Big Data Processing Course

Spring Semester 2016

Homework Assignment 1.

March 10, 2016

Contents

1 Introduction **2**

1.1 Goals . . . 3

1.2 Get Started . . . 3

1.2.1 Install VirtualBox . . . 3

1.2.2 Install Vagrant . . . 3

1.3 Java JDK, Maven . . . 4

1.3.1 Maven . . . 4

1.3.2 Maven tutorials . . . 5

2 Assignment Details **6**

2.1 File format . . . 6

2.1.1 Content sample . . . 6

2.2 Queries . . . 7

2.3 Auxilary classes . . . 10

2.3.1 The Main Class. . . . 10

2.4 Assignment folders structure . . . 11

2.4.1 The Vagrant file . . . 11

2.4.2 Vagrant commands . . . 12

2.5 Development environments . . . 12

2.5.1 Get started with Eclipse. Importing Maven project. . . . 13

2.5.2 How to assignment will be checked . . . 15

3 Submission **17**

Chapter 1

Introduction

1. Homework submission is strictly in **groups of 4 students**, groups with less than 4 should ask for an explicit permition from the lecturer.
2. Homework deadline is due to: **27/03/2016 23:59**.
3. Please pay attention to the due date. I urge you to start working on the homework right away and not wait until the last minute.
4. Please try to provide clear and careful solutions.
5. You should provide comments for your code so it will be completely clear what you are trying to achieve.
   - **WARNING!** Lack of comments might lead to points reduction.
6. Homework will pass automatic check, therefore please pay attention to output format and required processing of input data, points will be automatically reduced for non complaint output (even if this is only a single digit or character).
7. Please remember that a significat part iof your homework solution is understanding of requirements, therefore I urge you to read through this document very careful and make sure you understand each single part of it. Solutions verified with automatic tooling, therefore in case of misunderstanding of requirements or in case of non compliance, points will not be returned back.

1.1 Goals

The main purpose of first homework assignment is to bridge the gap and teach you basic Java concepts of developing the code, working with IDE and testing. During the assignment you will learn how to implemement data mining application with Java. The assignment covers different aspects of work with file/disk IO and resource management. Since course assumes common and base knowledge of Java programming it's completely up to you fullfill missing gap by reading online material and books proposed on the moodle course site. Questions of how to complete different programming goals in Java will not be covered during lectures and course staff doesn't eligleble to answer on questoins which related to it.

1.2 Get Started

Following the instructions, which allows you to get started with work on your assignment solution. In order to provide unified development and testing environment you're required to validate and compile your solution using virtual boxes. Since our check is automated you also required to work with **Vagrant** to manage virtual machine activity. Please read carefully tutorials following links below based on your preferences.

1.2.1 Install VirtualBox

VirtualBox is a cross-platform virtualization application. For one thing, it installs on your existing Intel or AMD-based computers, whether they are running Windows, Mac, Linux or Solaris operating systems. Secondly, it extends the capabilities of your existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time. So, for example, you can run Windows and Linux on your Mac, run Windows Server 2008 on your Linux server, run Linux on your Windows PC, and so on, all alongside your existing applications. You can install and run as many virtual machines as you like – the only practical limits are disk space and memory.

VirtualBox is deceptively simple yet also very powerful. It can run everywhere from small embedded systems or desktop class machines all the way up to datacenter deployments and even Cloud environments. Bellow links to the tutorials which explains how to install Virtualbox on different environments, please make sure you download and install VirtualBox of version 5.0 or greater.

Windows

Virtualbox installation for Windows platform follow instructions from follow link:
http://www.htpcbeginner.com/install-virtualbox-on-windows/

Linux (Ubuntu)

Virtualbox installation for linux Ubuntu distribution platform follow instructions in follow link:
http://ubuntuhandbook.org/index.php/2015/07/install-virtualbox-5-0-ubuntu-15-04-14-04-12-04/

1.2.2 Install Vagrant

Vagrant is the application which aims to help in creating virtual machines in automatic manner, facilitating many reusable options and configurations. Since during the course you will learn how to work with clustered environment of Hadoop and Spark nodes. Vagrant will help you to create and setup such clustered environments on your laptops/private computers and validate your solutions. Moreover using Vagrant files will enable unified automatic check of your assignments. Bellow links to the tutorials which explains how to install Vagrant.

