

# Java Fundamental Project

IAMCORE PROJECT

ABHISHEK NARKAR

## Table of Contents

<b>Subject description .....</b>	<b>3</b>
<b>Subject analysis .....</b>	<b>3</b>
Major features .....	3
Application Feasibility .....	3
Data description .....	3
Expected results .....	4
Scope of the application .....	4
<b>Conception .....</b>	<b>4</b>
Data structures .....	4
Identities .....	4
Users .....	4
Global application flow .....	4
Authentication and Menu .....	5
Option 1: Create an identity .....	6
Option 2: Modify an identity .....	6
Option 3: Delete an Identity .....	7
Option 4: Search identity .....	8
Option 5: Quit .....	8
Global schema and major features schema .....	8
<b>Configuration instructions .....</b>	<b>9</b>
Prerequisites .....	9
Schema creation .....	10
Database Configuration .....	10

## Subject description

The IAM Project is the final project presented for the Fundamental semester students of the Java Fundamentals class.

This project is the implementation of an Identity Management Software. The project was partially developed during the semester's class and has been completed during the last week during the end of the semester.

The implementation of this project has been done by command line only, no UI is provided.

## Subject analysis

### Major features

The IAMCORE Project has 4 major features.

- Authentication: A User should be authenticated before been able to work and manage the Identities.
- Manage Identities: the user should be able to
  - Create,
  - Update,
  - Delete
  - Search Identities from the application.

To do this, the Identities and the users are stored as application data. We use Derby as the Database engine for managing the entities.

### Application Feasibility

With the knowledge obtained from the Java class during the semester and taking into account the development done in class we can easily see that this application is completely feasible under the Java platform.

Also, we can clearly see the utility of building such an application that allow us to clearly see the process of managing entities.

### Data description

In this case we have to represent two different types of entities: Identities and Users.

The requirements do not specify anything regarding the managing of users, only the Identities, for simplicity no implementation has been made to manage users, the users are stored in a XML file mainly because it is not expected for them to be handled too much. If needed this file can be edited manually.

On the other hand, the identities are stored in a relational database, are represented by one table only and a Data Access Context has been created to help with their management.

### Expected results

The user is expected to interact with the application using the command line input/output. The three basic management functions should be completely functional so the software to be useful and compliant with specifications.

### Scope of the application

As stated above, the application is limited to only manage Identities and not to manage the Users. This is of course one of the first steps to be improved in a next revision.

Also, the Identities are only going to be managed on a database, there is no current way to select a file as storage mode for Identities, so the user has to have installed a Database manager software.

## Conception

### Data structures

#### Identities

Identities are basically the entities that are going to be managed, within the application we should be able to create, update and delete these. All identities are going to be stored in a Derby database.

Per the requirements an identity can have:

- UId: User Id, a unique id auto-generated and auto-incremented as a Primary Key in the Identity table.
- Display Name: represented as a string.
- Email: represented as a string also.

