# Database Frameworks Exam Preparation 2

## Mass Defect

The year is 2306. The united galactic civilizations (UGC), are steadily progressing through the process of evolution. Almost a century ago, an unexpected phenomenon was introduced to the young human civilization and to other alien civilizations. People from the different civilizations were being randomly teleported across the galaxy, without any pattern or logic behind. The situation was too awkward to be explained, so the civilizations decided to form a unity to deal with this phenomenon. That is how the UGC was formed. The UGC established a Database which would keep track of the random teleports and their victims. The human civilization describes these anomalies as the worst discovery in their history. The civilizations of the galaxy call it… Mass Defect.

Use the provided skeleton to implement the needed functionality. You are not allowed to change anything in the project structure or interfaces.

## Data Model Definition

You have been tasked to create a **code first data model** in **Spring Data** for the Mass Defect Database. You will also need to write several data-driven mini applications in Java for importing, querying and exporting data from the database. For some reason, the UGC has decided that **JSON** and **XML** are the default data formats, so your imports and exports will be performed with those formats.

The database you need to implement has 5 main entities:

##### Solar Systems

* Have **Id** and **Name**

##### Stars

* Have **Id** and **Name**
* The Stars **must** have a **Solar System**

##### Planets

* Have **Id** and **Name**
* The Planets **must** have a **Sun**
* The Planets **must** have a **Solar System**

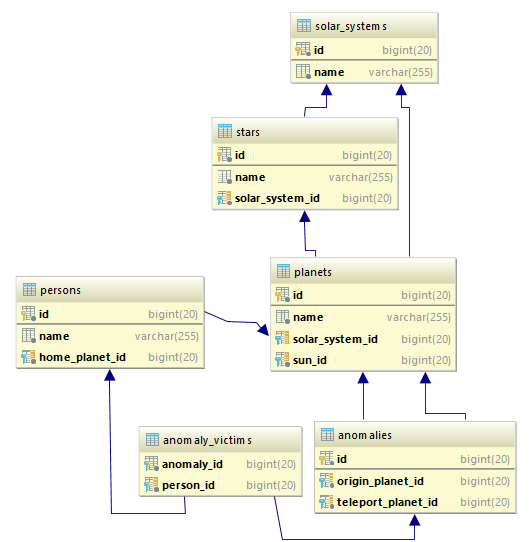
##### Persons

* Have **Id** and **Name**
* The Persons **must** have a **Home** **Planet**

##### Anomalies

* Has **Id**
* The Anomalies **must** have an **Origin Planet** and **Teleport Planet**
* The Anomalies can have **many Persons** as victims, and one **Person** can be a victim to **many** **Anomalies**

Here is a E/R diagram of the database to make it easier for you:



Create a **Spring application** to implement the **data model** for the database exactly as specified above.

## Importing Data

Now that you have successfully created your database, you will need to import the data that has been recorded on paper all this time because the UGC didn’t have a database developer such as yourself.

NOTE: Implement Services and Repositories for your entities. The **Services’** main job is to assure the **validity of the data** they are going to **persist**.

**NOTE**: Notice that the FileParser object uses a pre-read JSON content and maps it to the appropriate dto implementation. Read every file with your FileIO implementation and pass it to the parsers.

**NOTE**: Every printing on the console is done by the ConsoleIO implementation.

### Importing Data from JSON

Implement your **DTOs** based on your **data model** and seed the data from the **JSON** files you received, into the database.

Make sure all fields have been entered, otherwise the import **entity** data **should not be considered valid**.

If you import correctly an entity of Solar System, Star, Planet or Person, you should print a message on the console, saying: “Successfully imported {entity} {entityName}.”.

If you successfully import data about Anomalies, you should print a message on the console saying: “Successfully imported anomaly.”.

Successful imports in the anomaly-victims, should not hold notification.

### Example:

#### Input

|  |
| --- |
| **solar-systems.json** |
| [  { "name" : "Kepler-Epsilon" },  { "name" : "Alpha-Nebula" },  { "name" : "Beta-Cluster" },  { "name" : "Voyager-Sentry" },  { "name" : "Zeta-Cluster" },  ] |

#### Output

|  |
| --- |
| Successfully imported Solar System Kepler-Epsilon.  Successfully imported Solar System Alpha-Nebula.  Successfully imported Solar System Beta-Cluster.  Successfully imported Solar System Voyager-Sentry.  Successfully imported Solar System Zeta-Cluster. |

In case one of the fields is missing In the import data, print a message:  
“Error: Invalid data.” … and ignore that **entity**.

### Example:

#### Input

|  |
| --- |
| **stars.json** |
| [  { "name" : "Visilus", "solarSystem" : "Kepler-Epsilon" },  { "name" : "Neb-X10", "solarSystem" : "Alpha-Nebula" },  { "name" : "Scarlet-Sentry" },  { "name" : "Indigo-Sentry", "solarSystem" : "Voyager-Sentry" },  { "name" : "Neb-X11", "solarSystem" : "Alpha-Nebula" },  ] |

#### Output

|  |
| --- |
| Successfully imported Star Visilus.  Successfully imported Star Neb-X10.  Error: Invalid data.  Successfully imported Star Indigo-Sentry.  Successfully imported Star Neb-X11. |

### Importing Data from XML

New reports have come about several new anomalies. The format of the reports, however, this time is **XML**. Nevertheless, you need to put them in the database.