Vagrant binaries could be downloaded here: https://www.vagrantup.com/downloads.html.

Window

Vagrant installation instruction for Windows platform can be found here:
http://www.sitepoint.com/getting-started-vagrant-windows/

Linux(Ubuntu)

Following instructions to install Vagrant on Ubuntu:
https://www.godaddy.com/garage/tech/config/install-vagrant-ubuntu-14-04/

1.3 Java JDK, Maven

In this assignment you are required to develop using Java JDK of version 1.8, you will be provided with **Vagrant** definition file which will include installation of correct Java version and it will be used to validate your solutions. You are free to develop using any other version of Java as long as it will compile on given virtual machine. Java project management will be held using **Maven**, maven is a software project management and comprehension tool. Maven can manage a project's build, reporting and documentation from a central piece of information.

1.3.1 Maven

Mavens primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Providing guidelines for best practices development
- Allowing transparent migration to new features

The POM

The pom.xml file is the core of a project's configuration in Maven. It is a single configuration file that contains the majority of information required to build a project in just the way you want. The POM is huge and can be daunting in its complexity, but it is not necessary to understand all of the intricacies just yet to use it effectively. This project's POM is:

Listing 1.1: pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
```

```
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>Maven Quick Start Archetype</name>
    <url>http://maven.apache.org</url>

    <dependencies>
      <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.8.2</version>
        <scope>test</scope>
      </dependency>
    </dependencies>
</project>
```

Dependencies section helps to manage project external libraries dependencies, for example you can add new dependency by:

```
<dependency>
  <groupId>com.google.guava</groupId>
  <artifactId>guava</artifactId>
  <version>18.0</version>
</dependency>
```

This adds dependency for Google Guava library (https://github.com/google/guava) version 18.0. This way you will be able to add any dependency to external libraries which expose functionality you need for your project. In this assignment you will be provided with ready to use **pom.xml** file, however you allowed to change it according to your needs.

1.3.2 Maven tutorials

We recommend you to cover other different aspects of maven by reading following tutorials:

- http://www.vogella.com/tutorials/ApacheMaven/article.html
- http://www.tutorialspoint.com/maven/
- https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
- http://tutorials.jenkov.com/maven/maven-tutorial.html

Chapter 2

Assignment Details

In this assignment you provided with file which includes movies reviews information collected from online resources. You are required to use this file to provide implementation of data analytic algorithms to asnwer different queries which will be described in the following section 2.2.

2.1 File format

Reviews files consist of 7 tab separated fields as following:

1. **product/productId** - the id of the movie from the online resource
2. **review/userId** - id of the user which provided review
3. **review/profileName** - name of the user profile
4. **review/helpfulness** - how helpful given review (rate by other users)
5. **review/score** - the rating score given by user
6. **review/time** - the timestamp in milliseconds, the time when review was submitted
7. **review/summary** - review summary
8. **review/text** - review text

2.1.1 Content sample

```
product/productId: B00004CK40  review/userId: A39IIHQF18YGZA review/profileName:
C. A. M. Salas  review/helpfulness: 0/0 review/score: 4.0 review/time: 1175817600
review/summary: Reliable comedy review/text: Nice script, well acted comedy,
and a young Nicolette Sheridan. Cusak is in top form.

product/productId: B00004CK40  review/userId: AP8GQ56CJ8MYC review/profileName:
C. MACHADO  review/helpfulness: 0/0 review/score: 5.0 review/time: 1171929600
review/summary: I still love this movie review/text: I first saw this movie when
it was released in the 80's. I loved it. Some 20 years later I had the opportunity
to view it again, and I still love it. It's storyline is timeless as is it's humor.
I found the exchanges between the two lead characters so natural and FUNNY. It is
still one of my very favorite movies of all time. John Cusak is fabulous!! Please
see it. If you want to feel good, it's a movie that will help you get there.
```

2.2 Queries

Extracting usefull information out of row data is very important and useful process in data mining. Your mission will be to provide an ability to answer different types of queries over the given dataset of movies reviews. In order to accomplish your goal you provided with interface and classes which you will have to implement and fullfill missing methods, moreover you are provided with main interface **IMoviesStorage** which defines queries you will have to answer during the assignment. Here is the list of the queries and interface signatures:

1. Compute overall average score for all reviewed movies

Listing 2.1: totalMoviesAverageScore