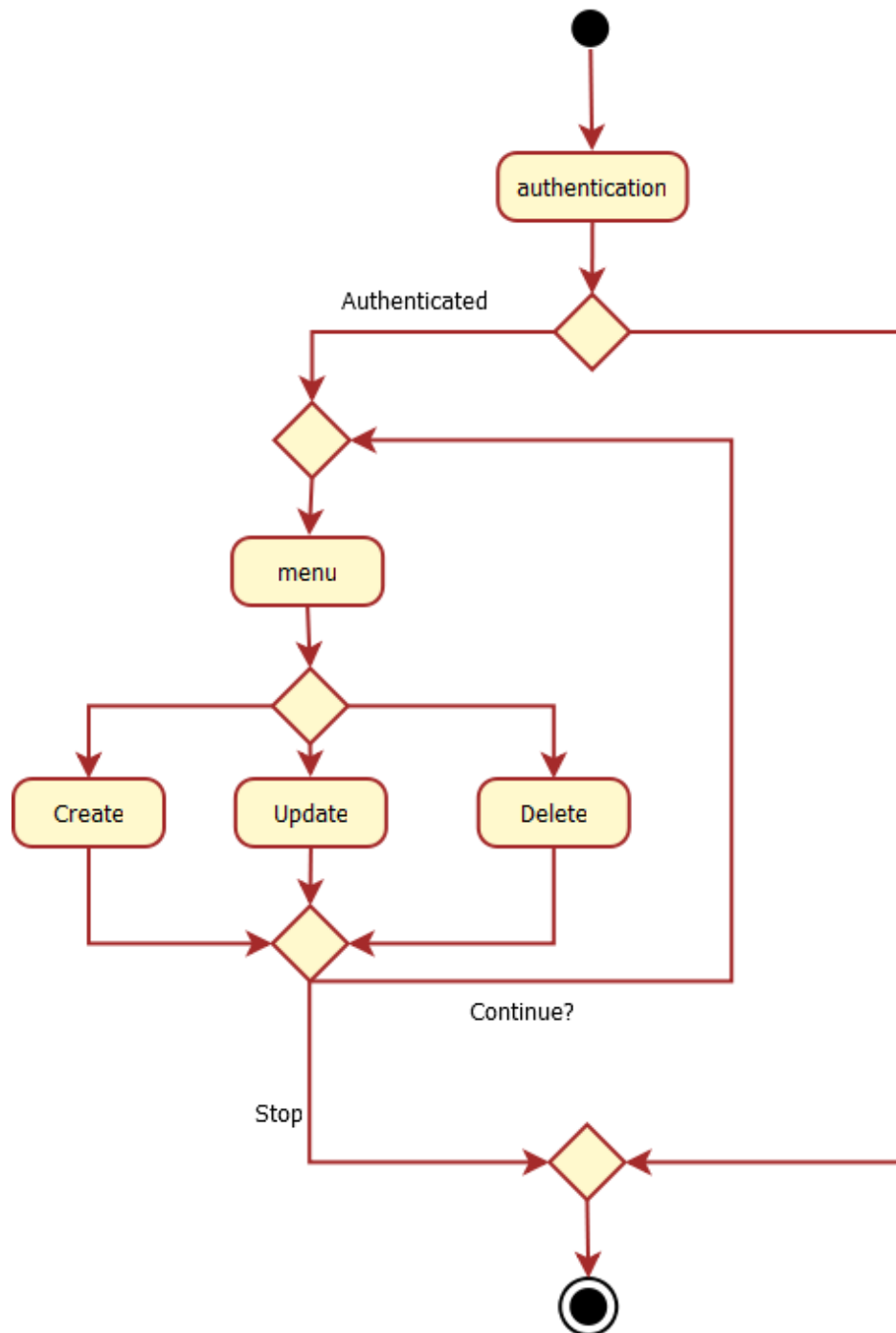
#### Users

Users are not supposed to be managed by the application, for these reason a decision was made that user entities will be stored in a text file.

A user will have a name and a password; both are stored as string files.

