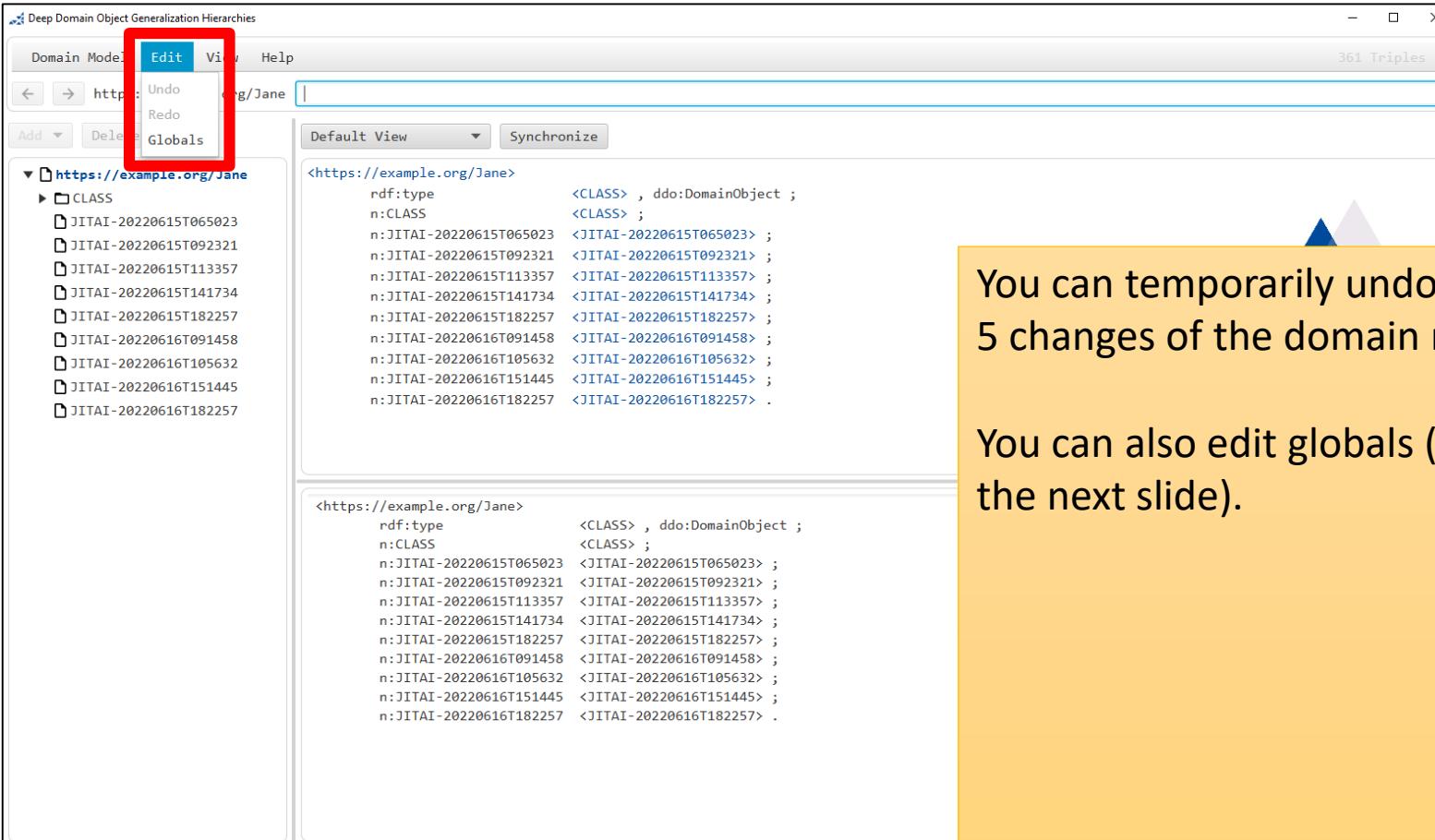
When validating your domain model, you may want to look into the details of the SHACL validation report.

Therefore, click on *Show Details*.

The main window and popups can be resized accordingly if necessary.

DDO-Playground

Edit

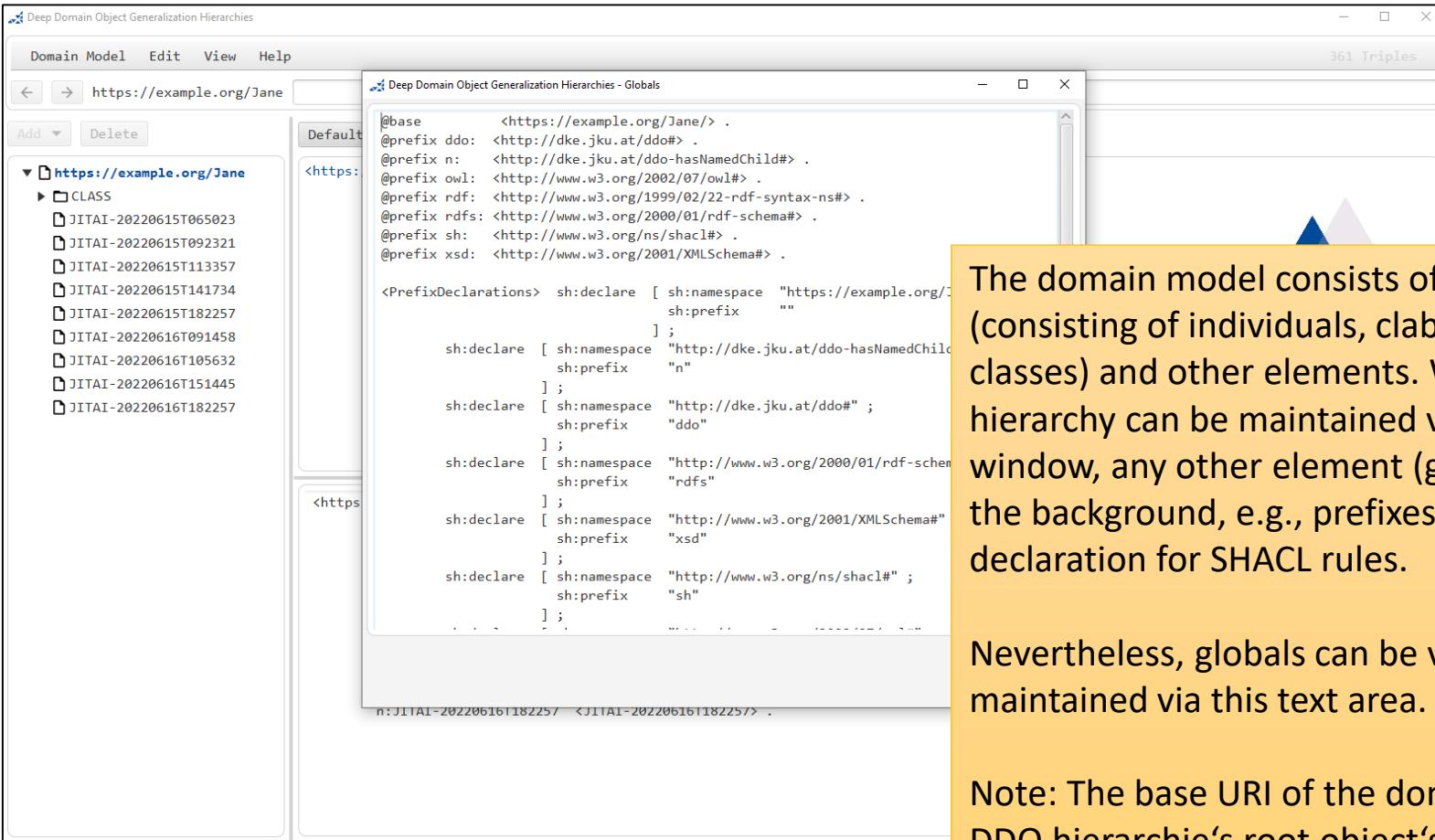


You can temporarily undo and redo up to 5 changes of the domain model.

You can also edit globals (more details on the next slide).

DDO-Playground

Globals



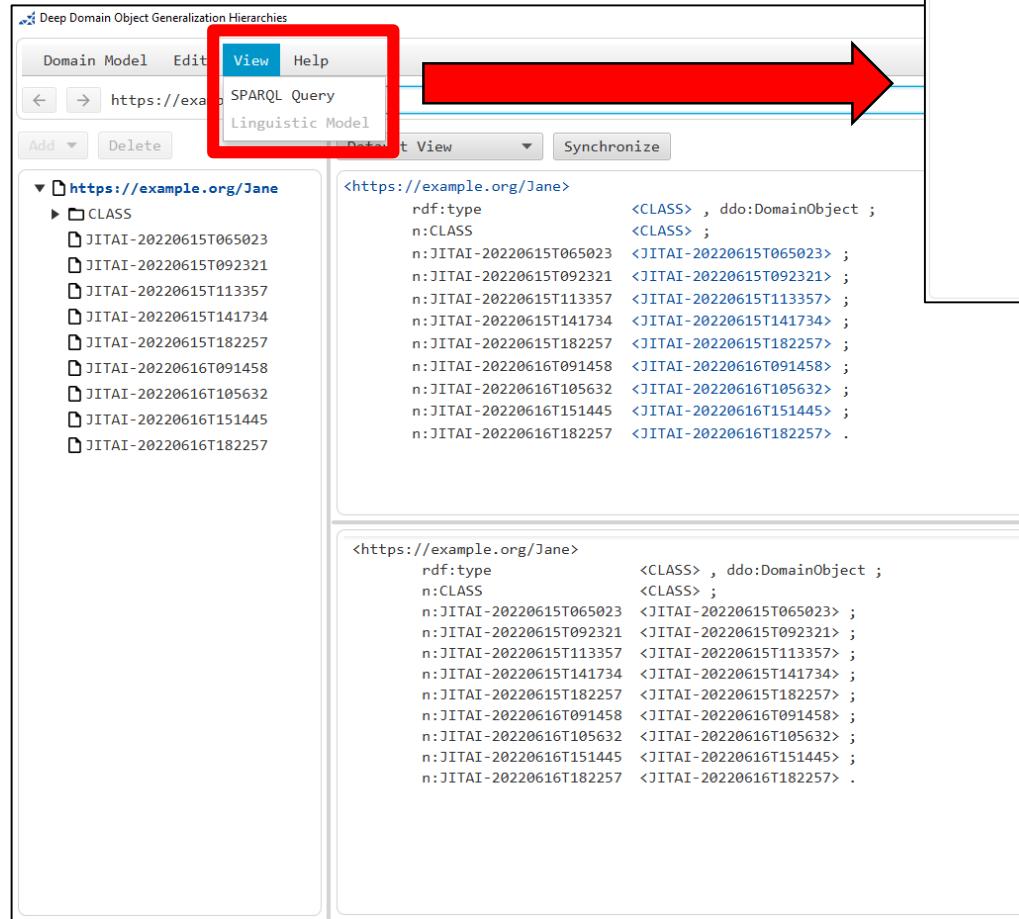
The domain model consists of a DDO hierarchy (consisting of individuals, clabjects and meta classes) and other elements. While the DDO hierarchy can be maintained via the main window, any other element (globals) remain in the background, e.g., prefixes or prefix declaration for SHACL rules.

Nevertheless, globals can be viewed or maintained via this text area.

Note: The base URI of the domain model is the DDO hierarchie's root object's URI + /.

DDO-Playground

SPARQL Queries

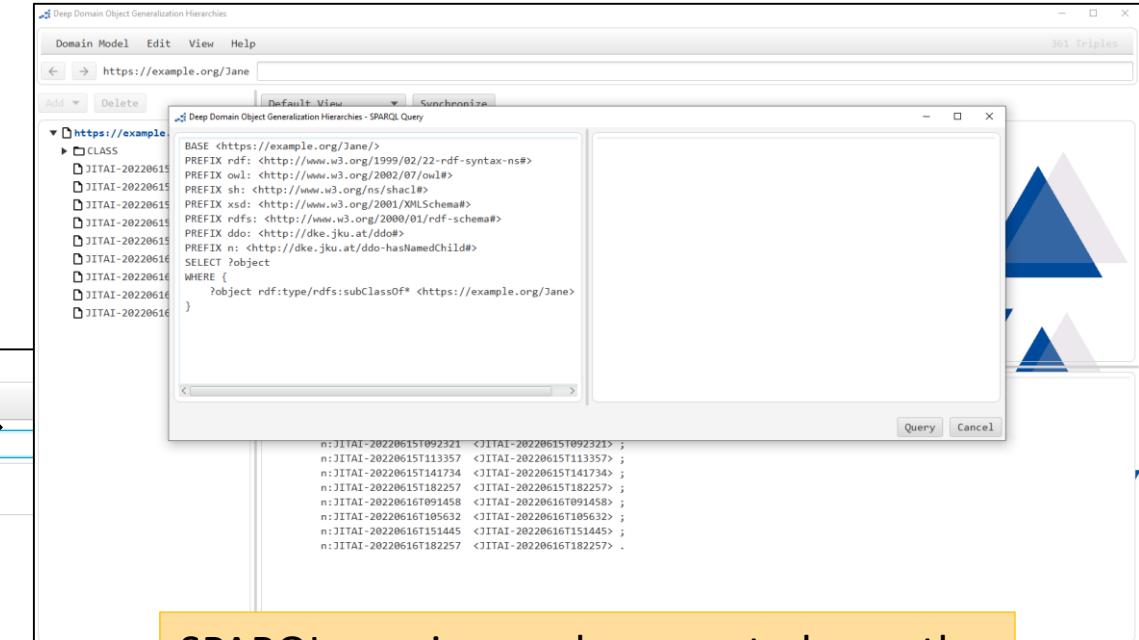


The screenshot shows the DDO-Playground application window. In the top menu bar, the 'View' tab is highlighted with a red box and a large red arrow pointing from the left towards the right side of the interface. Below the menu, there are buttons for 'Add', 'Delete', 'Default View', and 'Synchronize'. The main area displays two panels. The left panel shows a tree view of the domain model under the URL <https://example.org/Jane>, with several class nodes listed. The right panel shows the results of a SPARQL query for the same URL, displaying triples such as:

```
<https://example.org/Jane> rdf:type <CLASS> , ddo:DomainObject ;  
n:CLASS <CLASS> ;  
n:JITAI-20220615T065023 <JITAI-20220615T065023> ;  
n:JITAI-20220615T092321 <JITAI-20220615T092321> ;  
n:JITAI-20220615T113357 <JITAI-20220615T113357> ;  
n:JITAI-20220615T141734 <JITAI-20220615T141734> ;  
n:JITAI-20220615T182257 <JITAI-20220615T182257> ;  
n:JITAI-20220616T091458 <JITAI-20220616T091458> ;  
n:JITAI-20220616T105632 <JITAI-20220616T105632> ;  
n:JITAI-20220616T151445 <JITAI-20220616T151445> ;  
n:JITAI-20220616T182257 <JITAI-20220616T182257> .
```