```
/**
 * Method which calculates total scores average for over all movies.
 *
 * @return - the average
 */
public double totalMoviesAverageScore();
```

2. Given movie (the product id), compute total average for that movie

Listing 2.2: totalMovieAverage

```
/**
 * For given movies calculates an average score of this movie.
 *
 * @param productId - id of the movie to calculate the average score.
 * @return - movie's average
 */
public double totalMovieAverage(final String productId);
```

3. Compute a list of overall total averages for all movies in the dataset. List should be sorted according the average score of movie and in case there are more than one movie with same average score, sort by product id lexicographically in natural order.

Listing 2.3: getTopKMoviesAverage

```
/**
 * For each movie calculates it's average score. List should be sorted
 * by average score and in case of same average tie should be broken by
 * natural order of product id.
 *
 * @param topK - the limit of movies to compute the average score.
 *
 * @return - list of movies where each @{@link Movie} includes it's average
 */
public List<Movie> getTopKMoviesAverage(final int topK);
```

4. Locate the movie with the highest average score across all movies in the dataset.

Listing 2.4: movieWithHighestAverage

```
/**
```

```
     * Finds movie with the highest average score among all available movies.
     * If more than one movie has same average score, then movie with lowest
     * product id ordered lexicographically.
     *
     * @return - the movies record @{@link Movie}
     */
    public Movie movieWithHighestAverage();
```

5. Return list of movies from given percentile of average score. List should be sorted according the average score of movie and in case there are more than one movie with same average score, sort by product id lexicographically in natural order. This method should return the list of movies for which the avarage score is above or equal than average score of given percent parameter.

Listing 2.5: getMoviesPercentile

```
    /**
     * Returns a list of movies which has average of given percentile
     *
     * @param percent - the percentile, value in range [0..100] (\%)
     * @return - movies list
     */
    public List<Movie> getMoviesPercentile(final double percent);
```

6. Return product id (the movie) which has hiest amount of distinct reviews.

Listing 2.6: mostReviewedProduct

```
    /**
     * @return - the product id of most reviewed movie among all movies if more th
     * one movie has same amount of user reviews tie should be broken by product
     * lowest (lexicographically) should come first.
     */
    public String mostReviewedProduct();
```

7. Return a map of product id and number of review the movies has received according to the dataset you are provided with. Map should be sorted according to the number of reviews in decreasing order in case of equality ties should be broker by natural order of product id.

Listing 2.7: reviewCountPerMovieTopKMovies

```
    /**
     * Computes reviews count per movie.
     *
     * @return - returns map with movies product id and the
     * count of over all reviews assigned to it.
     */
    public Map<String, Long> reviewCountPerMovieTopKMovies(final int topK);
```

8. Compute most popular(with highest average score) movie which was reviewed by at least K users.

Listing 2.8: mostPopularMoviewReviewedByKUsers

```
/**
 * Computes most popular movie which has been reviewed by at least
 * numOfUsers (provided as parameter). If there are more than two movies
 * with same number of reviews movie with lowest product id should be picked
 * up, according to lexicographical order.
 *
 * @param numOfUsers - limit of minimum users which reviewed the movie
 * @return - movie which got highest count of reviews
 */
public String mostPopularMovieReviewedByKUsers(final int numOfUsers);
```

9. Construct the map of words used for movies review and number of word occurences. Consider words as separated by space.

Listing 2.9: moviesReviewWordsCount

```
/**
 * Compute map of words count for top K words, words are separated by space.
 *
 * @param topK - top k number
 * @return - map where key is the word and value is the
 * number of occurrences of this word in the reviews
 * summary, map ordered by words count in decreasing order, for words
 * with same number of occurances lexicographocal order should be applied.
 */
public Map<String, Long> moviesReviewWordsCount(final int topK);
```

10. Give a map of top X words count for top Y most reviewed movies, sorteded by words counts in decreasing order.

Listing 2.10: topYMoviewsReviewTopXWordsCount

```
/**
 * Compute words count map for top Y most reviewed movies.
 * Map includes top K words.
 *
 * @param topMovies - number of top review movies
 * @param topWords - number of top words to return
 * @return - map of words to count, ordered by count in decreasing order, for
 * with same number of occurances lexicographocal order should be applied.
 */
public Map<String, Long> topYMoviewsReviewTopXWordsCount(final int topMovies,
                                                         final int topWords);
```

Listing 2.11: topKHelpfullUsers

11.