### Global application flow

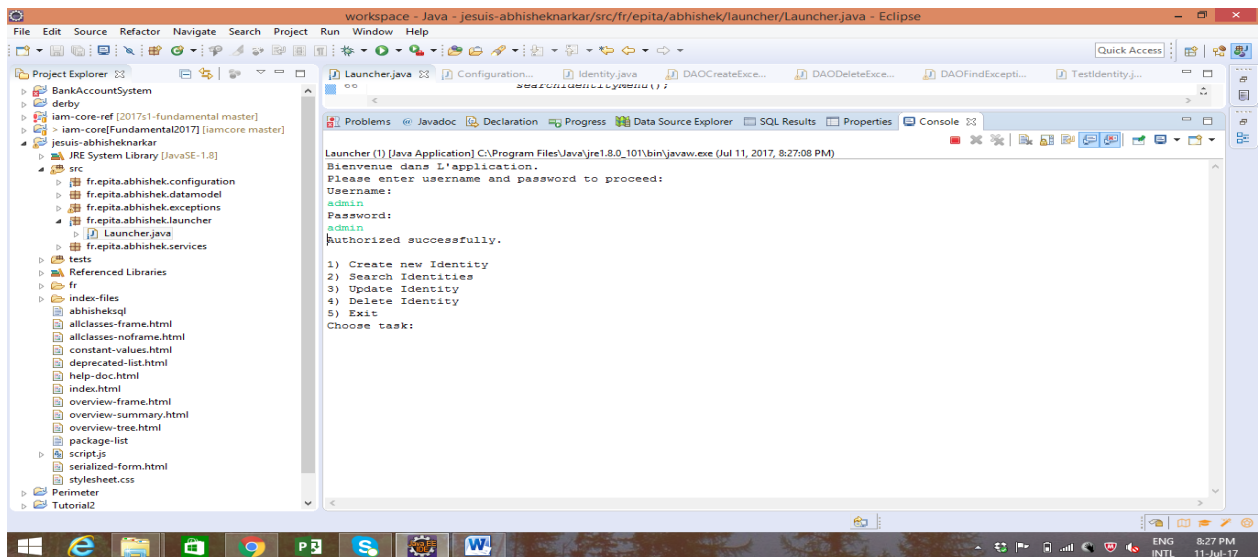
The application flow was presented in the requirements of the project as the following diagram.



So, the implementation respected this diagram in order to fulfill the requirements. A description of this flow is done with screenshots in the following pages.

#### Authentication and Menu

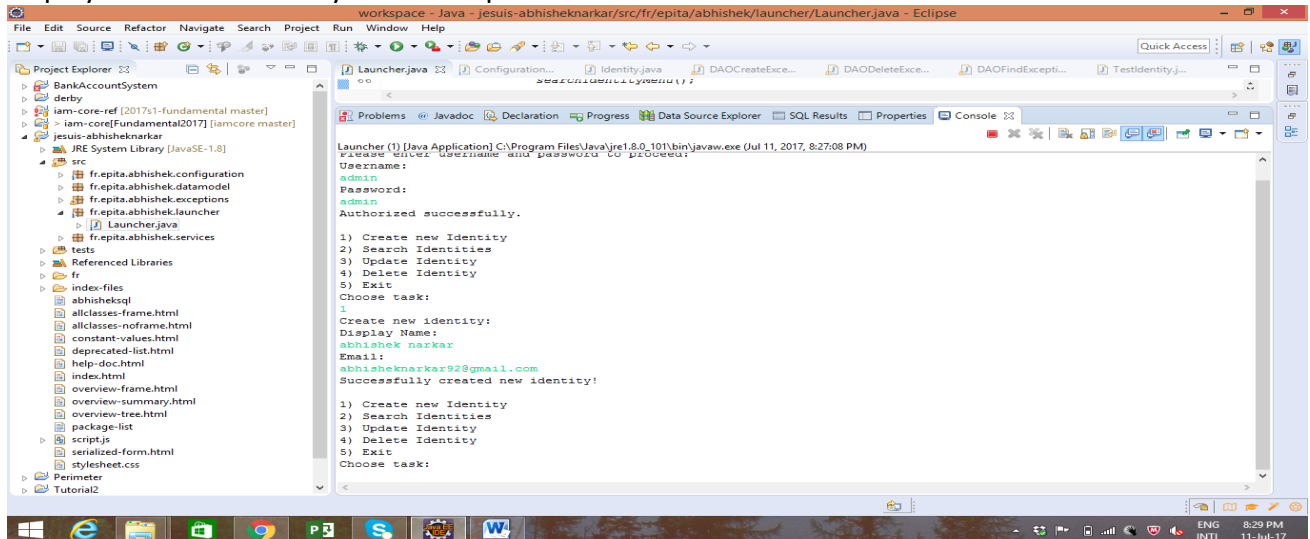
The Application first requests the user for authentication and then displays a menu so the user can choose between 5 options.