Below this, another set of triples is shown for the same URL:

```
<https://example.org/Jane> rdf:type <CLASS> , ddo:DomainObject ;  
n:CLASS <CLASS> ;  
n:JITAI-20220615T065023 <JITAI-20220615T065023> ;  
n:JITAI-20220615T092321 <JITAI-20220615T092321> ;  
n:JITAI-20220615T113357 <JITAI-20220615T113357> ;  
n:JITAI-20220615T141734 <JITAI-20220615T141734> ;  
n:JITAI-20220615T182257 <JITAI-20220615T182257> ;  
n:JITAI-20220616T091458 <JITAI-20220616T091458> ;  
n:JITAI-20220616T105632 <JITAI-20220616T105632> ;  
n:JITAI-20220616T151445 <JITAI-20220616T151445> ;  
n:JITAI-20220616T182257 <JITAI-20220616T182257> .
```



The screenshot shows a separate window titled 'Deep Domain Object Generalization Hierarchies - SPARQL Query'. The URL <https://example.org/Jane> is entered in the address bar. The main area contains a SPARQL query:

```
BASE <https://example.org/Jane/>  
PREFIX rdf: <http://www.w3.org/1999/02/rdf-syntax-ns#>  
PREFIX owl: <http://www.w3.org/2002/07/owl#>  
PREFIX sh: <http://www.w3.org/ns/shacl#>  
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
PREFIX ddo: <http://dke.jku.at/ddo#>  
PREFIX n: <http://dke.jku.at/ddo-hasNamedChild#>  
SELECT ?object  
WHERE {  
?object rdf:type/rdfs:subClassOf* <https://example.org/Jane>  
}
```

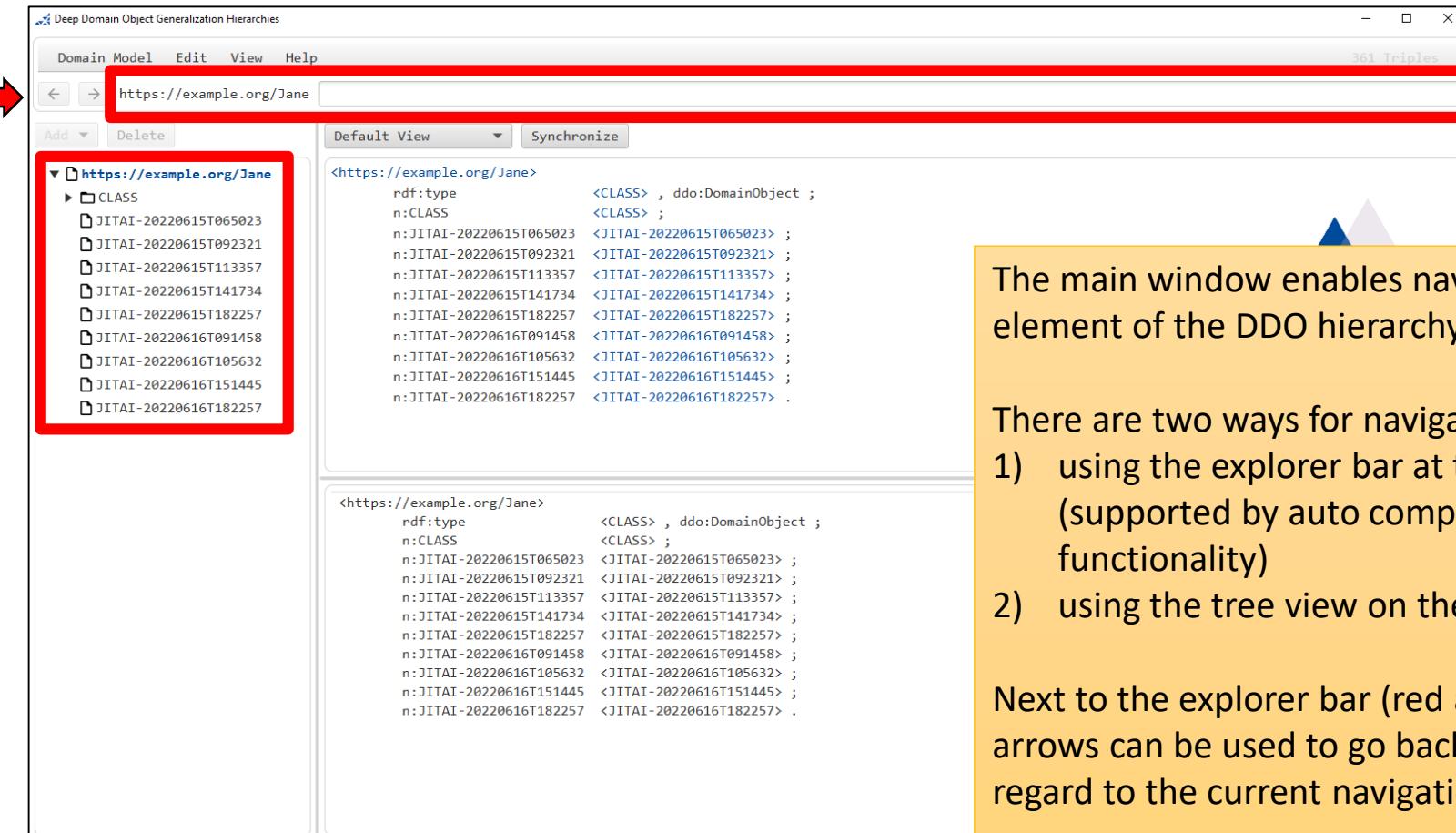
Below the query, the results are displayed as a list of triples:

```
n:JITAI-20220615T092321 <JITAI-20220615T092321> ;  
n:JITAI-20220615T113357 <JITAI-20220615T113357> ;  
n:JITAI-20220615T141734 <JITAI-20220615T141734> ;  
n:JITAI-20220615T182257 <JITAI-20220615T182257> ;  
n:JITAI-20220616T091458 <JITAI-20220616T091458> ;  
n:JITAI-20220616T105632 <JITAI-20220616T105632> ;  
n:JITAI-20220616T151445 <JITAI-20220616T151445> ;  
n:JITAI-20220616T182257 <JITAI-20220616T182257> .
```

SPARQL queries can be executed over the whole domain model, therefore, a SPARQL query editor pops up.

By default, a query selecting all the direct and indirect instances of the currently selected element is created.

DDO-Playground Navigation



The main window enables navigation to any element of the DDO hierarchy.

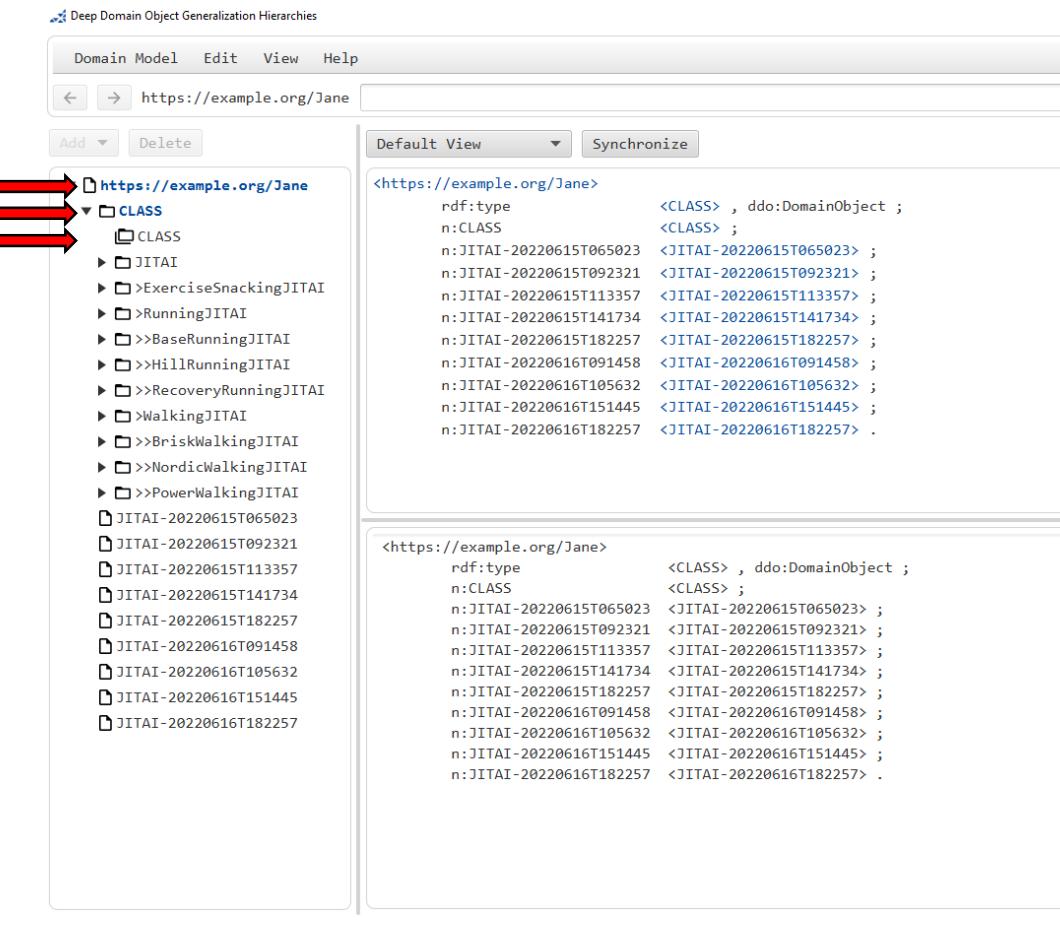
There are two ways for navigation:

- 1) using the explorer bar at the top
(supported by auto complete functionality)
- 2) using the tree view on the left hand side

Next to the explorer bar (red arrow), the arrows can be used to go back and forth with regard to the current navigation history.

DDO-Playground

Tree View Symbols



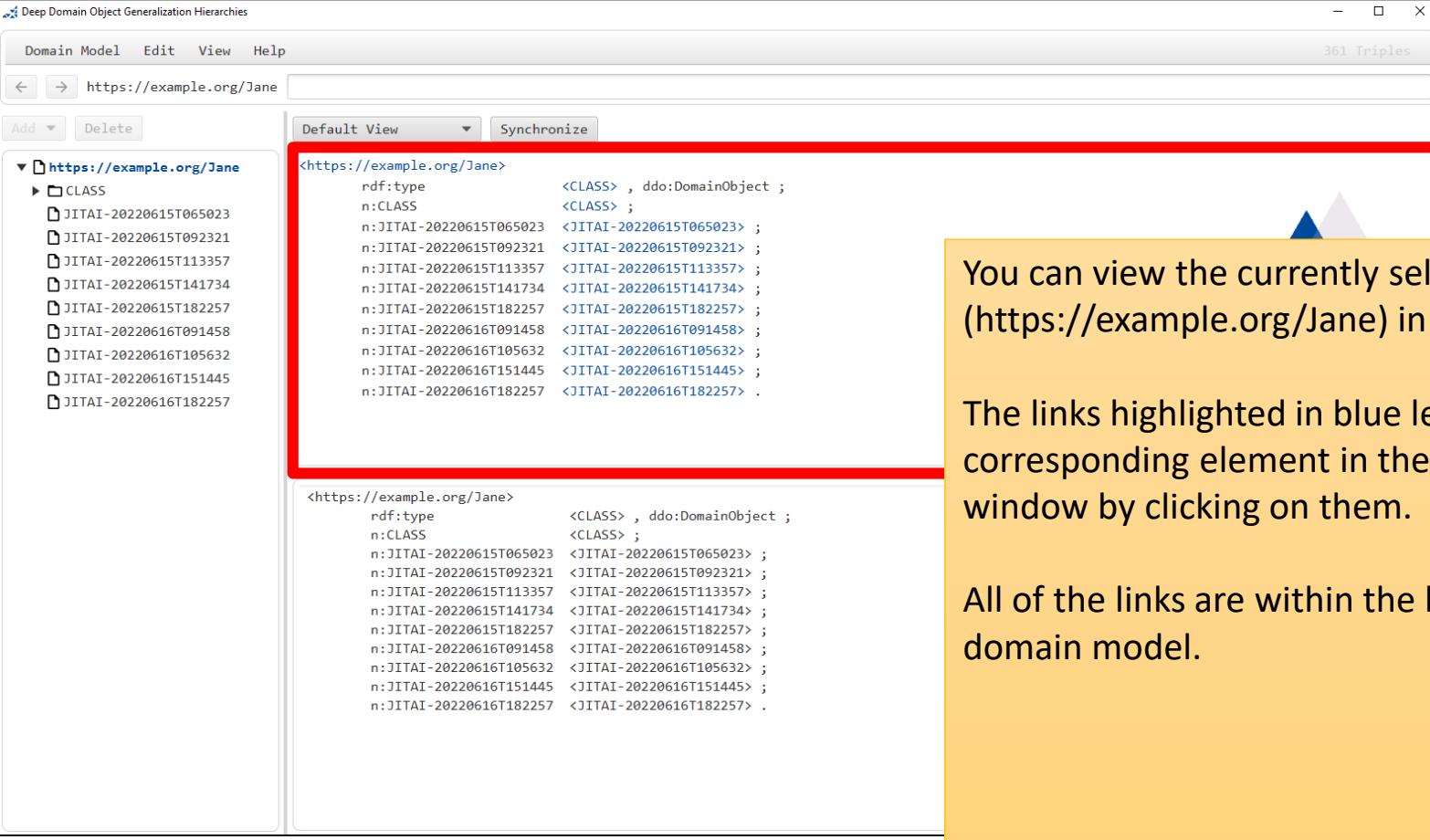
The screenshot shows the DDO-Playground application window titled "Deep Domain Object Generalization Hierarchies". The URL "https://example.org/Jane" is entered in the address bar. The interface includes a menu bar with "Domain Model", "Edit", "View", and "Help", and a status bar indicating "362 Triples". On the left, a tree view displays classification levels. Red arrows point to three specific symbols: a single square icon for an individual (level 0), a square with a horizontal line for a class object (level 1), and a square with a vertical line for a meta class (level 2). The main pane shows RDF triples for the resource <https://example.org/Jane>, including statements about its type and various sub-classes and individuals.

