# Databases Frameworks Exam – Hiberspring Inc.

**Hiberspring Incorporation** is a new non-profit organization which has products of almost any type.   
By some rare circumstances, though, it became the monopoly of the garbage bag business part of the market. Anyways … Since they are a non-profit organization, they had to find a non-profit charitable database specialist for their database application. Guess what, you fit the description perfectly.

### Overview

The Hiberspring Incorporation’s database holds 5 entities. Employees, EmployeeCards, Branches, Products and Towns. You will have to implement them in your application, so that they can be generated on the database. Hiberspring would also like for the application to support functionality for **importing** and **exporting** datain JSON and XML formats, so you’ll have to implement that too.

The technologies and ways which the Incorporation prefers that you use, is a **Code-First** **approach** application with **Hibernate** and **Spring Data**.

### Data Models

You can see here what properties each model has:

#### Employee

* Id – an **integer**.
* First Name – a **string**.
* Last Name – a **string**.
* Position – a **string**.
* Card – an EmployeeCard, could be any EmployeeCard. Must be **UNIQUE though**.
* Branch – a Branch, could be any Branch.

#### EmployeeCard

* Id – an **integer**.
* Number – a **string**. Should be **UNIQUE**.

#### Branch

* Id – an **integer**.
* Name – a **string**.
* Town – a Town, could be any Town.

#### Product

* Id – an **integer**.
* Name – a **string**.
* Clients – an **integer**.
* Branch – a Branch, could be any Branch.

#### Town

* Id – an **integer**
* Name – a **string**.
* Population – an **integer**.

**All data** is **REQUIRED**, unless it is explicitly said that it **null** is **allowed**.

### Importing Data

So here comes the **Importing** of **data** and the **populating** of the **database**. You have to **import** data from **JSON** and **XML** files.

Implement the needed **DTOs** for the imports.

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

You will also have to print a simple message indicating if the data has been imported successfully or there was an error.

In case of **SUCCESS** the message format is “Successfully imported {entityClassName} {entityField}.”. The entityField may vary, from card number to employee full name, to just name.

In case of **ERROR** you always print “Error: Invalid data.”.

#### Importing from JSON format

#### Towns

##### Input

|  |
| --- |
| **towns.json** |
| [  { "name" : "Sofia", "population" : 2001050 },  { "name" : "Kairo", "population" : 5432000 },  { "name" : "New York", "population" : 11563790 },  { "name" : "Tokyo", "population" : 27634593 },  { "name" : "Moscow", "population" : 4523120 },  { "name" : "Rome", "population" : 3021333 },  { "name" : "Madrid", "population" : 7403213 },  { "name" : "Paris", "population" : 8900043 },  { "name" : "Zanzibar" },  { "name" : "Rio de Janeiro", "population" : 6345231 },  . . .  ] |

##### Output

|  |
| --- |
| Successfully imported Town Sofia.  Successfully imported Town Kairo.  Successfully imported Town New York.  Successfully imported Town Tokyo.  Successfully imported Town Moscow.  Successfully imported Town Rome.  Successfully imported Town Madrid.  Successfully imported Town Paris.  Error: Invalid data.  Successfully imported Town Rio de Janeiro.  . . . |

#### Branches

##### Input

|  |
| --- |
| **branches.json** |
| [  { "name" : "Sofia Western Branch", "town" : "Sofia" },  { "name" : "Tokyo Main Branch", "town" : "Tokyo" },  { "name" : "Headquarters", "town" : "Sofia" },  { "town" : "New York" },  { "name" : "Kairo Central Branch", "town" : "Kairo" },  { "name" : "Tokyo Underground Branch", "town" : "Tokyo" },  { "name" : "USA Main Branch", "town" : "Washington DC" },  { "name" : "Sofia Eastern Branch", "town" : "Sofia" },  { "name" : "Central Branch of New York" },  { "name" : "Central Park Branch", "town" : "New York" },  . . .  ] |

##### Output

|  |
| --- |
| Successfully imported Branch Sofia Western Branch.  Successfully imported Branch Tokyo Main Branch.  Successfully imported Branch Headquarters.  Error: Invalid data.  Successfully imported Branch Kairo Central Branch.  Successfully imported Branch Tokyo Underground Branch.  Successfully imported Branch USA Main Branch.  Successfully imported Branch Sofia Eastern Branch.  Error: Invalid data.  Successfully imported Branch Central Park Branch.  . . . |

#### EmployeeCards

##### Input

|  |
| --- |
| **employee\_cards.json** |
| [  { "number" : "zi4n5-y41Pq-ugz5v-3vrNH-Dv21y" },  { "number" : "UAIP0-0UVao-3axBt-vWF8c-45paZ" },  { "number" : "65RrK-NRzLZ-pJLZN-Chp3q-tovmA" },  { "number" : "DXKwE-pprkA-dLT9g-bGnbp-1304U" },  { "number" : "3mQuf-dGsVC-v5RhD-esuzu-0XcXp" },  . . .  ] |

##### Output

|  |
| --- |
| Successfully imported Employee Card zi4n5-y41Pq-ugz5v-3vrNH-Dv21y.  Successfully imported Employee Card UAIP0-0UVao-3axBt-vWF8c-45paZ.  Successfully imported Employee Card 65RrK-NRzLZ-pJLZN-Chp3q-tovmA.  Successfully imported Employee Card DXKwE-pprkA-dLT9g-bGnbp-1304U.  Successfully imported Employee Card 3mQuf-dGsVC-v5RhD-esuzu-0XcXp.  . . . |

#### Importing from XML format

The other **2 tables** must be populated with data in **XML** format.