### Option 1: Create an identity

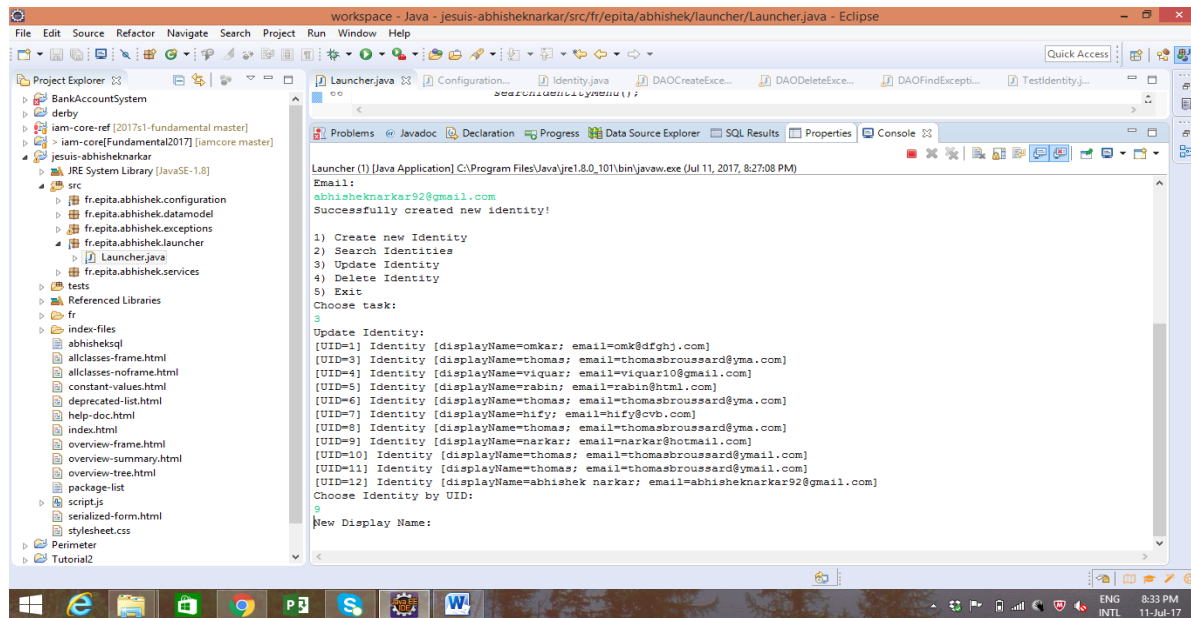
When this option is selected the user is required to input the Display Name, email and birthdate for the new Identity. The birthdate should be entered in the format that is compatible with Derby: “Year-Month-day” represented in 4 digit years, two digits for the month and also two digits for the day.

Once the data collection is completed if everything goes correctly a success message is displayed and the identity created is printed:



