Exercise – Java Programming 4

Software Development and Software Engineering

Preparation

- Install the Git client for version control from https://git-scm.com/
- Create a Git Hub account at https://github.com/
- NOTE: It is worth checking out the Git command line client to get a feeling for the commands. However, you can also use Eclipse's eGit plug-in (should already be installed) or TortoiseGit etc. if you like a GUI.

Homework Assignment

Below are instructions for writing a very simple data processing application in Java. Produce the respective code and be prepared for discussion in class.

Extend the program to read a CSV (comma separated value) file of records regarding various cities of the last exercise so that it can process that data. The contents of the CSV file are similar to the example below:

Example CSV Data

```
Id,Year,City,Population
1,2003,"Copenhagen",501285
3,2012,"Oslo",618683
4,2004,"Copenhagen",501664
5,2009,"Austin",794520
6,2012,"Lisbon",530847
7,2008,"Oslo",567980
8,2010,"Copenhagen",528208
9,2001,"Lisbon",564657
10,2001,"Austin",689450
11,2008,"Lisbon",489562
12,2005,"Austin",714650
13,2001,"Copenhagen",499148
14,2012,"Austin",854350
15,2011,"Oslo",606258
16,2012,"Copenhagen",549050
17,2001,"Oslo",508726
18,2010,"Lisbon",474697
```

Connect with Git

- Use Git to clone the repository at https://github.com/chseidl/sdse_students.git
- NOTE: Before cloning the repository, you can also fork the repository. This way, a copy of the repository becomes part of your GitHub account and you can also push your modifications there. However, also note that you will have to make sure that its state is still up-to-date with the original repository via fetching/pulling.
- You can find the material to work on this homework in the repository you checked out under Exercises/Java4/CSVProcessor with an Eclipse project that you can import.

- As you are creating a new "feature" of the "software system", it is appropriate to create a new branch for development until the new functionality is completed and merged back into master. Create a new GIT branch "dataprocessor" and check it out.
- Commit your changes to your local repository after each meaningful program increment.

Create Data Class for CSV Records

In the previous exercise, you already parsed CSV data. Now it is time to capture the data of each record in a dedicated data class as preparation for further calculations:

- Create a class CityRecord that can hold id, year, city and population of a single line in the CSV file. Give each field an appropriate data type.
- After loading a new city record, print it to the console via System.out.println(cityRecord); (At this point, this will produce gibberish.)
- In the class CityRecord, override the method toString() to make sure that city records are formatted appropriately when printed, e.g., to the console. Have the toString() method create a string representation for the values contained in the CityRecord class similar to: "id: 1, year: 2003, city: Copenhagen, population: 501285"
- Commit your work to your Git repository using a sensible commit message, e.g., "Added data class for city records." Note that you may have to add newly created files to Git first, e.g., the CityRecord class. A graphical tool, such as TortoiseGit or eGit, will prompt you.

Write a Simple Data Processor for CSV Files

- **Collect all read records in a List**, e.g., List<CityRecord> allRecords. Make sure that the list is empty when performing another run of the processor.
- To prepare the data for processing, also use a **Map to collect records by their city**. The key of the map has to be a string (city name) and the value of the map has must be a list of city records, i.e., List<CityRecord>. In consequence, you will get a construct Map<String, List<CityRecord>>.
  - Note that map is an interface. To create a fitting instance, use the class HashMap.
  - With the method put(key, value), you can place a new entry, i.e., a list of records for a city.
  - With the method get(key), you can retrieve a previously put value.
  - Make sure that, when encountering a new city, you create a new list for the city records (as this will be null initially).
  - Also make sure that the map is empty when performing another run of the processor.
- **Process data of each city** by finding a) the total number of entries for that city, b) the minimum year, c) the maximum year and d) the average population over that period (i.e., the sum of all population measurements divided by the total number of entries).
  - You can iterate over the entries of a map when using a for-each loop on the result of calling the map's method entrySet() like so:
    ```
    for (Entry<String, List<CityRecord>> entry : recordsByCity.entrySet()) {
       //...
    }
    ```
  - An entry contains a key (the city name) and a value (the records of that city).

  - You can then iterate over the individual entries in a for-each loop like so:
    ```
    for (CityRecord record : recordsOfCity) {
       //...
    }
    ```
- **Output the processed data to the console** in a format similar to:
  Average population of Copenhagen (2003-2004; 4 entries): 501285
- Commit your work to your Git repository using a sensible commit message, e.g., “Implemented average calculation on city records.”

Wrap Up Git

- As you are done with developing the new “feature”, you should now make your work available on the master branch. To do this, switch back to the “master” branch by checking it out. Then, merge the “dataprocessor” branch into your current branch (“master”) to make those changes available on that branch.
- As you have committed your work, it is already part of your local repository. If you had forked the repository to your own GitHub account, you can now push the changes to your repository so that they are also available online (i.e., in principle, usable by others in a collaboration).

Classroom Assignments

Version Control and Versioning

1. Which are version control systems mentioned in the lecture?

- ☐ A) SVN
- ☐ B) Gimp
- ☐ C) BitKiller
- ☐ D) Mercurial
- ☐ E) Chrome
- ☐ F) Solar

2. For one of the core libraries of your system, a colleague recommends using a "pre-release" version. Argue for or against that idea with at least two benefits/drawbacks that this may entail.

3. Contrast client-server version control systems and distributed version control systems using the examples of SVN and Git. Specifically mention, for each type of system, which repositories exist, where they are stored and how communication between them works. For describing the communication, use the terms of the respective commands in SVN and Git.

Tools and Their Purpose

Below is a list of tools. Decide for each tool, which (exactly one) of the proposed answers is the correct one.

Git

- ☐ UML Diagram Tool
- ☐ Version-Control System
- ☐ Security Procedure

Cassandra

- ☐ Distributed Data Store for Big Data
- ☐ Educational Platform for Java
- ☐ Query Language for HBase Databases

Eclipse

- ☐ Tool to Automatically Generate Documentation for F# Programs
- ☐ Framework for Parallel, Distributed Data Processing
- ☐ Integrated Development Environment (IDE) suitable for Java Development

Spark

- ☐ Time Planning Tool
- ☐ Cloud Storage Service
- ☐ Framework for Parallel, Distributed Data Processing

Big-Data Tools in Java

A colleague wants to use Java and its tools to create a big-data application. However, they may have gotten some of their assumptions wrong. Decide which of these statements are true and which are false by putting a cross in the respective column:

Statement	True	False
Java has many tools for data storage, data query and calculations.		
For storing my data in the cloud, Apache Pig seems like a good solution!		
Apache HBase is the fastest data storing technology of them all and alternatives are really bad.		
Because Apache HBase builds on HDFS, it integrates well with Apache Hive.		
I will use Map/Reduce to program for Hadoop.		
Apache Gardener will help me to coordinate configurations between different machines.		