Hacettepe University
Computer Science and Engineering Department

Advisers	:	RA Kazım SARIKAYA and RA Ali ÇAĞLAYAN {kazimsarikaya,alicaglayan}@cs.hacettepe.edu.tr
	:	Dr. Sevil ŞEN ssen@cs.hacettepe.edu.tr
Course	:	BİL138 Programming Lab. II
Experiment No	:	4
Subject	:	Experiment Submit Database
Deadline	:	04/05/2012 23:59:59

Contents

1 Introduction

In real world the applications process data and stores them into the file system or reads from file system. The process of reading/writing data is performed by database applications. The database applications have their own data file format.

An **API**[1] is consist of several functions, data structures, variables written before. While coding we can reuse the APIs. Hence Java applications can use different APIs from the core java classes. Theses APIs can depend other APIs. Hence management of APIs could be difficult. The Maven[2] system provides a dependency resolving system for java programming. You can add different APIs you your project and maven will download this APIs and their dependencies from a maven repository. Maven also provides several project templates and default dependencies of them. These templates are called as **archtype**s. A simple java application's template is **maven-archtype-quickstart**. Each dependecy has the main attributes **group-id** that denotes the company or main project of API and **artifact-id** that denotes the API. There is a version attribute that you should choose the newer one.

JSON[3] is a format for serializing the objects. There are several APIs for writing or reading JSON formatted objects. Most useful one is Jackson JSON API[4]. The API can be found inside the maven repositories. The group-id of API is **org.codehaus.jackson** and the two artiact ids of APIs are **jackson-core-asl** and **jackson-mapper-asl**. The all operations can be performed with the class **ObjectMapper**.

While object oriented programming with java you use object patterns. Two of them are **State**[5] and **Memento**[6] **Pattern**s. State pattern is useful while parsing String. A state means a part of work, like a method. When you parse some words from the string you will change the state into a new one. The Memento pattern is useful while storing the previous values of the class. It can be used for databases for querying the values of entities at a special date. Also it can be used for revisions of the data.

2 Experiment

This section contains four subsections. The first one is about the problem, and the second one is about the report. Second section describes how you will write your report. The third one is the about **constraints** that you should obey. Otherwise your experiment will not be evaluated by your adviser RA Kazım SARIKAYA and RA Ali ÇAĞLAYAN. The last one is about the submission.

2.1 Problem Definition

In this experiment you should design a java application which performs operations of a experiment submission database. The application has own query language. The application should read queries from the console and prints results to the console, The printed result should be an error message if an error occurred or "OK" if operation is successful or prints results of operation for **get** query with JSON format. Also the application should store the data inside the several files with JSON format.

2.1.1 Project Type

The project should be a maven project with archtype **maven-archtype-quickstart**.

2.1.2 Data Types

In this database there are three entities: **User**, **Assignment**, and **Assignment User**. The entity User have attributes **user id**, **name**, **surname** and **user type** attributes. User type attribute should be one of **adviser** or **student**. The user id is an unique value, hence you should use only this attribute inside equals method. The entity Assignment have attributes **assignment id**, **assignment name**, **start** and **end date**s. The assignment id is an unique value, hence you should use only this attirbute inside equals method. The entity Assignment User have attributes **assignment id user id** and **score**. Both assignment id and user id is an unique value, hence you should use only these two attribute inside equals method. The attribute score should be float and the attributes start and end dates should be *java.util.Date*. Other attributes should be *String*. The string format of Date will be YYYY-MM-DD**T**HH:mm:ss. In here **T** is a constant.

Each entity should have use **Memento Pattern**. Before changing any attribute of the instance, a clone of the instance should be stored inside the class with an incremental **revision number**. Hence an entity should have **revision number** and **revisions** attributes. The revisions attribute should be an array type of the entity class. While cloning set clones revisions attribute to null.

All attributes should have getter and setter methods. All entities should have a parameterless public constructor.

2.1.3 Query Language

The database has its own query language. The query will be lower case. The format of the language is

OPERATOR OBJECT [FILTER] [VALUES]

The OPERATOR is one of **get**, **add**, **update**, **delete**. The OBJECT is one of entity name. The FILTER has the format

filter $< attributename >=< attributevalue >$ ...

The VALUES has the format

set $< attributename >=< attributevalue >$ ...

An example for **add**

add assignmentuser set userId=user1 assignmentId=assign1 score=75

An example for **delete**

delete user filter userId=user2

An example for **update**

update user filter userId=user3 set name=xxx

An example for **get**

get assignment filter assignmentId=assign1

While parsing the query you should use **State Pattern**. There are four states: **OPERATOR**, **OBJECT**, **FILTER** and **VALUES**.