The tree view distinguishes between classification levels by different symbols:

-  Individual (level 0)
-  Class object (level 1)
-  Meta Class (level 2)

DDO-Playground

Element View



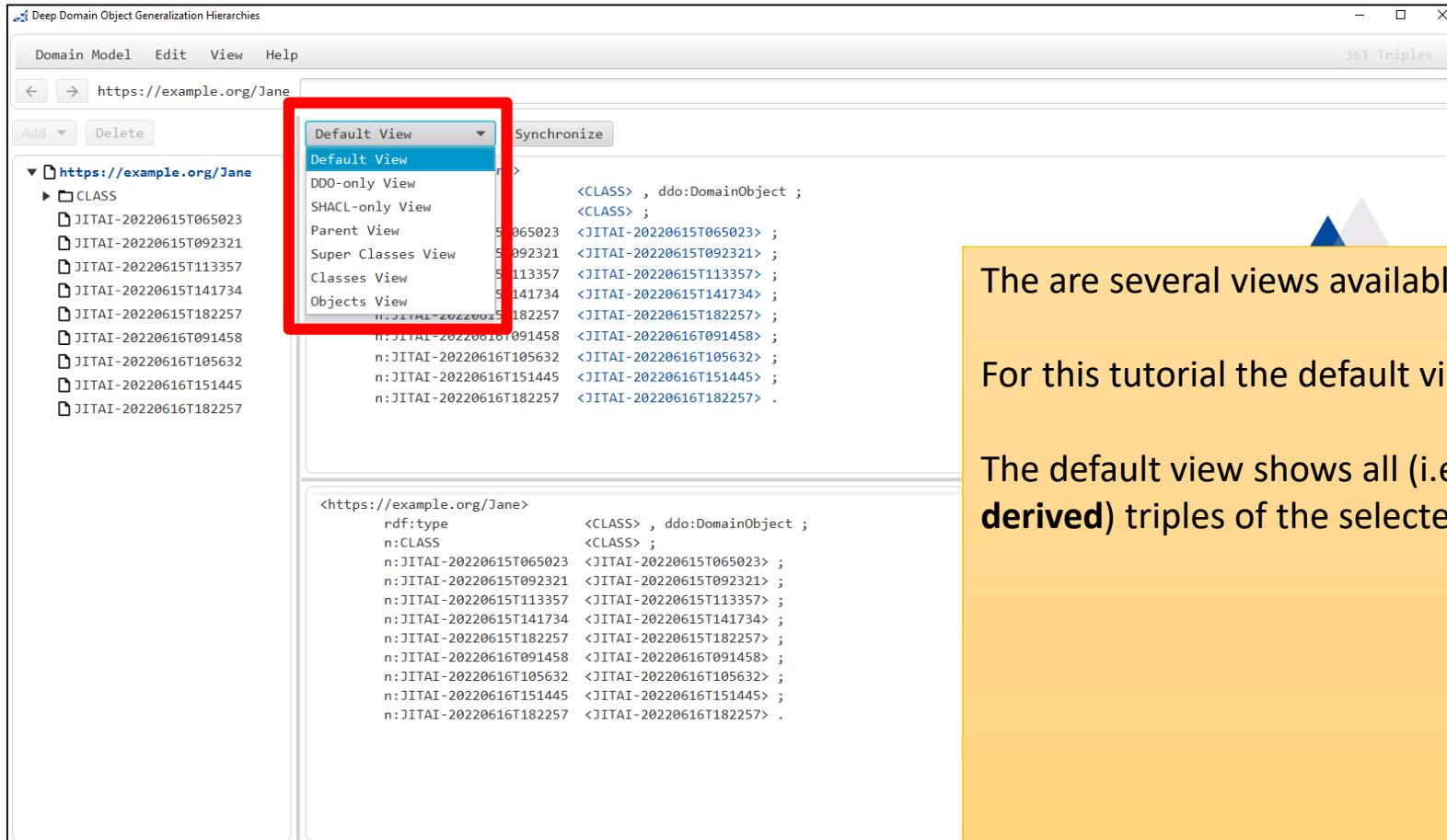
The screenshot shows the DDO-Playground Element View application interface. The title bar reads "Deep Domain Object Generalization Hierarchies". The menu bar includes "Domain Model", "Edit", "View", and "Help". The address bar shows the URL "https://example.org/Jane". The main window has a toolbar with "Add", "Delete", "Default View", and "Synchronize" buttons. On the left, a tree view under "https://example.org/Jane" shows a "CLASS" node expanded, with several child nodes listed: JITAI-20220615T065023, JITAI-20220615T092321, JITAI-20220615T113357, JITAI-20220615T141734, JITAI-20220615T182257, JITAI-20220616T091458, JITAI-20220616T105632, JITAI-20220616T151445, and JITAI-20220616T182257. The right pane displays the RDF triples for the selected element, with the entire list highlighted by a red rectangle. A yellow callout box on the right contains three text snippets.

You can view the currently selected element (<https://example.org/Jane>) in RDF/Turtle.

The links highlighted in blue lead to the corresponding element in the application window by clicking on them.

All of the links are within the base URI of the domain model.

DDO-Playground Views

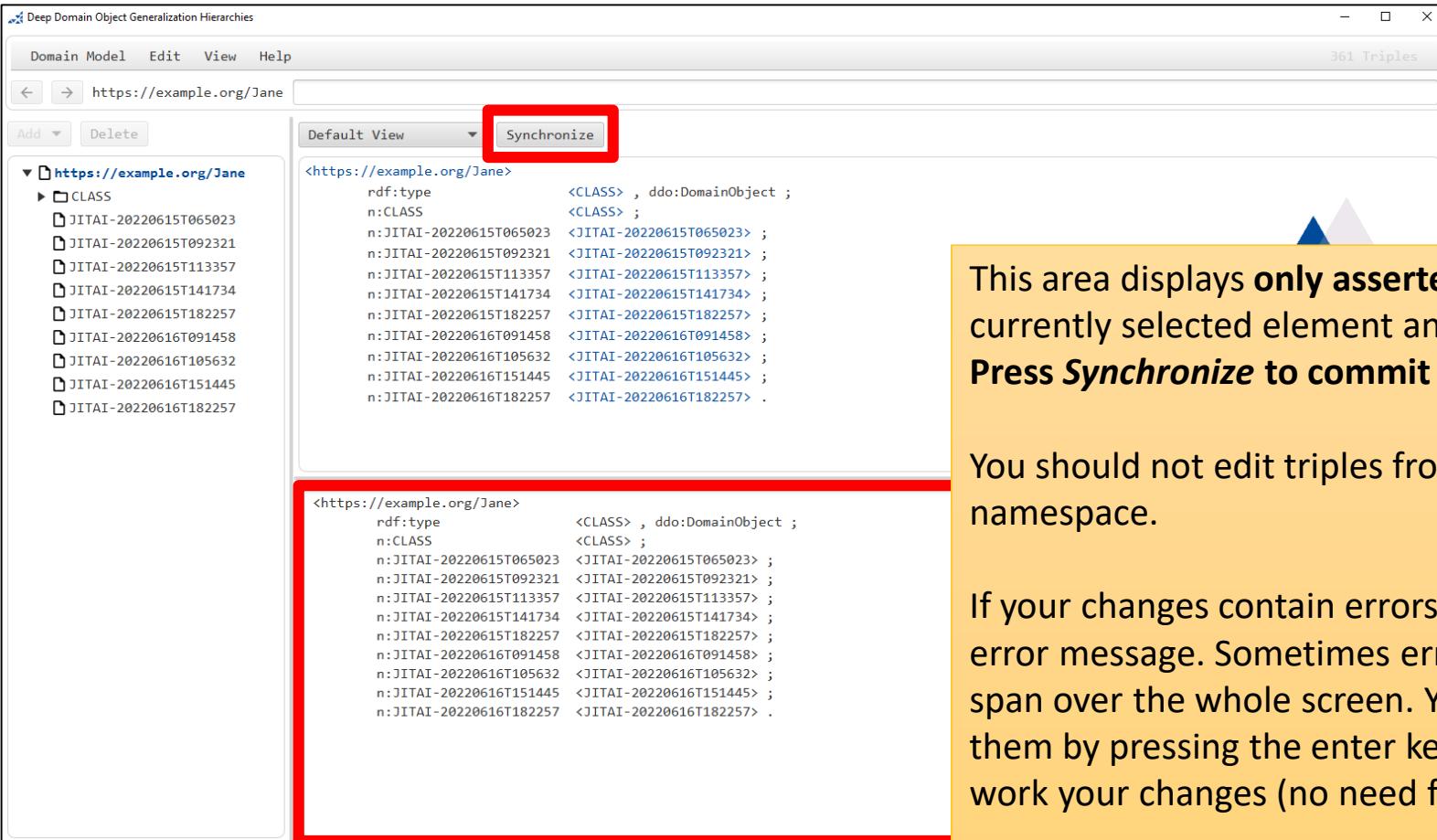


There are several views available.

For this tutorial the default view is sufficient.

The default view shows all (i.e., **asserted and derived**) triples of the selected element.

DDO-Playground Editor



The screenshot shows the Deep Domain Object Generalization Hierarchy (DDO) editor interface. The title bar reads "Deep Domain Object Generalization Hierarchies". The menu bar includes "Domain Model", "Edit", "View", and "Help". The address bar shows the URL "https://example.org/Jane". The toolbar has "Add" and "Delete" buttons, and a "Synchronize" button which is highlighted with a red box. The left sidebar shows a tree view under "https://example.org/Jane" with a node "CLASS" expanded, listing various JITAID entries. The main pane displays asserted triples for the selected element. A large yellow callout box on the right provides instructions:

This area displays **only asserted** triples of the currently selected element and can be **edited**. Press **Synchronize** to commit your changes.

You should not edit triples from the ddo: or n: namespace.

If your changes contain errors, you will face an error message. Sometimes error messages span over the whole screen. You can close them by pressing the enter key. You can re-work your changes (no need for an undo).

DDO-Playground

DDO-specific Constructs

The screenshot shows the DDO-Playground application window titled "Deep Domain Object Generalization Hierarchies". The menu bar includes "Domain Model", "Edit", "View", and "Help". A status bar at the top right indicates "361 Triples". The main area has tabs for "Default View" and "Synchronize". On the left, a tree view shows the hierarchy under "<https://example.org/Jane>". It starts with a "CLASS" node, which branches into several specific class instances like "JITAI-20220615T065023", "JITAI-20220615T092321", etc. On the right, two large text boxes show the RDF triples for these classes. The top box is for "<https://example.org/Jane>" and the bottom box is for another instance of the same class. Both boxes show the same set of triples:

```
rdf:type <CLASS> , ddo:DomainObject ;
n:CLASS <CLASS> ;
n:JITAI-20220615T065023 <JITAI-20220615T065023> ;
n:JITAI-20220615T092321 <JITAI-20220615T092321> ;
n:JITAI-20220615T113357 <JITAI-20220615T113357> ;
n:JITAI-20220615T141734 <JITAI-20220615T141734> ;
n:JITAI-20220615T182257 <JITAI-20220615T182257> ;
n:JITAI-20220616T091458 <JITAI-20220616T091458> ;
n:JITAI-20220616T105632 <JITAI-20220616T105632> ;
n:JITAI-20220616T151445 <JITAI-20220616T151445> ;
n:JITAI-20220616T182257 <JITAI-20220616T182257> .
```

The DDO-Playground is intended for DDO-based multi-level modeling.

It supports further DDO-specific constructs which are not necessary for this tutorial.

We introduce DDO-specific construct only if necessary.

In most cases we can get by with `rdf:type` and `rdfs:subClassOf`, and can ignore other properties.

Use Case JITAI Clabjects

Deep Domain Object Generalization Hierarchies

Domain Model Edit View Help

https://example.org/Jane

Add Delete Default View Synchronize

https://example.org/Jane

CLASS

- JITAI
- >ExerciseSnackingJITAI
- >RunningJITAI
- >>BaseRunningJITAI
- >>HillRunningJITAI
- >>RecoveryRunningJITAI
- >>WalkingJITAI
- >>BriskWalkingJITAI
- >>NordicWalkingJITAI
- >>PowerWalkingJITAI

JITAI-20220615T092321

https://example.org/Jane

```

<https://example.org/Jane> rdf:type <CLASS> , ddo:DomainObject ;
n:CLASS <CLASS> ;
n:JITAI-20220615T065023 <JITAI-20220615T065023> ;
n:JITAI-20220615T092321 <JITAI-20220615T092321> ;
n:JITAI-20220615T113357 <JITAI-20220615T113357> ;
n:JITAI-20220615T141734 <JITAI-20220615T141734> ;
n:JITAI-20220615T182257 <JITAI-20220615T182257> ;
n:JITAI-20220616T091458 <JITAI-20220616T091458> ;
n:JITAI-20220616T105632 <JITAI-20220616T105632> ;
n:JITAI-20220616T151445 <JITAI-20220616T151445> ;
n:JITAI-20220616T182257 <JITAI-20220616T182257> .

```

JITAI-20220615T113357

JITAI-20220615T141734

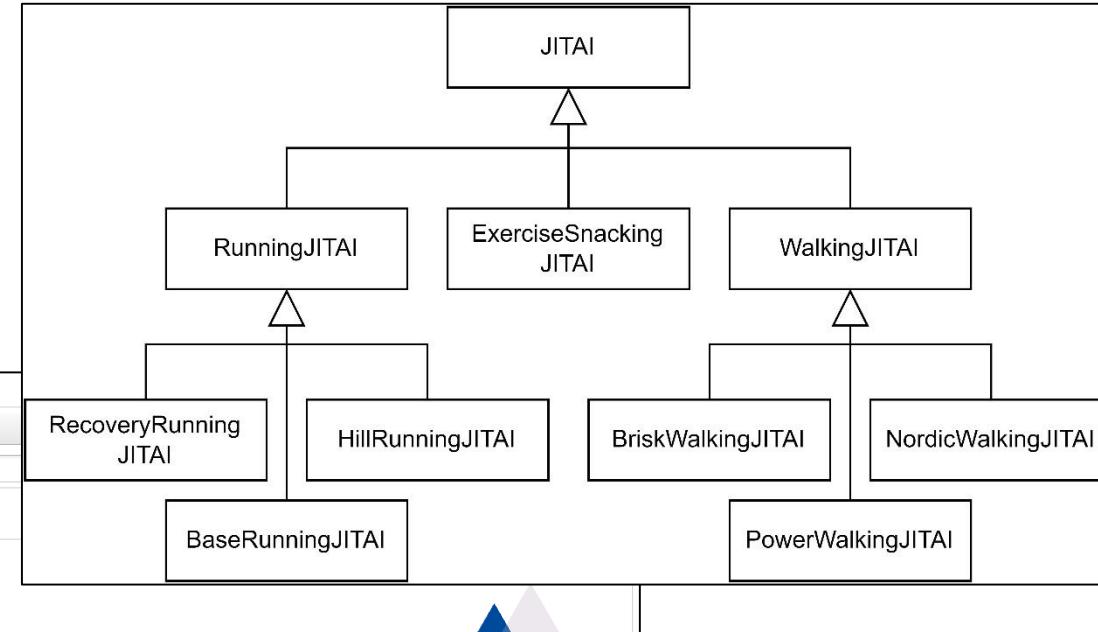
JITAI-20220615T182257

JITAI-20220616T091458

JITAI-20220616T105632

JITAI-20220616T151445

JITAI-20220616T182257



In this example, we were given access to Jane's JITAI data for the purpose of analysis and personalization.

The resource <<https://example.org/Jane>> is a representation of Jane.

Jane's singleton class (<CLASS>) introduces a class hierarchy for different JITAI classes which is represented by the class hierarchy above.

The class hierarchy is encoded using rdfs:subClassOf properties.

Note: The URI of an element is related to its arrangement in the tree view.