### Option 2: Update an identity

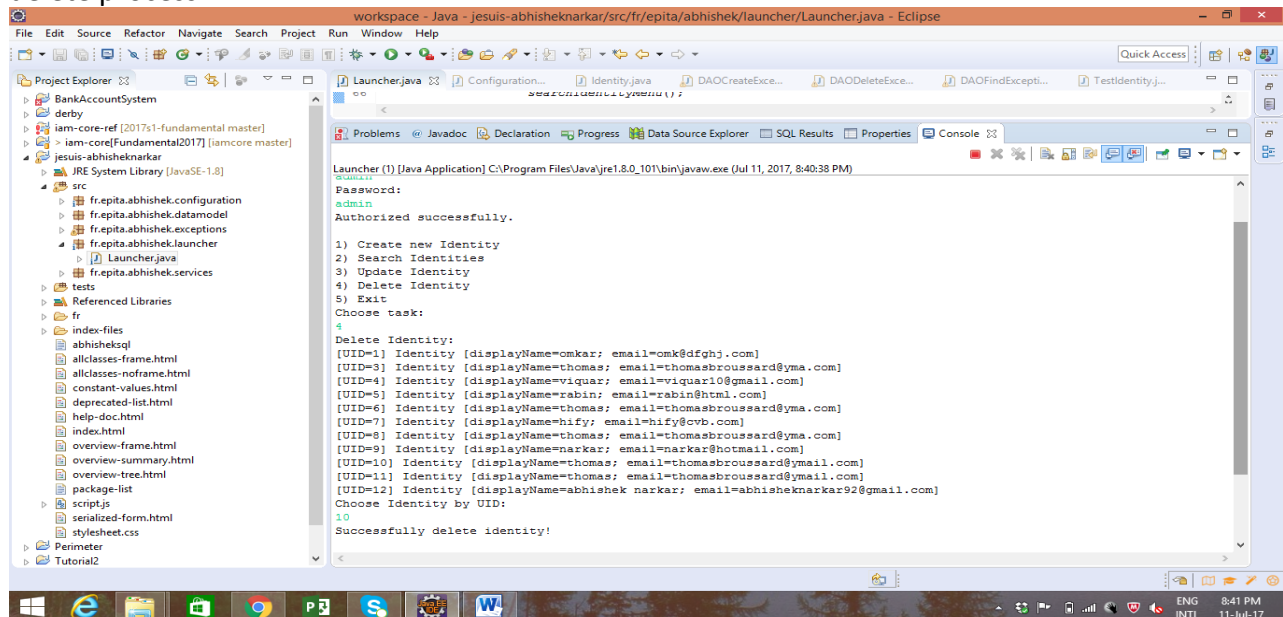
The second functionality is to update an existing identity. For this purpose, when this option is chosen a list of all available identities will be displayed to the user, and it will be required from him to input the Id of the Identity he wishes to modify.



### Option 3: Delete an Identity

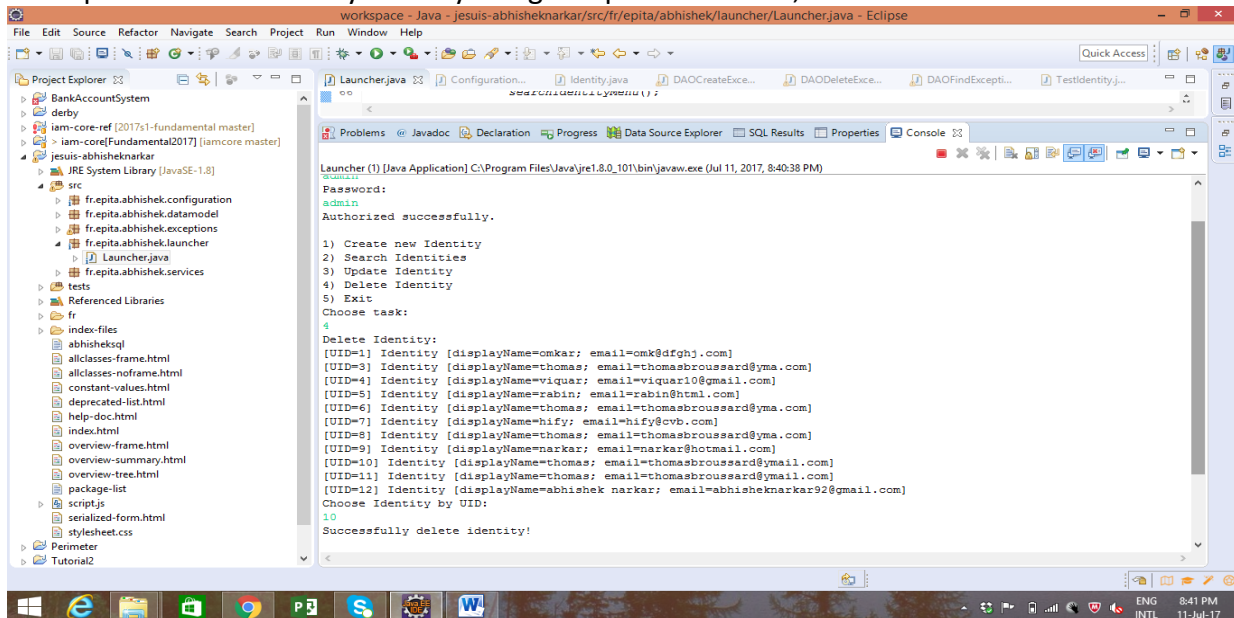
Similar to the Update Identity, the Delete Identity will first display all the available identities to the user and then let him introduce the Id of the one he wishes to delete. Again, if the Id introduced by the user is not valid, then the process is aborted.