2.1.4 Array Beans

For each entity there should be an **array bean** class. There should be a abstract **array bean base** class which should have abstract methods

- public abstract Object create(String values);
- public abstract void add(Object o);
- public abstract void delete(Object o);
- public abstract Object get(String filer);
- public abstract void persist(String file);

Each array bean class should have an array attribute with its type (Example UserArrayBean should have an array attribute with type User). This attribute should have **setter** and **getter** methods. The **create** method should create a new instance of entity. However the entity should not added into the array bean. You will use the parameter that comes from *VALUES* (omit the keyword *set*). The **add** method should add the given object to the array bean, the **delete** method should remove the given object from the array bean. The **get** method should return the filtered values from the array bean. If filter is empty then all values should be returned. The parameter **filter** is the same value from *FILTER* (omit the keyword *filter*). Notice that the getter method of array and get methods are different methods. The persist method should print the values to the file that denoted with the parameter. Each array bean class should have a static **load** method which should have a String parameter that denotes the file name where entity values should be read.

Array bean classes should use JSON format for reading from the file and writing to the file. You should use **ObjectMapper** class of **Jackson JSON** api. For using this API you should add the dependencies **jackson-core-asl** and **jackson-mapper-asl** to the maven project. Each entity should have a file with name $< entityname >$ *.txt* like user.txt inside working path.

While operations you should load array bean from the file then perform query operation then persist array bean.

2.1.5 Exceptions

You should write your **Exception** classes for each exception that not exists inside java such as query parse exception.

2.2 Report

In your report you should explain the usage of maven projects, Memento and State pattern usages, JSON file format and IO operations. Also you should give a class diagram of project.

2.3 Constraints

You should obey these constraints. Otherwise your experiment will not be evaluated as well as you expected.

1. The project should be a **maven project**.
2. Obey class names; attribute types and names.
3. Obey java naming convention.
4. Use JSON format for files.

2.4 Submission

You should name your project **experiment4**. You should **zip** the experiment4 folder. Inside **src** folder put the **experiment4.zip** and, inside the **report** folder put **report.pdf**. Zip the src and report folders and then submit it.

3 Grading

Your Experiment results will be evaluated with the following rules. They are fixed. They will never be changed. Hence obey these rules. Before you contest your grade re-evaluate yourself with these rules.

- Submission is 1 point.
- Report is 29 points.
- Execution is 20 points.

- State pattern is 10 points.
- Memento pattern is 10 points.
- File IO with 10 points.
- Exception handling 5 points.
- Object Oriented Programming 10 points.
- Java Naming Convention 5 points.

4 Notes

These notes are very important please read them and obey them.

- **No submissions except Submission System of Computer Engineering Department will be accepted. Do not send your experiment with emails.**
- **Do not miss the deadline. No submissions will be accepted after 04/05/2012 23:59:59.**
- Respect your adviser RA Kazım SARIKAYA and RA Ali ÇAĞLAYAN office hours: Tuesday, 15:00-16:45 (RA K. SARIKAYA) and Wednesday, 14:00-16:00 (RA A. ÇAĞLAYAN).
- For every question or problem please contact with your adviser RA Kazım SARIKAYA and RA Ali ÇAĞLAYAN. Do not contact with teacher of experiment directly.
- You can send email to your adviser's email {kazimsarikaya,alicaglayan}@cs.hacettepe.edu.tr.
- **Cheating is forbidden. Collaboration is forbidden. Both of them will be graded with F3**.
- Use meaningful names for your variables, procedures and functions.
- Obey general software engineering principles.
- Save all your work until the experiment is graded
- **First do your own research about kernel development, and then ask to your adviser! An important part of this project is the research you will do during implementation.**

5 Useful Information

Under this section you will find useful information about the project. They are at the beginning level. For advanced information you need to make additional research. For cloning a Java Object you should implement the interface Clonable[7]. Then you should override the method **clone**. A clone of the object is new instance of the source object. However if you alter the source object it does not have an effect over the cloned instance. You can use the cloning feature of Java for creating revisions of the Objects.

Eclipse may not installed with build in Maven distributions. You should install Maven binaries and **m2eclipse** plugin. For installing you should choose **Install New Software** form **Help** menu. You should add the repository **http://www.eclipse.org/m2e/download/** and choose **m2e** from the list.

Netbeans comes with build in maven support.

References

[1] API http://en.wikipedia.org/wiki/Api

[2] Maven http://en.wikipedia.org/wiki/Apache_Maven

[3] JSON http://en.wikipedia.org/wiki/Json

[4] Jackson JSON API http://wiki.fasterxml.com/JacksonInFiveMinutes

[5] State Pattern http://en.wikipedia.org/wiki/State_pattern

[6] Memento Pattern http://en.wikipedia.org/wiki/Memento_pattern

[7] Clonable http://docs.oracle.com/javase/6/docs/api/java/lang/Cloneable.html