Use Case JITAI Occurrences

The screenshot shows the Deep Domain Object Generalization Hierarchies (DDOGH) interface. The top navigation bar includes 'Domain Model', 'Edit', 'View', and 'Help'. The address bar shows the URL <https://example.org/Jane> and the path `/JITAI-20220615T065023`. A status bar at the top right indicates '362 Triples'. The main area has tabs for 'Default View' and 'Synchronize'. On the left, a tree view shows a node for `JITAI-20220615T065023`, which is expanded to show several sub-nodes, all of which are highlighted with a red box. On the right, the detailed view for `<JITAI-20220615T065023>` is shown, with its properties also highlighted with a red box. The properties listed are:

```
rdf:type ddo:Occurrence , <CLASS/BriskWalkingJITAI> ;
ddo:instanceOf n:BriskWalkingJITAI ;
ddo:name "JITAI-20220615T065023" ;
ddo:parent <https://example.org/Jane> ;
<duration> 30 ;
<intensity> 11 .
```

Below this, another instance of the same class is shown with identical properties.

There are 9 JITAI occurrences for Jane within 2 days.

These occurrences are instances of different JITAI classes, e.g., `<JITAI-20220615T065023>` is instance of `<CLASS/BriskWalkingJITAI>`.

The properties of interest are `<duration>` and `<intensity>`.

`<duration>` is the JITAI system's estimate of the duration of the suggested activity in minutes.

`<intensity>` is the JITAI system's estimate of the perceived intensity of the suggested activity according to the BORG scale.

Use Case

Duration & Intensity

Deep Domain Object Generalization Hierarchies

Domain Model Edit View Help

https://example.org/Jane /CLASS/JITAI

Add Delete Default View Synchronize

https://example.org/Jane CLASS JITAI

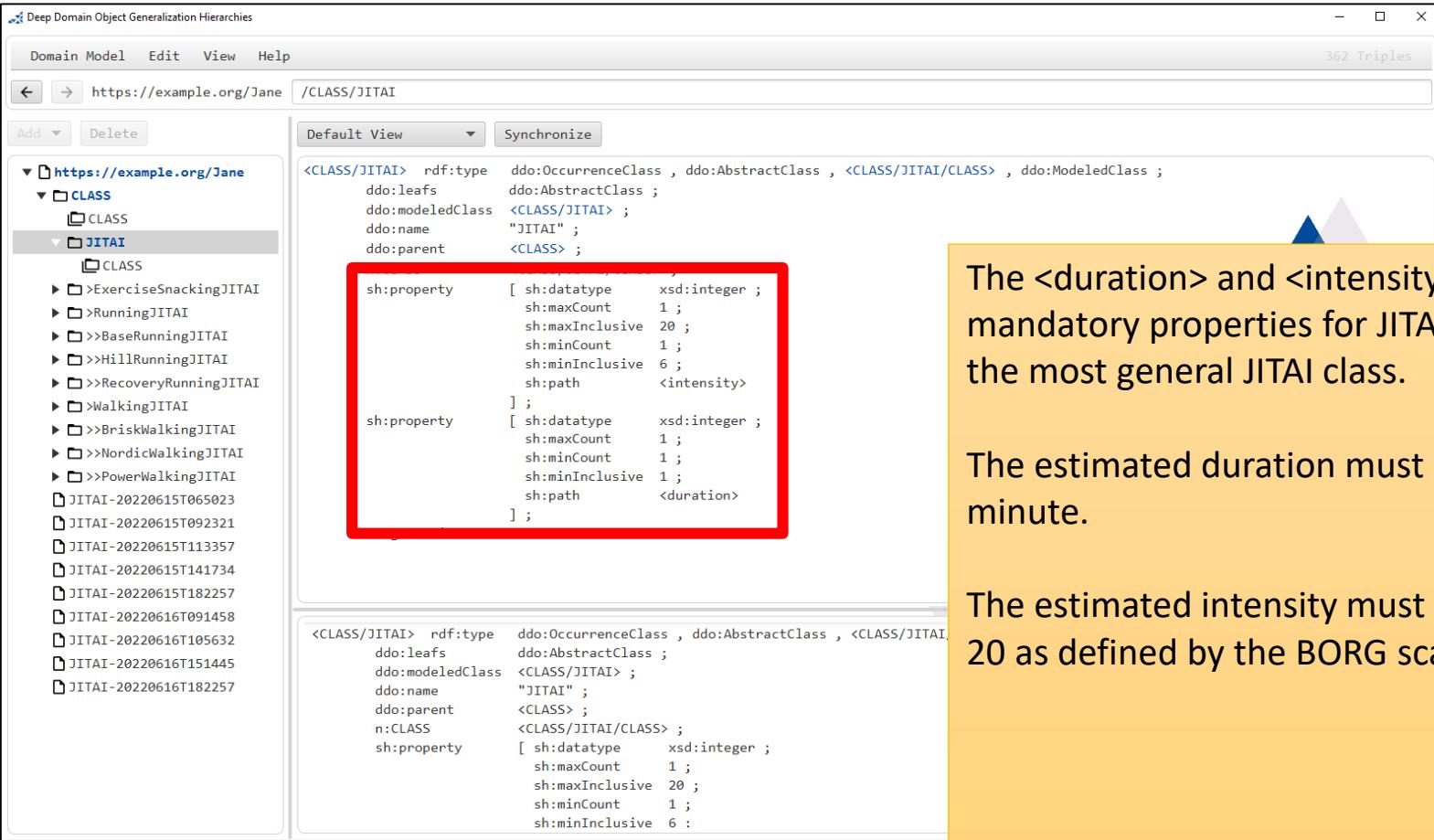
<CLASS/JITAI> rdf:type ddo:OccurrenceClass , ddo:AbstractClass , <CLASS/JITAI/CLASS> , ddo:ModeledClass ;
ddo:leafs ddo:AbstractClass ;
ddo:modeledClass <CLASS/JITAI> ;
ddo:name "JITAI" ;
ddo:parent <CLASS> ;

sh:property [sh:datatype xsd:integer ;
sh:maxCount 1 ;
sh:maxInclusive 20 ;
sh:minCount 1 ;
sh:minInclusive 6 ;
sh:path <intensity>] ;

sh:property [sh:datatype xsd:integer ;
sh:maxCount 1 ;
sh:minCount 1 ;
sh:minInclusive 1 ;
sh:path <duration>] ;

<CLASS/JITAI> rdf:type ddo:OccurrenceClass , ddo:AbstractClass , <CLASS/JITAI> , ddo:ModeledClass ;
ddo:leafs ddo:AbstractClass ;
ddo:modeledClass <CLASS/JITAI> ;
ddo:name "JITAI" ;
ddo:parent <CLASS> ;
n:CLASS <CLASS/JITAI/CLASS> ;
sh:property [sh:datatype xsd:integer ;
sh:maxCount 1 ;
sh:maxInclusive 20 ;
sh:minCount 1 ;
sh:minInclusive 6 :] ;

362 Triples

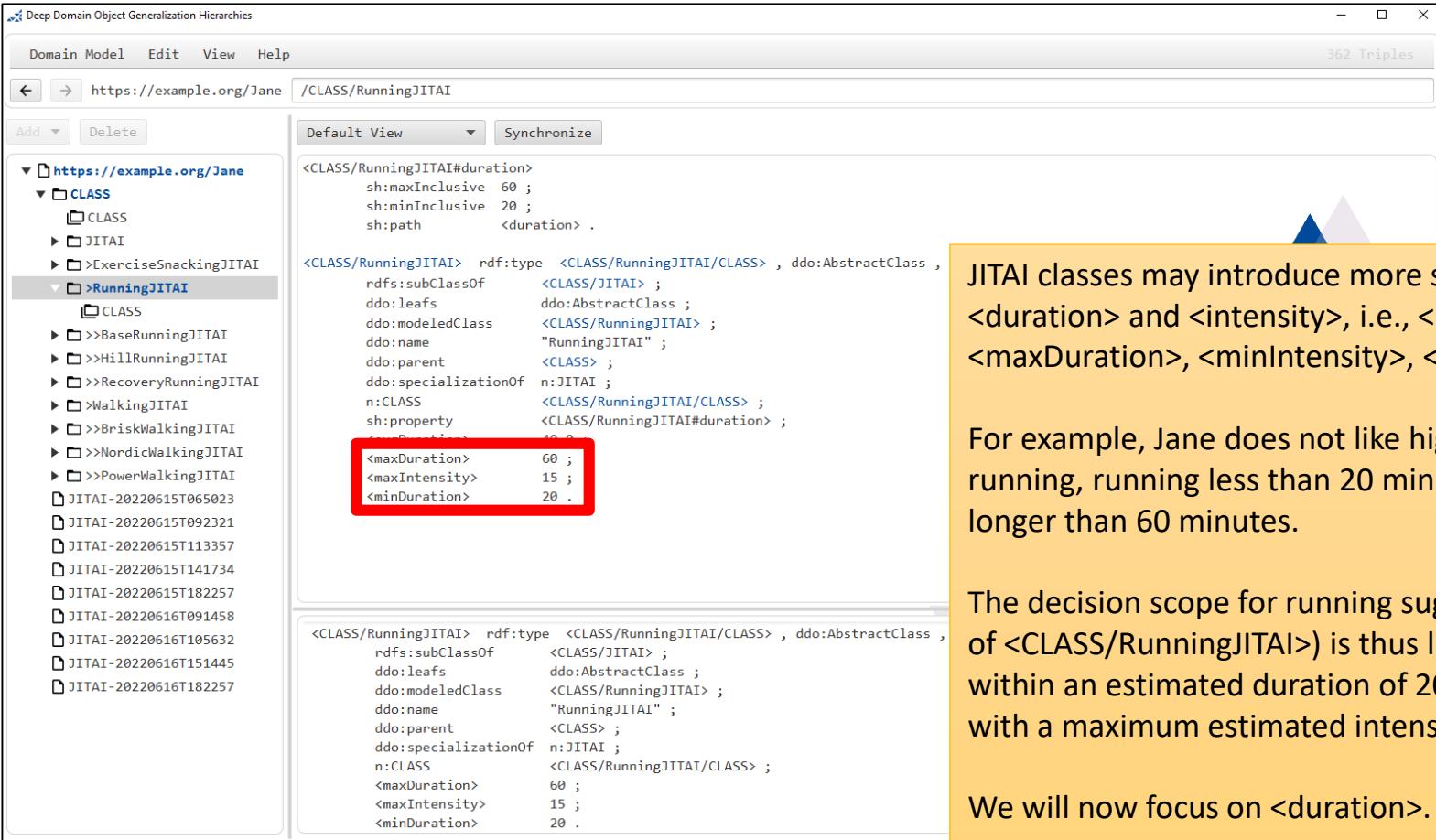


The <duration> and <intensity> are defined as mandatory properties for JITAI occurrences in the most general JITAI class.

The estimated duration must be more than 1 minute.

The estimated intensity must be within 6 and 20 as defined by the BORG scale.

Multi-level Modeling JITAI Clabjects revisited



The screenshot shows a software interface for managing domain models. The left pane displays a tree view of classes under <https://example.org/Jane>, including **CLASS**, **JITAI**, and several subclasses like **RunningJITAI**. The right pane shows the RDF triples for the **RunningJITAI** class, specifically its properties and restrictions:

```
<CLASS/RunningJITAI#duration>
    sh:maxInclusive 60 ;
    sh:minInclusive 20 ;
    sh:path <duration> .
```

```
<CLASS/RunningJITAI> rdf:type <CLASS/RunningJITAI/CLASS> , ddo:AbstractClass ,
    rdfs:subClassOf <CLASS/JITAI> ;
    ddo:leafs ddo:AbstractClass ;
    ddo:modeledClass <CLASS/RunningJITAI> ;
    ddo:name "RunningJITAI" ;
    ddo:parent <CLASS> ;
    ddo:specializationOf n:JITAI ;
    n:CLASS <CLASS/RunningJITAI/CLASS> ;
    sh:property <CLASS/RunningJITAI#duration> ;
    <maxDuration> 60 ;
    <maxIntensity> 15 ;
    <minDuration> 20 .
```

A red box highlights the duration restrictions (`<maxDuration> 60 ;`, `<maxIntensity> 15 ;`, `<minDuration> 20 .`) in the triple list.

JITAI classes may introduce more specific limits for `<duration>` and `<intensity>`, i.e., `<minDuration>`, `<maxDuration>`, `<minIntensity>`, `<maxIntensity>`.

For example, Jane does not like high intensity running, running less than 20 minutes and running longer than 60 minutes.

The decision scope for running suggestions (instances of `<CLASS/RunningJITAI>`) is thus limited to activities within an estimated duration of 20 to 60 minutes and with a maximum estimated intensity of 15.

We will now focus on `<duration>`.

34

Multi-level Modeling Powertype (1)

The screenshot shows the 'Deep Domain Object Generalization Hierarchies' application interface. The left sidebar displays a tree view of domain objects under 'https://example.org/Jane'. The 'CLASS' node has several sub-nodes, including 'JITAI', which is expanded to show its subclasses: 'ExerciseSnackingJITAI', 'RunningJITAI', 'BaseRunningJITAI', 'HillRunningJITAI', 'RecoveryRunningJITAI', 'WalkingJITAI', 'BriskWalkingJITAI', 'NordicWalkingJITAI', 'PowerWalkingJITAI', and two specific instances: 'JITAI-20220615T092321' and 'JITAI-20220615T113357'. The right panel shows the RDF triples for the 'CLASS/JITAI/CLASS' meta-class. A red box highlights the following triple patterns:

```
sh:property [ sh:datatype xsd:integer ; sh:maxCount 1 ; sh:minInclusive 1 ; sh:path <maxDuration> ] ;  
sh:property [ sh:datatype xsd:integer ; sh:maxCount 1 ; sh:minInclusive 1 ; sh:path <minDuration> ] ;
```

These patterns define the powertype for the most general JITAI class, allowing for limiting values for duration.

Since JITAI classes are clabjects with values for limiting <duration>, these properties are introduced at their respective meta class, i.e., powertype.

For each clabject, there is a powertype. The clabject and all its subclasses are instances of the powertype. The specialization between powertypes mirrors the JITAI class hierarchy. The URI of a powertype corresponds to the clabject's URI + /CLASS.

For example, the powertype of the most general JITAI class is <CLASS/JITAI/CLASS> and has all the JITAI classes as instances. Therefore, the property shapes for limiting <duration> and <intensity> are introduced there.

Multi-Level Modeling PowerType (2)

The screenshot shows the 'Deep Domain Object Generalization Hierarchies' application interface. The left pane displays a tree view of domain classes under 'https://example.org/Jane /CLASS/JITAI/CLASS'. The right pane shows a 'SPARQL Query' results table with a red box highlighting the 'object' column.

SPARQL Query Results:

object
<CLASS/JITAI>
<CLASS/RunningJITAI>
<CLASS/HillRunningJITAI>
<CLASS/BaseRunningJITAI>
<CLASS/RecoveryRunningJITAI>
<CLASS/ExerciseSnackingJITAI>
<CLASS/WalkingJITAI>
<CLASS/BriskWalkingJITAI>
<CLASS/NordicWalkingJITAI>
<CLASS/PowerWalkingJITAI>

Text Overlay:

Using the default query of the SPARQL query editor, we can check all instances of the <CLASS/JITAI/CLASS> ...