After choosing a proper Id, the application will ask for confirmation before proceeding with the delete process.



## Option 4: Search identity

This option is to search any identity using Unique Id or name, email.

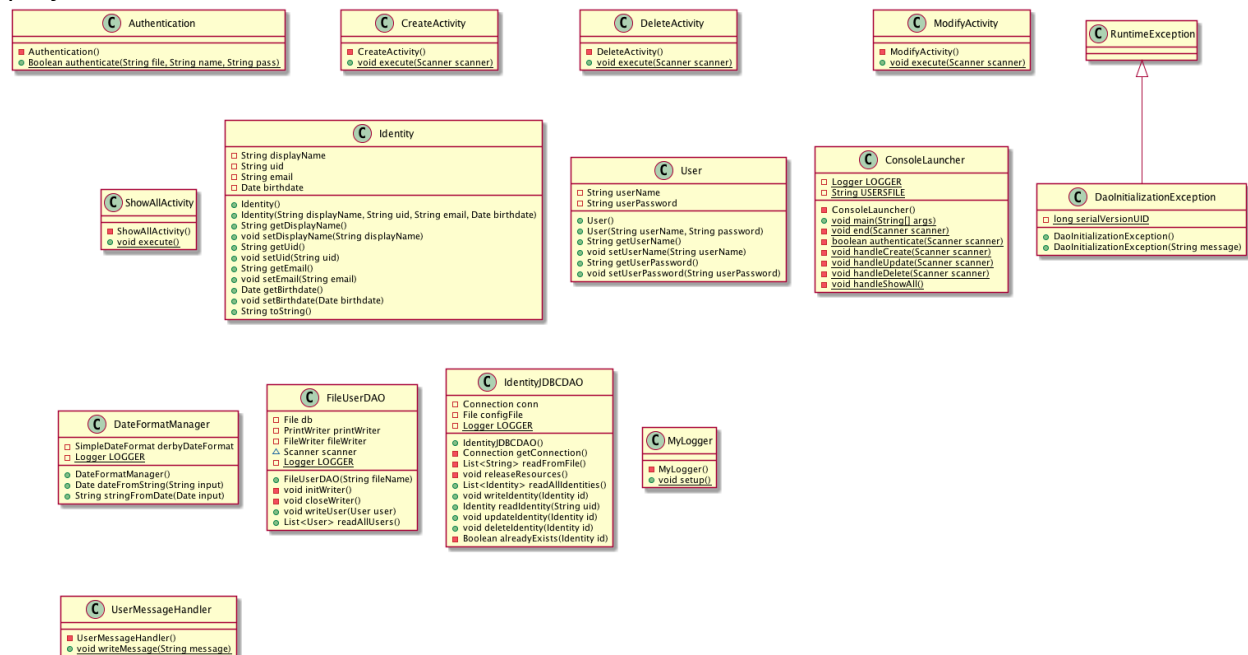


## Option 5: Quit

Terminates the program.

