D03 – Persistence models Requirements

Requirements for levels C, B, and A

Item 1. A UML domain model regarding the “Acme Handy Worker” project. Please, document your changes with regard to the version that you released in your previous deliverable and justify them.

Item 2. An Eclipse/Maven project that implements a Java domain model and a persistence model regarding project “Acme Handy Worker”. Please, document your changes regarding the version that you released in your previous deliverable and justify them.

Item 3. A set of JPQL statements that implement the queries requested in the administrator’s dashboard. Document your JPQL statements as follows:

a. Put a header of the form “Query C/1”, “Query C/2”, “Query B/1”, “Query B/2”, and the like. It must be clear the level and the query number.
b. Copy the specification of the query, which is provided in the requirements elicitation document.
c. Write your JPQL query; make sure that it can be copied from your document and pasted into the “QueryDatabase” utility. Note that some queries in the requirements elicitation document may require several JPQL statements; in such cases, you must provide a pseudo-code that helps understand how the queries must be combined together.
d. Write a short description in natural language that describes how your query works. Note that the depth of the description must depend on the complexity of the query: simple queries to compute a statistic on a single entity do not require too many explanations; complex queries that require several JPQL statements may require a bit more explanation.
e. Write the results of your query when it’s run on your sample database.

Requirements for level A$^+$

Item 4. Queries that include operator “LIKE” allow to search for objects that contain a key word in a string attribute. Unfortunately, this operator is very inefficient. We are using a component called Hibernate to implement our persistence models. Hibernate provides a technology called Full-Text Search that allows to search for arbitrary combinations of key words very efficiently.

a. Write a simple console application that allows to search your database for fix-up tasks that contain an arbitrary combination of key words in their tickers, descriptions, or addresses. The application is expected to read the key words from the keyboard and to output the results to the console. Explore classes “SchemaPrinter” and “ConsoleReader”, which are distributed with your project template. They help pretty print arbitrary objects out and read text from the console.
b. Write a report in which you explain how you have implemented your application. The report must provide enough details for another person to repeat your steps.