```
/**
 * Compute top k most helpful users that provided reviews for movies
 *
 * @param k - number of users to return
 * @return - map of users to number of reviews they made.
```

```
     * Map ordered by number of reviews in decreasing order or
     * according to the user id sorted lexicographically.
     */
    public Map<String, Double> topKHelpfullUsers(final int k);
```

12. Total movies count

Listing 2.12: moviesCount

```
    /**
     * Total movies count
     */
    public long moviesCount();
```

2.3 Auxilary classes

In addition to the **IMovieStorage** interface you provided with following classes: **Movie**, **MovieReview** and **MovieProvider** interface which encapsulates capability of reading and parsing the sources with movies reviews. You need to add your implementation for **FileIOMoviesProvider** which inherits (implements) **MovieProvider** interface to read movies file from the disk. **FileIOMoviesProvider** need to be used inside your implementation of **IMovieStorage** to give an ability to answer the queries.

2.3.1 The Main Class.

Also you provided with a sckelethon of the main java class which will execute queries and dump the output into the file. The class name is **MoviesReviewsQueryRunner** it has only one method **public static void main(String...args);** the main class receives two input parameters:

- **-inputFile** - the name of the file with movies reviews
- **-outputFile** - the name of the file where program will output query results.

The main capability of this class is to create and initialize relevant object to read file with movies reviews and provide an ability to answer on queries and output into file ("outputFile").

You may assume that in automatic check you will be given with file of size which fully fits in memory.

2.4 Assignment folders structure

Listing 2.13: Assignment files

```
|---|
    |--- Vagrantfile
    |--- grade.sh
    |
    |--- hw1
           |---|
               |--- pom.xml
               |--- src
               |        |---|
               |            |--- main
               |                    |---|
               |                        |--- java
               |                        |       |---|
               |                        |           |--- univ.bigdata.course
               |                        |--- resources
               |                                |---|
               |                                    |--- movies-sample.txt
               |                                    |--- queries_out.txt
               |--- target
```

You should not change the folders structure, you need to import hw1 (maven project) into your IDE. The **hw1** is mounted inside virtual machine to run tests and validations, hence it need to stay untouched.

- **Vagrantfile** - vagrant file with definitions of virtual machine you are going to work with
- **grade.sh** - the shell script to validate correctness of your output
- **hw1** - the main Java project folder
- **pom.xml** - maven project configurations files with base definitions
- **src** - actual project sources folder
- **java** - the src subfolder where you will keep all java classes, interfaces and implementations
- **resources** - the src subfolder where you have an example input file and relevant output files for you to compare and validate your implementation during the work on assignment
- **target** - folder where maven builder will keep compiled Java binaries

2.4.1 The Vagrant file

The content of vagrant file in your assignment is following:

Listing 2.14: Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
```

```
  config.vm.synced_folder "hw1/", "/home/vagrant/hw1"
  config.vm.provider "virtualbox" do |v|
    v.memory = 2048
  end
  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update && apt-get upgrade -y
    sudo apt-get install -y vim curl wget telnet htop nmon
    wget --no-check-certificate \
        https://github.com/aglover/ubuntu-equip/raw/master/equip_java8.sh
    ./bash equip_java8.sh
    wget --no-check-certificate \
        https://github.com/aglover/ubuntu-equip/raw/master/equip_maven3.sh
    ./bash equip_maven3.sh
    sudo echo "PATH=$PATH:/opt/maven/bin" >> /etc/profile.d/maven.sh
  SHELL
end
```

In this assignment you are not supposed to change content of this files. However I'd like to strictly encourage you to learn about the format of this file and different options it might provide, since you will need it for further homeworks.

