SDT Projects

1. Logging Framework for Java

Implement a logging framework for Java. Logging is broken into three major pieces: the Logger, Formatter and the Handler (Appender). The Logger is responsible for capturing the message to be logged along with certain metadata and passing it to the logging framework. After receiving the message, the framework calls the Formatter with the message. The Formatter formats it for output. The framework then hands the formatted message to the appropriate Appender for disposition. This might include a console display, writing to disk, appending to a database, or email. http://en.wikipedia.org/wiki/Java_logging_framework

Create a test program to show the features of your framework.

2. ORM Persistence Framework for Java

Implement an ORM persistence framework for Java that allows the programmer to store and retrieve objects from the database without using SQL statements. Your API should allow retrieving of singular objects or collections of objects based on their unique id, or using filters. The API should be a fluent API. http://en.wikipedia.org/wiki/Fluent_interface

Create a test program to show the features of your framework.

3. LAN Communication Framework

Provide a framework for the abstraction of network communication. It should contain classes for peer to peer transmission or broadcasting of text files, binaries and serialized object over TCP and RMI. The framework must allow setting up of different types of filters (size, type of files). Create a test program to show the features of your framework.

4. Smart Home

A smart home can be help monitor elderly persons or persons with disabilities. Depending on the severity and type of disability the house can be fitted with different types of sensors (temperature, light, motion detector, noise). Provide a graphical user interface that allows designers of smart homes to place the sensors and, depending on their activation, to issue custom commands (eg. if sensor1 and sensor2 are activated send notification to <email>). Simulate the system by activating the sensors when clicking on them.

5. *Robot game*

Implement a game in which you control a robot in a 2D landscape. In addition to the standard commands (move_left, move_right, move_up, move_down), more complex commands can be added to the instruction set.

The 2D grid landscape is composed of simple cells or more complex cells that can act as obstacles.

6. *CV Database*

Create a program that can process CV documents written in multiple templates, found in a template library. More templates can be added to that library. Your program should receive those documents, analyze them, store them in a relational database and be able to respond to custom queries.

7. *Enterprise Service Bus*

Create a lightweight ESB that is used to integrate data from air quality sensors. Each sensor ouputs data in a text file in a specific folder. The folder is watched continuously for changes. When a text file is changed, the ESB creates a new message, filters it for redundant data and routes it to a desired host (also a folder). The ESB is customizable allowing different processes to be added in the future.

http://www.enterpriseintegrationpatterns.com/toc.html

http://camel.apache.org/index.html

8. *Asynchronous messaging service*

Implement a messaging framework that allows hosts to communicate asynchronously, point to point, over TCP or UDP, such as Java Messaging Service. The users can send text strings or files. Do not use existing messaging frameworks.

9. *HTTP server*

Write a lightweight, multi-threaded HTTP server in Java for embedded devices with few resources, which implements the main HTTP methods.

10. *Geometry toolkit*

Create a desktop toolkit (Java SE) for school-level educators that allows them to easily create geometry problems for their pupils. The program should allow drawing geometric primitives such as lines, points and geometric shapes such as rectangles, ellipses by drag and drop or by specifying coordinates and sizes.

11. *Dependency injection framework*

Create a dependency injection framework (https://en.wikipedia.org/wiki/Dependency_injection), inspiring from Spring Framework. The framework should process either XML files either annotations to inject objects.

Observation: You will be graded according to your judicious choice, usage and documentation of design patterns in your project as well as to the overall quality of your produce software. You should use at least five non-trivial design patterns. Be advised that the design patterns are taken into consideration only if they are implemented.