## Global schema and major features schema

Here is the class schema generated with PlantUML. It shows all the classes involved in the project.

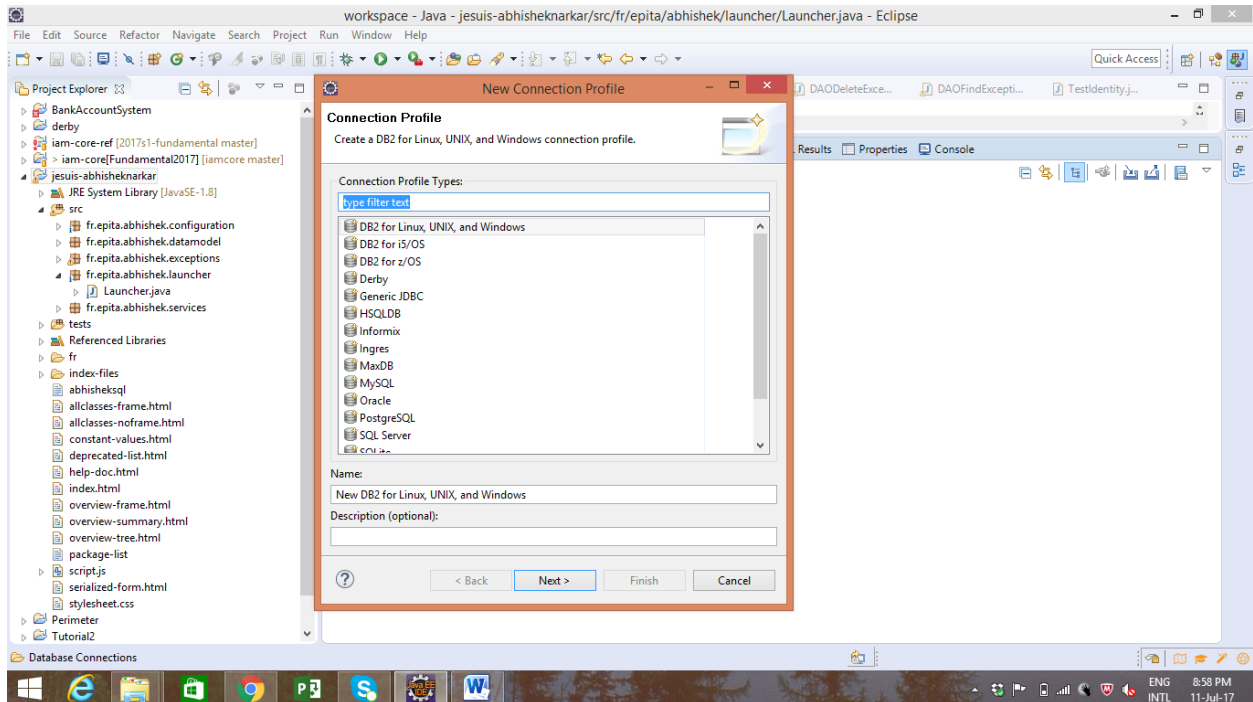





## Configuration instructions

### Prerequisites

Please make sure you have installed the Java JDK, Eclipse Neon Java EE and the Derby.



1. In the contextual window select Derby. Click Next.
2. In the new window click on the “New Driver definition button”
3. Select any Definition on the Name/Type tab and then move to the JAR List tab.  

4. On the JAR List tab click on Clear all button, so no files are listed. Then click on the Add and navigate to the lib folder inside your Derby database. Once there select the derbyclient.jar file.
5. Click Open, then the “New Derby Connection Profile” window should be updated. Please take note of the URL and change the user and password if desired. Remember those because we are going to need them later on.