The given input and expected output is as follows…

### Example:

#### Input

|  |
| --- |
| **new-anomalies.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <anomalies>  <anomaly origin-planet="Kepler-3" teleport-planet="Voyager-10">  <victims>  <victim name="Eifell Sync" />  </victims>  </anomaly>  <anomaly origin-planet="Kepler-1">  <victims>  <victim name="Eifell Sync" />  </victims>  </anomaly>  <anomaly origin-planet="Voyager-10" teleport-planet="Voyager-11">  <victims>  <victim name="Eifell Sync" />  </victims>  </anomaly>  ...  </anomalies> |

#### Output

|  |
| --- |
| Successfully imported data.  Error: Invalid data.  Successfully imported data.  ... |

If any field data is **missing**, you should print an **error message**, and **ignore** that **input data**. If a victim’s data is **invalid**, not only the victim, but **the whole anomaly input** should be **ignored**.

## Data Exporting

The UGC has requested applications for **exporting** **data** from the database, so that statistics can be made and presented publicly. You know the drill, you will have to build applications for exporting data in both **JSON** and **XML** formats. People need to be informed of the events that are happening around the Galaxy. There are several **query** **tasks** you need to do. Create a spring data application for those tasks.

For the data exporting, you’ll need explicit DTOs, so it is recommended that you build ones for the entities that are **exported**. The DTOs are simple – they just have to hold the **fields** and **properties**, that are **required** for the **output**.

You will have to extend the parsers though. Add one more method to them, so that they can write files too:

Here are the several tasks you need to do.

### Planets which have no people teleported FROM them

Extract all planets from the database, which are not an **origin planet** to any Mass Defect anomaly. Extract the planets’ **names**.

|  |
| --- |
| **planets.json** |
| [  {  "name": "Alpha-N45"  }  ] |

### Anomaly which affected the most people

Extract the anomaly which has affected the most victims. Extract the anomaly’s **id**, **origin planet name**, **teleport planet name**, and **number of victims**.

|  |
| --- |
| **anomaly.json** |
| [  {  "id": 14,  "originPlanet": {  "name": "Kepler-1"  },  "teleportPlanet": {  "name": "Alpha-N20"  },  "victimsCount": 5  }  ] |

### Exporting to XML

You also need to export data about the anomalies in **XML** format, for the **3-rd Galaxy Solar Systems**, which still haven’t evaluated enough to read **JSON** format.

The UGC agent has given us pretty much any information so far, and he is getting tired, maybe we should not ask for any more help from him. The DTOs follow almost the same logic anyways. Use the **service** to **export the data**, and the parser to **extract xml output**.

### All anomalies ordered

Extract **all anomalies** ordered by their id in the given xml format:

|  |
| --- |
| **anomalies.xml** |
| <?xml version="1.0" encoding="UTF-8" standalone="yes"?>  <anomalies>  <anomaly origin-homePlanet="Alpha-N20" teleport-homePlanet="Kepler-3"/>  <anomaly origin-homePlanet="Kepler-3" teleport-homePlanet="Voyager-10"/>  <anomaly origin-homePlanet="Kepler-6" teleport-homePlanet="Alpha-N20"/>  <anomaly origin-homePlanet="Indigo-3" teleport-homePlanet="Kepler-1"/>  <anomaly origin-homePlanet="Voyager-11" teleport-homePlanet="Indigo-3"/>  <anomaly origin-homePlanet="Voyager-10" teleport-homePlanet="Voyager-11"/>  <anomaly origin-homePlanet="Alpha-N20" teleport-homePlanet="Kepler-3"/>  <anomaly origin-homePlanet="Kepler-3" teleport-homePlanet="Kepler-6"/>  <anomaly origin-homePlanet="Indigo-3" teleport-homePlanet="Voyager-10"/>  <anomaly origin-homePlanet="Indigo-3" teleport-homePlanet="Alpha-N20"/>  <anomaly origin-homePlanet="Indigo-3" teleport-homePlanet="Voyager-10"/>  <anomaly origin-homePlanet="Voyager-11" teleport-homePlanet="Alpha-N20"/>  <anomaly origin-homePlanet="Alpha-N20" teleport-homePlanet="Indigo-3"/>  <anomaly origin-homePlanet="Alpha-N20" teleport-homePlanet="Indigo-3"/>  <anomaly origin-homePlanet="Kepler-6" teleport-homePlanet="Kepler-1"/>  <anomaly origin-homePlanet="Kepler-3" teleport-homePlanet="Kepler-1"/>  <anomaly origin-homePlanet="Kepler-3" teleport-homePlanet="Kepler-6"/>  <anomaly origin-homePlanet="Alpha-N20" teleport-homePlanet="Kepler-1"/>  <anomaly origin-homePlanet="Voyager-10" teleport-homePlanet="Kepler-1"/>  <anomaly origin-homePlanet="Kepler-1" teleport-homePlanet="Alpha-N20"/>  </anomalies> |