#### Products

##### Input

|  |
| --- |
| **products.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <products>  <product name="Hydrogen Car Engine" clients="20000">  <branch>Tokyo Main Branch</branch>  </product>  <product name="McDonalds Burger" clients="5001023">  <branch>Central Park Branch</branch>  </product>  <product name="Garbage Bag" clients="102849">  <branch>Headquarters</branch>  </product>  ...  </products> |

##### Output

|  |
| --- |
| Successfully imported Product Hydrogen Car Engine.  Successfully imported Product McDonalds Burger.  Successfully imported Product Garbage Bag.  . . . |

#### Employees

##### Input

|  |
| --- |
| **employees.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <employees>  <employee first-name="John" last-name="Winchester" position="Security Manager">  <card>zi4n5-y41Pq-ugz5v-3vrNH-Dv21y</card>  <branch>USA Main Branch</branch>  </employee>  <employee first-name="Leeroy" last-name="Gips" position="Security Manager">  <card>3mQuf-dGsVC-v5RhD-esuzu-0XcXp</card>  <branch>Kairo Central Branch</branch>  </employee>  <employee first-name="Rick" last-name="Sanchez" position="Head Scientist">  <card>65RrK-NRzLZ-pJLZN-Chp3q-tovmA</card>  <branch>Headquarters</branch>  </employee>  <employee first-name="Tony" last-name="Dolfin" position="Cleaner">  <card>a45xz-dkgw1-zadv1-aXXXc-491Az</card>  </employee>  . . .  </employees> |

##### Output

|  |
| --- |
| Successfully imported Employee John Winchester.  Successfully imported Employee Leeroy Gips.  Successfully imported Employee Rick Sanchez.  Error: Invalid data.  . . . |

### Exporting Data

So here comes the **Exporting** of **data** from the **database**. You have to **export** data from **JSON** and **XML** files.

#### Exporting to JSON format

First you will need to **write functionality** for exporting **filtered data** in **JSON** format.

#### Free Cards

Export all cards which remain **unused** by **ANY employee**. Extract everything and **order** it by **id** in **ascending order**.

##### Output

|  |
| --- |
| **free\_cards.json** |
| [  { "number" : "ukBy1-rliJm-tMvvk-jRLtI-Z4byL" },  { "number" : "YZkdP-LuRyn-pM6S2-gZwKP-0OK3o" },  { "number" : "99rXG-1Ynwp-ahF4J-btmQ6-wg6d3" },  { "number" : "j3hwx-Z4TH0-V5oXu-XGczx-P7h0T" },  { "number" : "1wBUO-DZHZH-gjg9J-4VXr4-rLgtp" },  . . .  ] |

#### Productive Employees

It gets better and better. Now this one is a bit more complicated. **Extract all** **Employees**, who are **working** in a **Branch**, which has **at least one product**. Extract the **Employee’s full name** (**first name** + **‘ ’** + **last name**), the **Employee’s position**, and the **Employee’s Card’s Number**.

Order the data by **full name** in **alphabetical order**, and then by **length** of **position** in **descending order**.

##### Output

|  |
| --- |
| **productive-employees.json** |
| [  {  "full\_name" : "Alex Mercer",  "position " : "Head Security",  "number" : "Vy2un-LBvJU-31FsV-GbD6B-WUkBT",  },  {  "full\_name" : "Jake Drinkwater",  "position " : "Head Biologist",  "number" : "M3Y0n-A10Ev-lCk8M-8BtnZ-25Rxv",  },  . . .  ] |

#### Exporting to XML format

Alright, we finished with the JSON, now let’s get it on with the XML.

#### Towns

The first task is about the Towns. Easy table, two properties … But not so fast. You are tasked to extract all towns, selecting the **town’s name**, the **town’s population**, and the **town’s clients**. The **town’s clients** are the **sum of all clients** from the **products**, in the **branches** at those **towns**.

**Order** the **data** in **descending order** by **town clients**.

##### Output

|  |
| --- |
| **towns.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <towns>  <town name="Pekin" population="23023020" town\_clients="20000000">  <town name="Mexico City" population="12041284" town\_clients="5502323">  <town name="New York" population="11563790" town\_clients="5001033">  <town name="Sofia" population="2001050" town\_clients="102850">  . . .  <town name="Washington DC" population="3204506" town\_clients="0">  </towns> |

#### Top Branches

For this task, you need to **extract** the **branch**, and order them by **sum** of **clients** of their **products**, in **descending order**. **For each branch** extract the **branch’s name**, its **town’s name**, its **total clients**.

##### Output

|  |
| --- |
| **top-branches.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <branches>  <branch name="Pekin Funny Branch" town="Pekin" total\_clients="20000000">  <branch name="Mexico Main Branch" town="Mexico City" total\_clients="5502323">  <branch name="Central Park Branch" town="New York" total\_clients="5001023">  <branch name="Headquarters" town="Sofia" total\_clients="102850">  . . .  <branch name="Kuala Lumpur Potato Branch" town="Kuala Lumpur" total\_clients="0">  </branches> |

Well, that’s all folks. One more thing! Hiberspring Incorporation would like from you to use the **best architecture** **possible**. Use all the knowledge you’ve gathered to create the most suitable **High Quality Structure** for the application.