Multi-Level Modeling Powertype (3)

The screenshot shows a software interface for managing domain object generalization hierarchies. The title bar reads "Deep Domain Object Generalization Hierarchies". The main window has a toolbar with "Domain Model", "Edit", "View", and "Help" buttons, and a status bar showing "362 Triples". The URL in the address bar is "https://example.org/Jane /CLASS/RunningJITAI/CLASS".

The left pane displays a tree view of the domain model under "https://example.org/Jane". It includes categories like "CLASS", "JITAI", "ExerciseName", and "RunningJITAI". A specific node under "RunningJITAI" is selected and highlighted in blue.

The central pane contains a "SPARQL Query" editor with the following code:

```
BASE <https://example.org/Jane/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ddo: <http://dke.jku.at/ddo#>
PREFIX n: <http://dke.jku.at/ddo-hasNamedChild#>
SELECT ?object
WHERE {
    ?object rdf:type/rdfs:subClassOf* <CLASS/RunningJITAI/CLASS>
}
```

The right pane shows the results of the SPARQL query. A red box highlights the first result, which is a dashed box containing the text "object". Below it is a list of URIs:

```
<CLASS/RunningJITAI>
<CLASS/HillRunningJITAI>
<CLASS/BaseRunningJITAI>
<CLASS/RecoveryRunningJITAI>
```

A large yellow box covers the entire right pane, containing the text "... or of <CLASS/RunningJITAI/CLASS>." This indicates that the results shown are for the class itself, not its subclasses.

Multi-Level Modeling Powertype (4)

The screenshot shows a software interface for managing domain models. The title bar reads "Deep Domain Object Generalization Hierarchies". The main window has a toolbar with "Domain Model", "Edit", "View", and "Help" buttons. Below the toolbar is a URL bar showing "<https://example.org/Jane> /CLASS/BaseRunningJITAI/CLASS". A status bar at the top right indicates "362 Triples".

The left pane displays a tree view of the domain model structure under "<https://example.org/>". The "CLASS" node is expanded, showing sub-nodes like "JITAI", "ExerciseSneaker", "RunningJITAI", and "BaseRunningJITAI". The "BaseRunningJITAI" node is also expanded, showing its subclasses: "HillRunnin", "RecoveryRun", "WalkingJITAI", "BriskWalk", "NordicWalk", and "PowerWalk". Below these are several "JITAI" entries with timestamps.

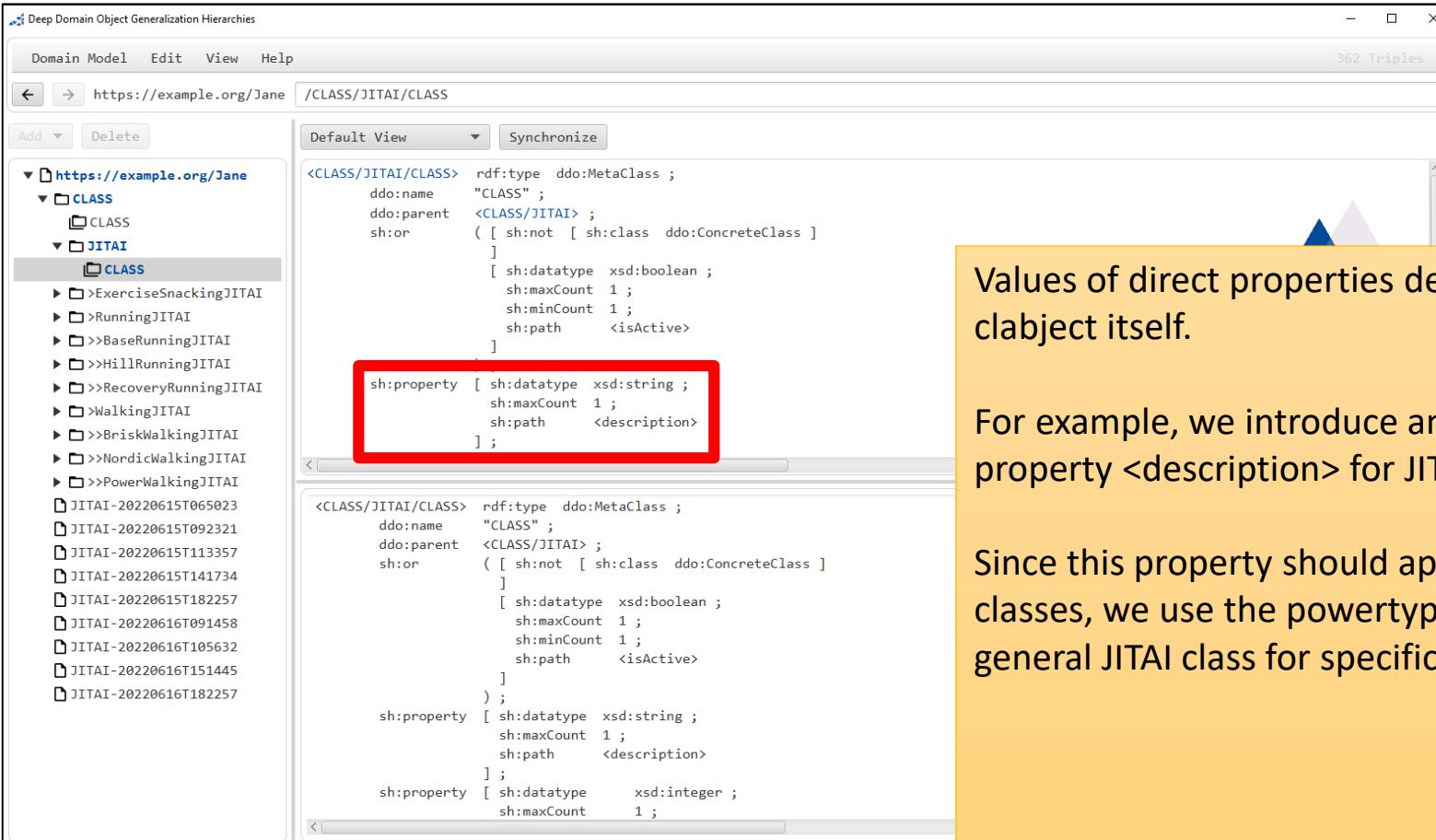
The central pane contains a "Default View" tab and a "SPARQL Query" tab. The SPARQL query is:

```
BASE <https://example.org/Jane/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ddo: <http://dke.jku.at/ddo#>
PREFIX n: <http://dke.jku.at/ddo-hasNamedChild#>
SELECT ?object
WHERE {
    ?object rdf:type/rdfs:subClassOf* <CLASS/BaseRunningJITAI/CLASS>
}
```

The results of the query are displayed in a table. One row from the table is highlighted with a red border, showing the path: "object" → "<CLASS/BaseRunningJITAI/CLASS>". A large yellow callout box points to this row with the text "... or of <CLASS/BaseRunningJITAI/CLASS>." The table also shows other rows corresponding to the subclasses listed in the SPARQL query results.

Multi-Level Modeling

Direct Properties (1)



The screenshot shows a software interface for managing domain models. On the left, there's a tree view of classes under a base URL. A specific class, 'JITAI CLASS', is selected. The main pane displays the RDF triples for this class. A red box highlights a section of the triple definition:

```
<CLASS/JITAI/CLASS> sh:property [ sh:datatype xsd:string ;  
                                         sh:maxCount 1 ;  
                                         sh:path <description>  
                                       ] ;
```

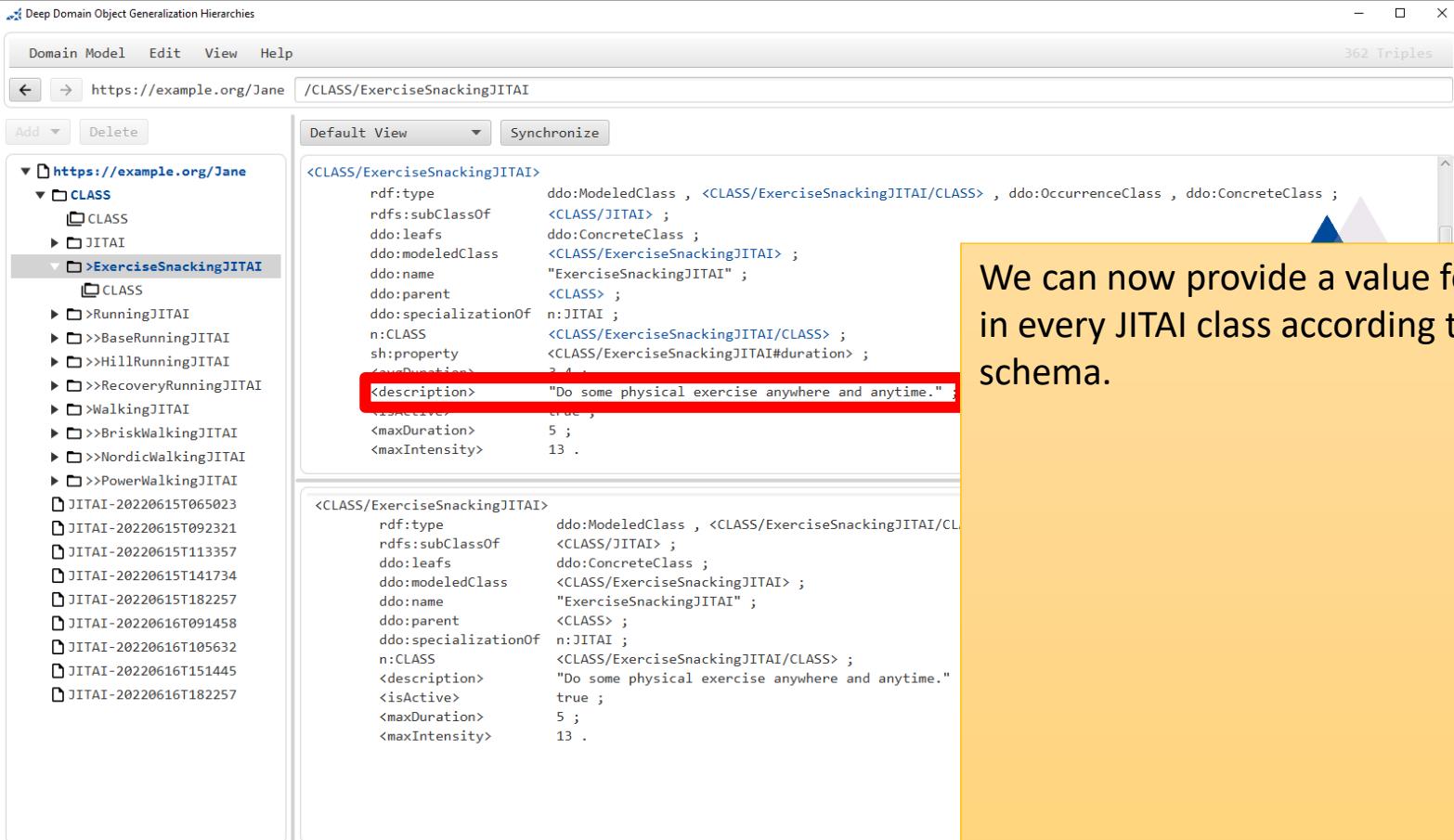
Values of direct properties describe the clobject itself.

For example, we introduce an optional direct property <description> for JITAI classes.

Since this property should apply to all JITAI classes, we use the powertype of the most general JITAI class for specification.

Multi-Level Modeling

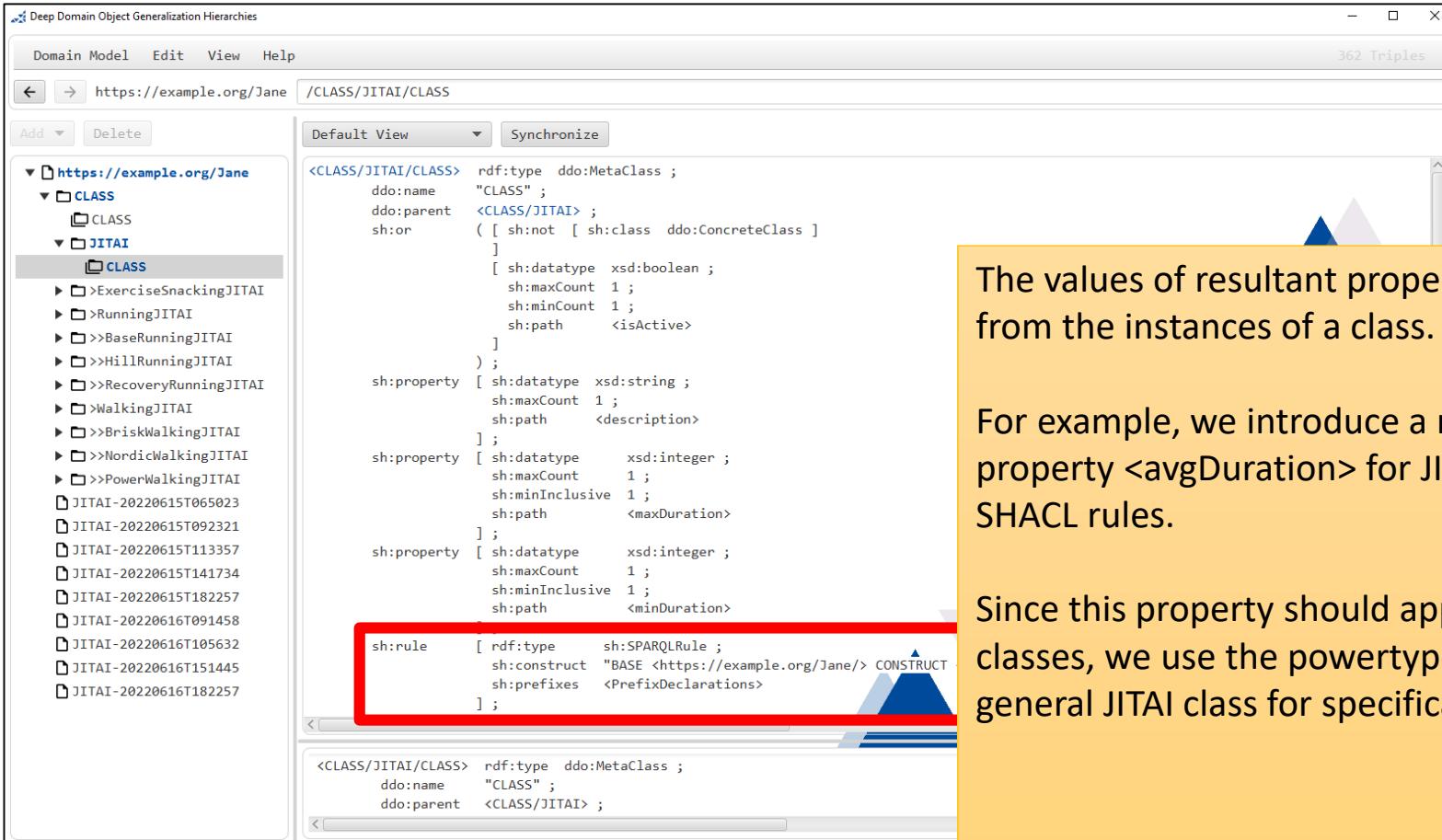
Direct Properties (2)



The screenshot shows a software interface for managing domain models. On the left, a tree view displays a class hierarchy under the base URL <https://example.org/Jane>. The hierarchy includes the root `JITAI`, which contains `ExerciseSnackingJITAI` and several subclasses of `CLASS` such as `>RunningJITAI`, `>>BaseRunningJITAI`, etc. Below `ExerciseSnackingJITAI` are further subclasses like `>>RecoveryRunningJITAI`, `>>WalkingJITAI`, etc. On the right, two detailed views of the `ExerciseSnackingJITAI` class are shown. Each view lists properties with their values, such as `rdf:type` being `ddo:ModeledClass`, `rdfs:subClassOf` being `<CLASS/JITAI>`, and `ddo:leafs` being `ddo:ConcreteClass`. A red box highlights the `<description>` property with the value "Do some physical exercise anywhere and anytime." This property is defined in both the top and bottom instances of the class. A yellow callout box to the right states: "We can now provide a value for this property in every JITAI class according to the specified schema."