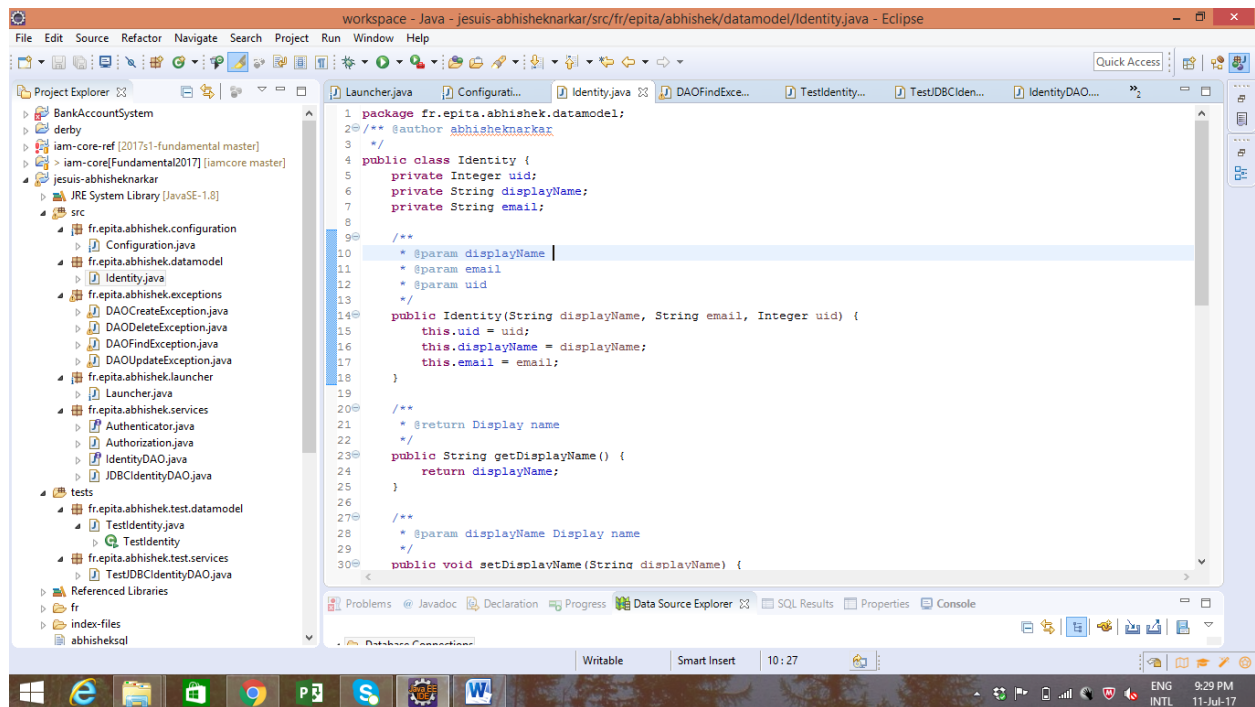
6. Make sure the Derby Database is running and click on Test Connection button, it should succeed.
7. Click on Finish button.

### Schema creation

1. Once the above process is done, within the Data Source Explorer window you should be able to expand all the way to Schemas:
2. Right click over the New Derby and select: Open SQL Scrapbook:
3. Once the new window opens, in the Project Explorer navigate: iam-core/SQL/IDENTITY table creation and open the file. Copy the file contents to the Scrapbook window and execute.
4. Right click anywhere on the Scrapbook window and select "Execute All".
5. Execution should be successful:
6. Go back to the Data Source Explorer window and right click on the Schemas folder, select Refresh. You should now see a Schema with a name that matches your user name, inside it the Identities table should have been created:
7. Once all of this is done close the Scrapbook and the Identity table creation sq. windows.

### Database Configuration

1. Open the XML found under: iam-core/Configs/DBConfig.xml
2. Select the Source view:
3. Modify the values with those according to your URL and user name and password:
4. Save the file.
5. To verify everything is ok, Run the Test file.



6. Further on you could try and run also the TestJDBCIdentity.java, all results should be true.