We can now provide a value for this property in every JITAI class according to the specified schema.

Multi-Level Modeling Resultant Properties (1)



The screenshot shows a software interface for managing domain models. On the left, a tree view displays a hierarchy of classes under 'https://example.org/Jane'. A specific node, 'CLASS' under 'JITAI', is selected. The main pane shows the SHACL validation results for this class. The results table lists several properties with their constraints. A red box highlights a row for the 'sh:rule' property, which contains a SPARQL rule definition:

```
sh:rule [ rdf:type sh:SPARQLRule ;
          sh:construct "BASE <https://example.org/Jane/> CONSTRUCT
                        PREFIX declarations <PrefixDeclarations>
                        ?s ?p ?o .
                      WHERE ?s ?p ?o ."
          sh:prefixes <PrefixDeclarations> ] ;
```

The values of resultant properties are derived from the instances of a class.

For example, we introduce a resultant property `<avgDuration>` for JITAI classes using SHACL rules.

Since this property should apply to all JITAI classes, we use the powertype of the most general JITAI class for specification.

Multi-Level Modeling Resultant Properties (2)

This property is now automatically derived in every JITAI class.

Multi-Level Modeling

Regularity Properties (1)

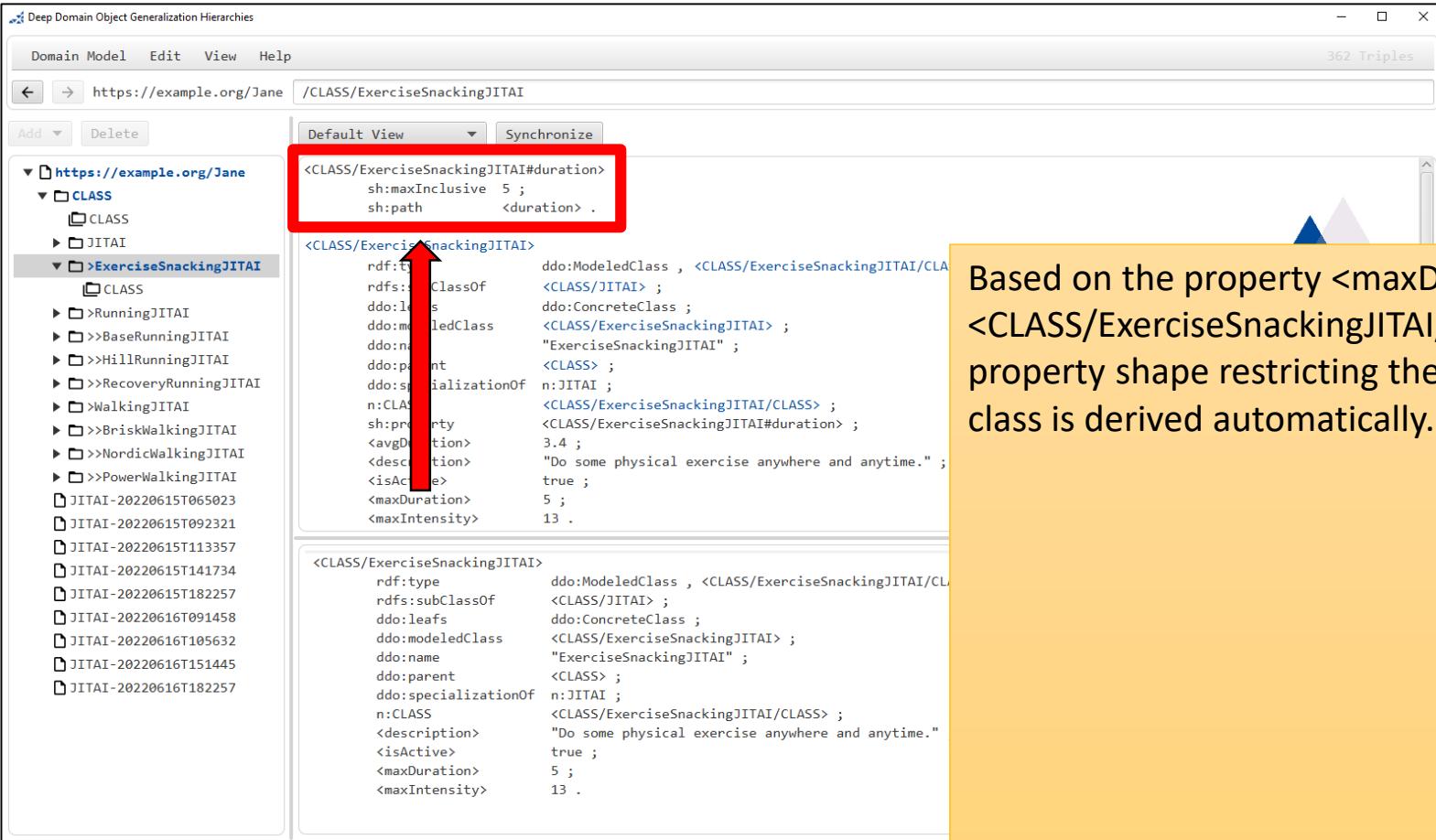
The screenshot shows a software interface for managing domain models. On the left, a tree view displays a hierarchy under `https://example.org/Jane`, with `CLASS` and `JITAI` as main categories. Under `JITAI`, there are numerous sub-categories like `>ExerciseSnackingJITAI`, `>RunningJITAI`, etc. On the right, the main pane shows SPARQL code for defining rules. A red box highlights a section of the code where multiple `sh:rule` statements are defined. Below this, another code block shows meta-class definitions for `<CLASS/JITAI/CLASS>`. A yellow callout box contains explanatory text about regularity properties.

Regularity properties regulate instances of the instances.

We now use meta class `<CLASS/JITAI/CLASS>` to define rules taking `<minDuration>` and `<maxDuration>` as input and deriving a property shape with the property's value as `sh:minInclusive` or `sh:maxInclusive`.

Multi-Level Modeling

Regularity Properties (2)



The screenshot shows a software interface for managing domain models. On the left, a tree view displays a hierarchy of classes under the base URL <https://example.org/Jane>. One node is expanded to show its properties. A red box highlights a specific property shape definition:

```
<CLASS/ExerciseSnackingJITAI#duration>
    sh:maxInclusive 5 ;
    sh:path <duration> .
```

An arrow points from this highlighted section to the explanatory text on the right.

Based on the property <maxDuration> in <CLASS/ExerciseSnackingJITAI/CLASS> a property shape restricting the instances of this class is derived automatically.

Multi-Level Modeling Conflict Detection (1)

The screenshot shows a software interface for managing domain models. The title bar reads "Deep Domain Object Generalization Hierarchies". The menu bar includes "Domain Model", "Edit", "View", and "Help". The address bar shows the URL "https://example.org/Jane /CLASS/RunningJITAI" and indicates there are "362 Triples".

The left pane displays a tree view of the domain model:

- https://example.org/Jane
- CLASS
 - JITAI
 - >ExerciseSnackingJITAI
 - >RunningJITAI
 - CLASS
 - >>BaseRunningJITAI
 - >>HillRunningJITAI
 - >>RecoveryRunningJITAI
 - >>WalkingJITAI
 - >>BriskWalkingJITAI
 - >>NordicWalkingJITAI
 - >>PowerWalkingJITAI
 - JITAI-20220615T065023
 - JITAI-20220615T092321
 - JITAI-20220615T113357
 - JITAI-20220615T141734
 - JITAI-20220615T182257
 - JITAI-20220616T091458
 - JITAI-20220616T105632
 - JITAI-20220616T151445
 - JITAI-20220616T182257

The right pane shows the RDF triples for the selected node, <CLASS/RunningJITAI>. Two definitions are present, both with a <minDuration> of 20:

```
<CLASS/RunningJITAI#duration>
    sh:maxInclusive 60 ;
    sh:minInclusive 20 ;
    sh:path      <duration> .
```

```
<CLASS/RunningJITAI> rdf:type <CLASS/RunningJITAI/CLASS> , ddo:AbstractClass ,
    rdfs:subClassOf      <CLASS/JITAI> ;
    ddo:leafs            ddo:AbstractClass ;
    ddo:modeledClass    <CLASS/RunningJITAI> ;
    ddo:name             "RunningJITAI" ;
    ddo:parent           <CLASS> ;
    ddo:specializationOf n:JITAI ;
    n:CLASS              <CLASS/RunningJITAI/CLASS> ;
    sh:property          <CLASS/RunningJITAI#duration> ;
    <avgDuration>        40.0 ;
    <maxDuration>        60 ;
    <maxIntensity>       15 ;
    <minDuration>        20 .
```

```
<CLASS/RunningJITAI> rdf:type <CLASS/RunningJITAI/CLASS> , ddo:AbstractClass ,
    rdfs:subClassOf      <CLASS/JITAI> ;
    ddo:leafs            ddo:AbstractClass ;
    ddo:modeledClass    <CLASS/RunningJITAI> ;
    ddo:name             "RunningJITAI" ;
    ddo:parent           <CLASS> ;
    ddo:specializationOf n:JITAI ;
    n:CLASS              <CLASS/RunningJITAI/CLASS> ;
    <maxDuration>        60 ;
    <maxIntensity>       15 ;
    <minDuration>        20 .
```

A callout box highlights the <minDuration> value of 20, indicating a conflict between the two definitions.

<CLASS/RunningJITAI> has a <minDuration> of 20 minutes, while ...

Multi-Level Modeling Conflict Detection (2)

The screenshot shows a software interface for managing domain models. On the left, a tree view displays various classes and their subclasses under the root node `https://example.org/Jane`. A specific class, `>>RecoveryRunningJITAI`, is selected. The main pane on the right shows the RDF triples for this class and its superclasses. A red box highlights a specific triple: `<maxDuration> 15 .` This triple is present in both the `>>RecoveryRunningJITAI` class definition and in the definition of its superclass, `>>RunningJITAI`.

`<CLASS/RunningJITAI>` has a `<minDuration>` of 20 minutes, while ...

... `<CLASS/RecoveryRunningJITAI>`, a subclass of `<CLASS/RunningJITAI>`, introduces a `<maxDuration>` of 15 minutes.

How can we detect this conflict?

Multi-Level Modeling Conflict Detection (3)

The screenshot shows a software interface for managing domain models. On the left, a tree view displays a hierarchy under `https://example.org/Jane`, with `CLASS` and `JITAI` as main categories. Under `JITAI`, there are several subclasses like `>ExerciseSnackingJITAI`, `>RunningJITAI`, etc. On the right, the main pane shows a list of triples. A specific triple is highlighted with a red box:

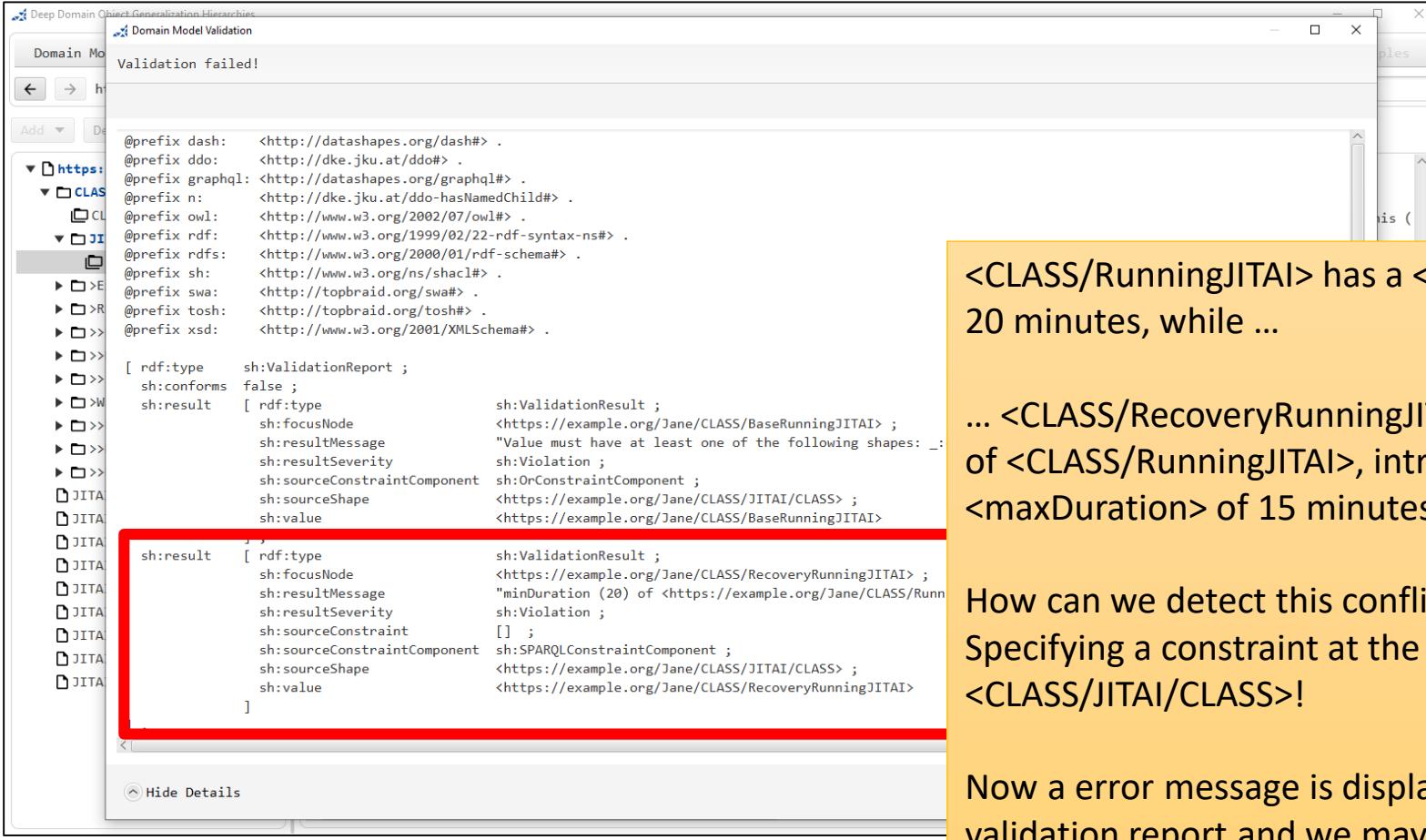
```
sh:sparql [ sh:message "minDuration ({?minDuration}) of {?minClass} ;  
sh:prefixes <PrefixDeclarations> ;  
sh:select "BASE <https://example.org/Jane/> SELECT $this .  
].
```

A yellow callout box contains the following text:

<CLASS/RunningJITAI> has a <minDuration> of 20 minutes, while ...
... <CLASS/RecoveryRunningJITAI>, a subclass of <CLASS/RunningJITAI>, introduces a <maxDuration> of 15 minutes.

How can we detect this conflict?
Specifying a constraint at the meta class <CLASS/JITAI/CLASS>!

Multi-Level Modeling Conflict Detection (4)



```
@prefix dash: <http://datashapes.org/dash#> .
@prefix ddo: <http://dke.jku.at/ddo#> .
@prefix graphql: <http://datashapes.org/graphql#> .
@prefix n: <http://dke.jku.at/ddo-hasNamedChild#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix swa: <http://topbraid.org/swat#> .
@prefix tosh: <http://topbraid.org/tosh#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

[ rdf:type sh:ValidationReport ;
  sh:conforms false ;
  sh:result [ rdf:type sh:ValidationResult ;
    sh:focusNode <https://example.org/Jane/CLASS/BaseRunningJITAI> ;
    sh:resultMessage "Value must have at least one of the following shapes: _: ;
    sh:Violation ;
    sh:sourceConstraintComponent sh:OrConstraintComponent ;
    sh:sourceShape <https://example.org/Jane/CLASS/JITAI/CLASS> ;
    sh:value <https://example.org/Jane/CLASS/BaseRunningJITAI> ] ]
  sh:result [ rdf:type sh:ValidationResult ;
    sh:focusNode <https://example.org/Jane/CLASS/RecoveryRunningJITAI> ;
    sh:resultMessage "minDuration (20) of <https://example.org/Jane/CLASS/RunningJITAI> is less than maxDuration (15) of <https://example.org/Jane/CLASS/BaseRunningJITAI>" ;
    sh:Violation ;
    sh:sourceConstraintComponent sh:SPARQLConstraintComponent ;
    sh:sourceShape <https://example.org/Jane/CLASS/JITAI/CLASS> ;
    sh:value <https://example.org/Jane/CLASS/RecoveryRunningJITAI> ] ]
```

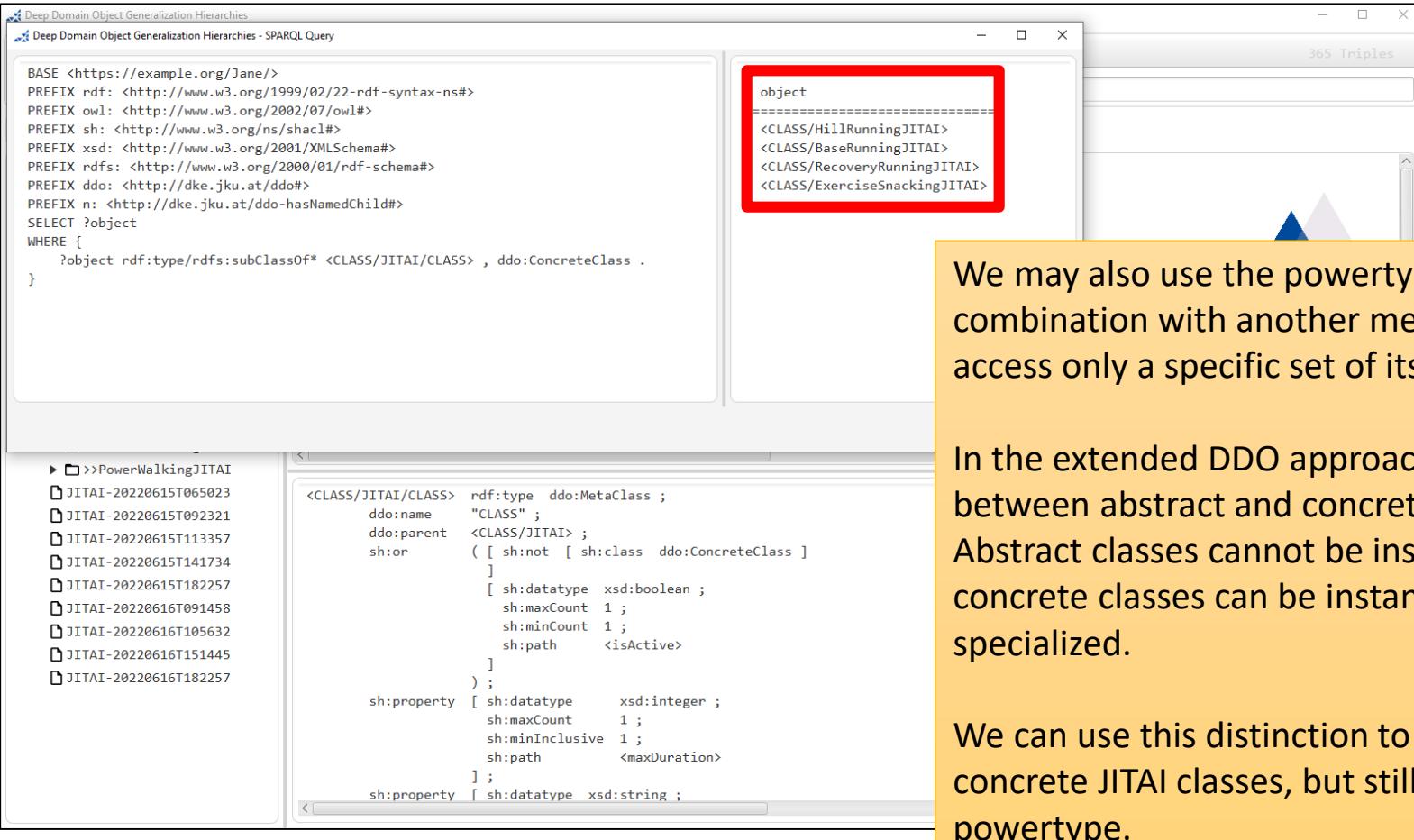
<CLASS/RunningJITAI> has a <minDuration> of 20 minutes, while ...

... <CLASS/RecoveryRunningJITAI>, a subclass of <CLASS/RunningJITAI>, introduces a <maxDuration> of 15 minutes.

How can we detect this conflict?
Specifying a constraint at the meta class <CLASS/JITAI/CLASS>!

Now a error message is displayed in the validation report and we may fix the problem.

Multi-Level Modeling Meta Class Combinations (1)



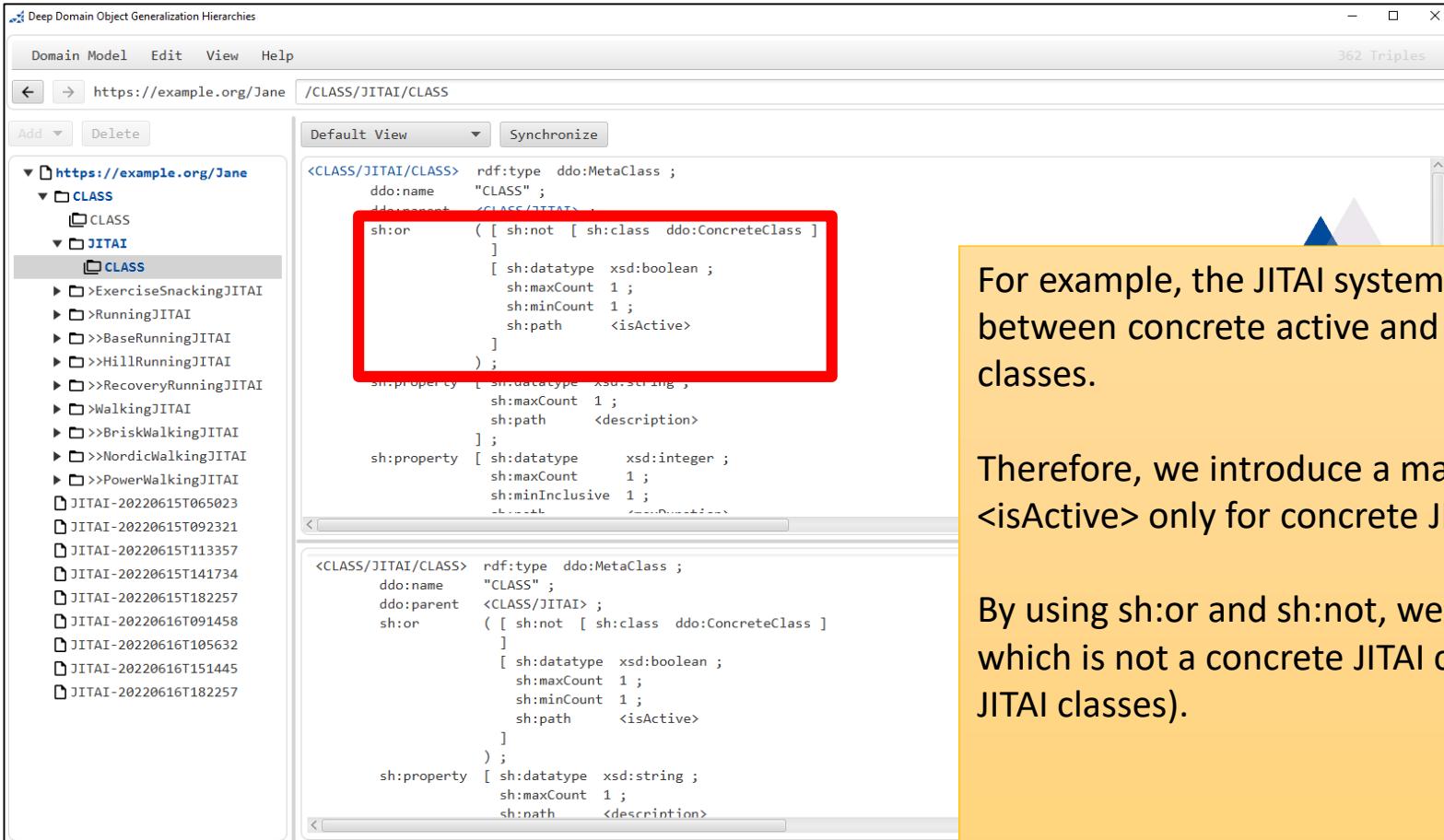
The screenshot shows a software interface for managing domain object generalization hierarchies. On the left, a SPARQL query window displays a query to find subclasses of a specific class. In the center, a tree view shows various JITAI classes under a parent node. On the right, a results window shows a list of concrete classes, with four specific ones highlighted in red: <CLASS/HillRunningJITAI>, <CLASS/BaseRunningJITAI>, <CLASS/RecoveryRunningJITAI>, and <CLASS/ExerciseSnackingJITAI>. A yellow callout box on the right contains the following text:

We may also use the powertype in combination with another mechanism to access only a specific set of its instances.

In the extended DDO approach, we distinguish between abstract and concrete classes. Abstract classes cannot be instantiated, whilst concrete classes can be instantiated but not specialized.

We can use this distinction to access only concrete JITAI classes, but still use the powertype.

Multi-Level Modeling Meta Class Combinations (2)



The screenshot shows a software interface for managing domain models. The left pane displays a tree view of classes under the URL `https://example.org/Jane`. The tree includes nodes for `CLASS`, `JITAI`, and various specific activity classes like `>ExerciseSnackingJITAI`, `>RunningJITAI`, etc. The right pane shows the RDF triples for the `JITAI` class. A red box highlights the following triple:

```
<CLASS/JITAI/CLASS> sh:or ( [ sh:not [ sh:class ddo:ConcreteClass ] ] [ sh:datatype xsd:boolean ; sh:maxCount 1 ; sh:minCount 1 ; sh:path <isActive> ] ) ;
```

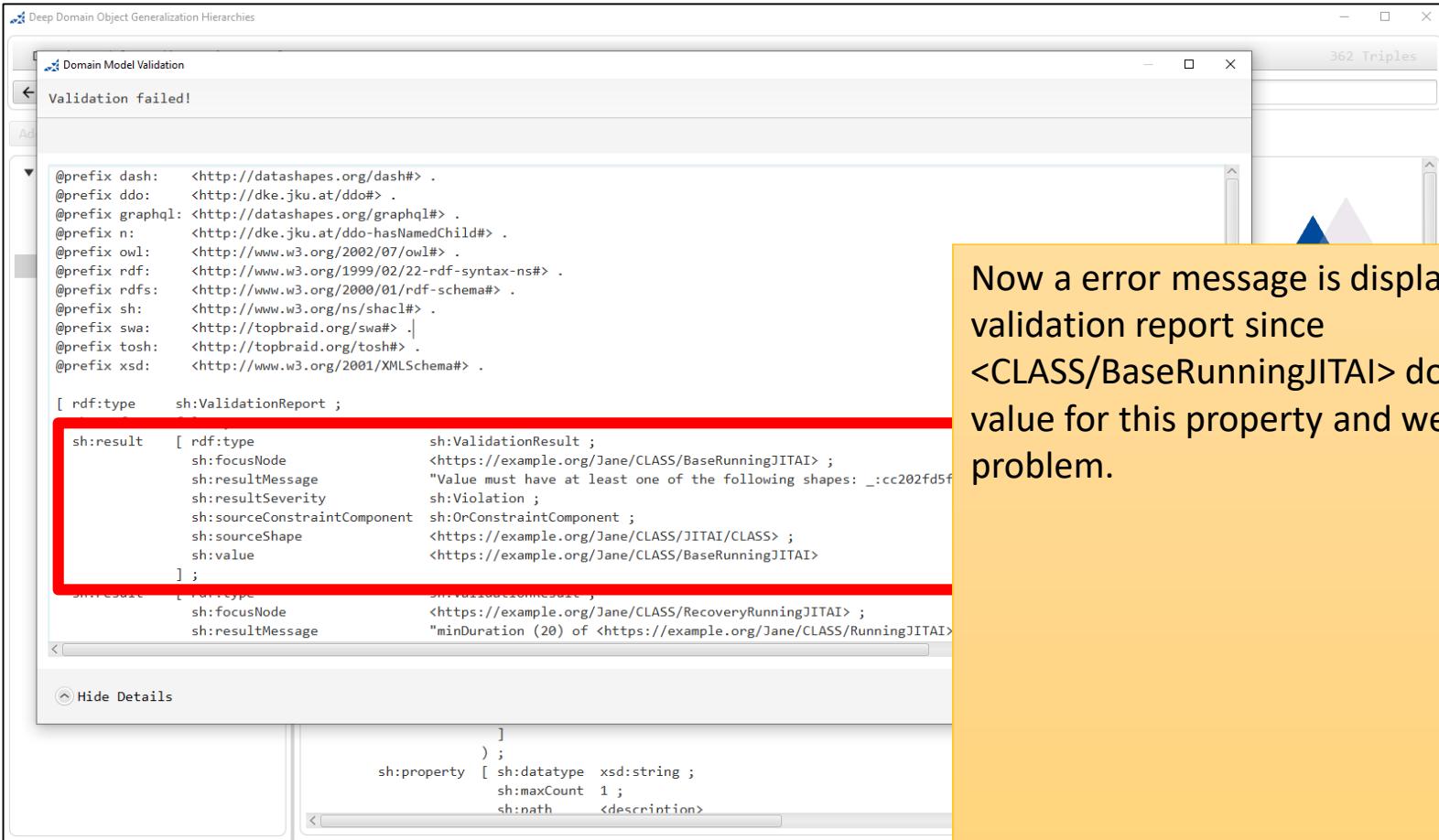
This triple uses `sh:or` to define a constraint that either excludes abstract classes or includes concrete classes where the `isActive` property is set to `true`.

For example, the JITAI system must distinguish between concrete active and inactive JITAI classes.

Therefore, we introduce a mandatory property `<isActive>` only for concrete JITAI classes.

By using `sh:or` and `sh:not`, we exclude any class which is not a concrete JITAI class (i.e., abstract JITAI classes).

Multi-Level Modeling Meta Class Combinations (3)



The screenshot shows the 'Deep Domain Object Generalization Hierarchies' interface. In the center, a validation report window is open with the title 'Domain Model Validation'. It displays the message 'Validation failed!' and a detailed list of errors. One specific error is highlighted with a red rectangle:

```
[ rdf:type sh:ValidationReport ;  
  sh:result [ rdf:type sh:ValidationResult ;  
    sh:focusNode <https://example.org/Jane/CLASS/BaseRunningJITAI> ;  
    sh:resultMessage "Value must have at least one of the following shapes: _:cc202fd5f" ;  
    sh:resultSeverity sh:Violation ;  
    sh:sourceConstraintComponent sh:OnConstraintComponent ;  
    sh:sourceShape <https://example.org/Jane/CLASS/JITAI/CLASS> ;  
    sh:value <https://example.org/Jane/CLASS/BaseRunningJITAI>  
  ] ;  
  sh:ValidationResult [ rdf:type sh:ValidationResult ;  
    sh:focusNode <https://example.org/Jane/CLASS/RecoveryRunningJITAI> ;  
    sh:resultMessage "minDuration (20) of <https://example.org/Jane/CLASS/RunningJITAI>" ]
```

A yellow callout box on the right side of the slide contains the following text:

Now a error message is displayed in the validation report since <CLASS/BaseRunningJITAI> does not provide a value for this property and we may fix the problem.

Multi-Level Modeling Meta Class Combinations (4)

The screenshot shows a software interface for managing domain models. The title bar reads "Deep Domain Object Generalization Hierarchies". The menu bar includes "Domain Model", "Edit", "View", and "Help". The address bar shows the URL "https://example.org/Jane /CLASS/RunningJITAI". A status bar at the top right indicates "371 Triples".

The left pane is a tree view of the domain model:

- https://example.org/Jane
- CLASS
 - JITAI
 - ExerciseSnackingJITAI
 - RunningJITAI
 - CLASS
 - >>BaseRunningJITAI
 - >>HillRunningJITAI
 - >>RecoveryRunningJITAI
 - >>WalkingJITAI
 - >>BriskWalkingJITAI
 - >>NordicWalkingJITAI
 - >>PowerWalkingJITAI
 - JITAI-20220615T065023
 - JITAI-20220615T092321
 - JITAI-20220615T113357
 - JITAI-20220615T141734
 - JITAI-20220615T182257
 - JITAI-20220616T091458
 - JITAI-20220616T105632
 - JITAI-20220616T151445
 - JITAI-20220616T182257

The right pane displays RDF triples for the selected class <CLASS/RunningJITAI>. One triple, <CLASS/RunningJITAI> rdf:type <CLASS/RunningJITAI/CLASS>, <byCategory>, is highlighted with a red box.

```
<CLASS/RunningJITAI#duration>
sh:maxInclusive 60 ;
sh:minInclusive 20 ;
sh:path <duration> .
```

```
<CLASS/RunningJITAI> rdf:type <CLASS/RunningJITAI/CLASS>, <byCategory>, ddo:AbstractClass ;
ddo:leafs <CLASS/RunningJITAI> ;
ddo:modeledClass <CLASS/RunningJITAI> ;
ddo:name "RunningJITAI" ;
ddo:parent <CLASS> ;
ddo:specializationOf n:JITAI ;
n:CLASS <CLASS/RunningJITAI/CLASS> ;
sh:property <CLASS/RunningJITAI#duration> ;
<avgDuration> 40.0 ;
<maxDuration> 60 ;
```

```
<CLASS/RunningJITAI> rdfs:subClassOf <CLASS/JITAI> ;
ddo:leafs <CLASS/RunningJITAI> ;
ddo:modeledClass <CLASS/RunningJITAI> ;
ddo:name "RunningJITAI" ;
ddo:parent <CLASS> ;
ddo:specializationOf n:JITAI ;
n:CLASS <CLASS/RunningJITAI/CLASS> ;
<maxDuration> 60 ;
<maxIntensity> 15 ;
<minDuration> 20 .
```

We may use further meta classes for clabject classification.

For example, we may differ between 3 JITAI categories with <CLASS/RunningJITAI> being one of them, therefore, in addition to its powertype, <CLASS/RunningJITAI> is instance of the meta class <byCategory>.

Multi-Level Modeling Meta Class Combinations (5)

The screenshot shows a SPARQL query interface with several panes. The top-left pane displays a SPARQL query:

```
BASE <https://example.org/Jane/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ddo: <http://dke.jku.at/ddo#>
PREFIX n: <http://dke.jku.at/ddo-hasNamedChild#>
SELECT ?object
WHERE {
    ?object rdf:type/rdfs:subClassOf* <CLASS/JITAI/CLASS> , <byCategory> .
}
```

The top-right pane shows the results of the query, which are three meta classes: `<CLASS/RunningJITAI>`, `<CLASS/ExerciseSnackingJITAI>`, and `<CLASS/WalkingJITAI>`. This result is highlighted with a red box.

The bottom-left pane shows a tree view of meta classes, with one node expanded to show its properties. The bottom-right pane shows the detailed description of the selected meta class, specifically `<CLASS/JITAI/CLASS>`.

Querying the instances of the powertype `<CLASS/JITAI/CLASS>` combined with the meta class `<byCategory>`, we receive all JITAI categories.

Task Introduction

- Before heading over to the task, take a moment, browse the model and especially consider rdf:type and rdfs:subClassOf relationships between elements.

Task

Subtask 1

- Define a SPARQL query that queries the direct and indirect instances of metaclass <CLASS/JITAI/CLASS> together with their descriptions and average duration.
- Expected result

class	description	avgDuration
<CLASS/JITAI>		16.89
<CLASS/RunningJITAI>		40.0
<CLASS/HillRunningJITAI>	"Vigorous running."	
<CLASS/BaseRunningJITAI>	"Moderate running."	40.0
<CLASS/RecoveryRunningJITAI>	"Light running"	
<CLASS/ExerciseSnackingJITAI>	"Do some physical exercise anywhere and anytime."	3.4
<CLASS/WalkingJITAI>		31.67
<CLASS/NordicWalkingJITAI>	"Walking with sticks."	
<CLASS/BriskWalkingJITAI>	"Moderate walking."	31.67
<CLASS/PowerWalkingJITAI>	"Vigorous walking."	

Task

Subtask 2

- Define a SPARQL query that counts for each JITAI class the number of individual JITAI occurrences (considering direct and indirect instances and also considering JITAI classes that have no instances).
- Use the appropriate meta class in your query.
- Expected result

class	numberOfJITAI
<CLASS/NordicWalkingJITAI>	0
<CLASS/HillRunningJITAI>	0
<CLASS/BriskWalkingJITAI>	3
<CLASS/JITAI>	9
<CLASS/PowerWalkingJITAI>	0
<CLASS/BaseRunningJITAI>	1
<CLASS/RecoveryRunningJITAI>	0
<CLASS/WalkingJITAI>	3
<CLASS/RunningJITAI>	1
<CLASS/ExerciseSnackingJITAI>	5

Task

Subtask 3

- Define a SPARQL query that selects WalkingJITAI classes (use the appropriate meta class) that are also instances of meta class <byIntensity>.
- Expected result

class
<CLASS/NordicWalkingJITAI>
<CLASS/BriskWalkingJITAI>
<CLASS/PowerWalkingJITAI>

Task

Subtask 4

- Define a SPARQL query that identifies for each JITAI class the limitations on <duration> made by SHACL property shapes using sh:minInclusive and sh:maxInclusive.
- Expected result

class	minInclusive	maxInclusive
<CLASS/JITAI>	1	
<CLASS/RunningJITAI>	20	60
<CLASS/HillRunningJITAI>		
<CLASS/BaseRunningJITAI>		
<CLASS/RecoveryRunningJITAI>		15
<CLASS/ExerciseSnackingJITAI>		5
<CLASS/WalkingJITAI>	10	60
<CLASS/NordicWalkingJITAI>		
<CLASS/BriskWalkingJITAI>		
<CLASS/PowerWalkingJITAI>		

Task

Subtask 5

- We have introduced various concepts of MLM using <duration>. Now we focus on regularity properties for <intensity>.
- Add SHACL property shapes for <minIntensity> and for <maxIntensity> at the appropriate powertype, specifying that <minIntensity> and <maxIntensity> have multiplicity 0..1 and datatype integer.
- You can evaluate these constraints by manually violating them.

Task

Subtask 6

- Add SHACL rules for transforming a JITAI class' value for <minIntensity> and <maxIntensity> into property shapes with sh:minInclusive and sh:maxInclusive.
- You may use the rules that transform <minDuration> and <maxDuration> into property shapes as a template.
- Expected result, e.g., for <CLASS/RunningJITAI>



The screenshot shows a SPARQL editor interface with two tabs: "Default View" and "Synchronize". The "Default View" tab is selected. The code area contains the following SPARQL query and its results:

```
Default View Synchronize

<CLASS/RunningJITAI#intensity>
    sh:maxInclusive 15 ;
    sh:path      <intensity> .
```

The first part of the code is highlighted with a red rectangle. The rest of the code is as follows:

```
<CLASS/RunningJITAI#duration>
    sh:maxInclusive 60 ;
    sh:minInclusive 20 ;
    sh:path      <duration> .
```

```
<CLASS/RunningJITAI> rdf:type <CLASS/RunningJITAI/CLASS> , <byCategory> , ddo:AbstractClass , ddo:ModeledClass , ddo:OccurrenceClass ;
    rdfs:subClassOf      <CLASS/JITAI> ;
    ddo:leafs            ddo:AbstractClass ;
    ddo:modeledClass    <CLASS/RunningJITAI> ;
    ddo:name             "RunningJITAI" ;
    ddo:parent           <CLASS> ;
    ddo:specializationOf n:JITAI ;
    n:CLASS              <CLASS/RunningJITAI/CLASS> ;
    sh:property          <CLASS/RunningJITAI#duration> , <CLASS/RunningJITAI#intensity> ;
    <avgDuration>        40.0 ;
    <maxDuration>         60 ;
    <maxIntensity>        15 ;
    <minDuration>         20 .
```

Task

Subtask 7

- Compute the average intensity <avgIntensity> as resultant property for each JITAI class by defining a SHACL rule at the appropriate meta class.
- You may use the rule that computes <avgDuration> as a template.
- Expected result, e.g., for <CLASS/ExerciseSnackingJITAI>



The screenshot shows a RDF browser interface with the following details:

- Toolbar: Default View, Synchronize
- Properties listed:
 - <CLASS/ExerciseSnackingJITAI#duration>
sh:maxInclusive 5 ;
sh:path <duration> .
 - <CLASS/ExerciseSnackingJITAI#intensity>
sh:maxInclusive 13 ;
sh:path <intensity> .
 - <CLASS/ExerciseSnackingJITAI>
rdf:type <byCategory> , ddo:ModeledClass , <CLASS/ExerciseSnackingJITAI/CLASS> , ddo:OccurrenceClass , ddo:ConcreteClass ;
rdfs:subClassOf <CLASS/JITAI> ;
ddo:leafs ddo:ConcreteClass ;
ddo:modeledClass <CLASS/ExerciseSnackingJITAI> ;
ddo:name "ExerciseSnackingJITAI" ;
ddo:parent <CLASS> ;
ddo:specializationOf n:JITAI ;
n:CLASS <CLASS/ExerciseSnackingJITAI/CLASS> ;
sh:property <CLASS/ExerciseSnackingJITAI#duration> , <CLASS/ExerciseSnackingJITAI#intensity> ;
sh:result > 10.4 ;
<avgIntensity> 10.4 ;
sh:description "Do some physical exercise anywhere and anytime." ;
sh:isActive true ;
sh:maxDuration 5 ;
sh:maxIntensity 13 .

Task

Subtask 8

- Detect conflicts between <minIntensity> and <maxIntensity> within the class hierarchy using SHACL SPARQL-based constraints at the appropriate meta class.
- You may use the shape that detects conflicts between <minDuration> and <maxDuration> as a template.
- Expected result (additional validation result, Domain Model > Validate ...)



Evaluation

- There is a voluntary and anonymized survey, you are kindly invited to participate after completing the tutorial and working through the task.
- Even if you could not successfully solve all subtasks, any feedback is highly important and appreciated.
- Here is the link (< 5min):
 - <https://survey.jku.at/jku2018/index.php/365873?lang=en>
- **Thank you!**

Contact

- If you have questions about the survey, encounter bugs or have other technical problems, you can contact
 - sebastian.gruber@dhp.lbg.ac.at
 - sebastian.gruber@salzburgresearch.at

References

- Almeida, J. P. A., Carvalho, V. A., Fonseca, C. M., & Guizzardi, G. (2021, October). A Note on Properties in Multi-Level Modeling. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 497-501). IEEE.
- Carvalho, V. A., & Almeida, J. P. A. (2018). Toward a well-founded theory for multi-level conceptual modeling. *Software & Systems Modeling*, 17(1), 205-231.
- Gruber, S., Neumayr, B., Schrefl, M., & Niebauer, J. (2021, October). Towards Multi-level Modeling of Just-in-Time Adaptive Interventions (JITAIls) in Mobile Health. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 541-545). IEEE.
- Hardeman, W., Houghton, J., Lane, K., Jones, A., & Naughton, F. (2019). A systematic review of just-in-time adaptive interventions (JITAIls) to promote physical activity. *International Journal of Behavioral Nutrition and Physical Activity*, 16(1), 1-21.
- Nahum-Shani, I., Smith, S. N., Spring, B. J., Collins, L. M., Witkiewitz, K., Tewari, A., & Murphy, S. A. (2018). Just-in-time adaptive interventions (JITAIls) in mobile health: key components and design principles for ongoing health behavior support. *Annals of Behavioral Medicine*, 52(6), 446-462.
- Neumayr, B., & Schrefl, M. (2022). Domain object hierarchies inducing multi-level models. *Software and Systems Modeling*, 21(2), 587-621.