2.4.2 Vagrant commands

Here listed most useful vagrant commands you might need during your work with virtualized environment (Virtualbox VMs)

- Startup and create virtual machine using definitions file

  ```
  vagrant up
  ```

- Stop running virtual machine

  ```
  vagrant halt
  ```

- Connect to available and running virtual machine

  ```
  vagrant ssh
  ```

- Destroy virtual machine and release all resources used by it

  ```
  vagrant destroy
  ```

- Checking current status of virtual machines

  ```
  vagrant status
  ```

More details and command could be found here: https://www.vagrantup.com/docs/

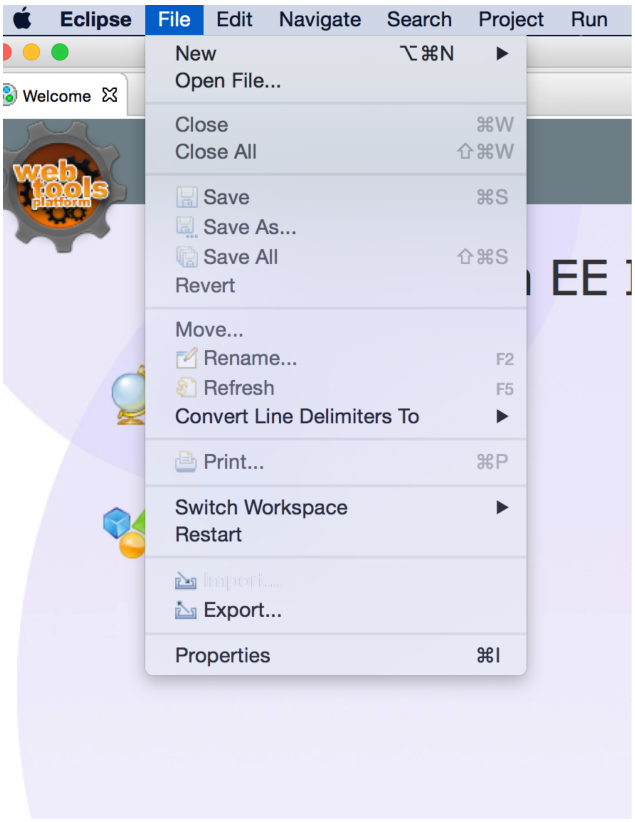
2.5 Development environments

You are free to work with any of available Java IDEs (Netbean, Eclipse, IDEA) and you can install whatever java you'd like to use during the development. But pay attenttion that all your homeworks will be checked using virtual machines and vagrant script described above.

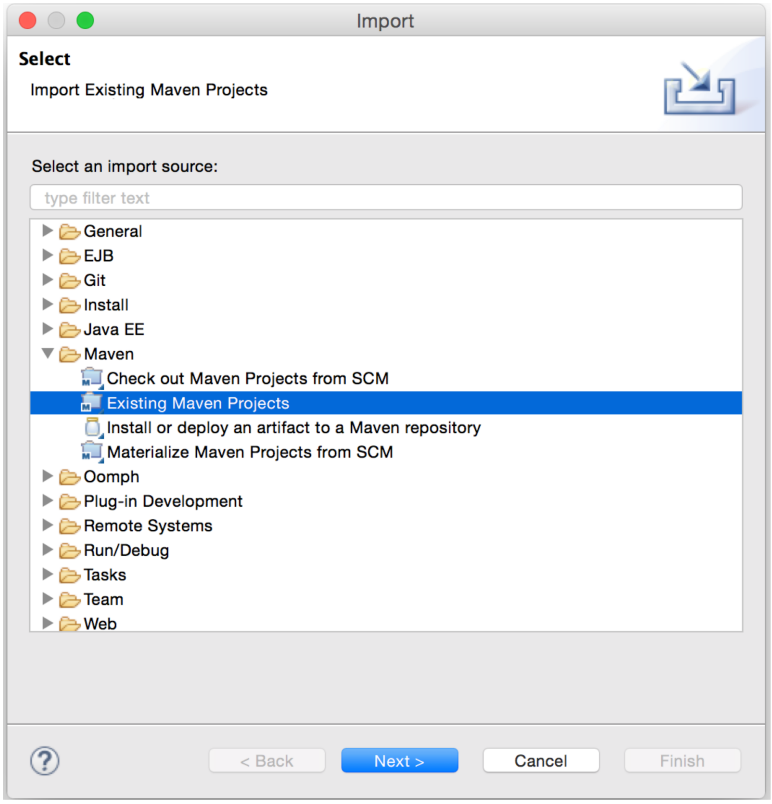
2.5.1 Get started with Eclipse. Importing Maven project.

Following set of steps you need to follow in order to import maven project into Eclipse workspace and start working on your assignment.

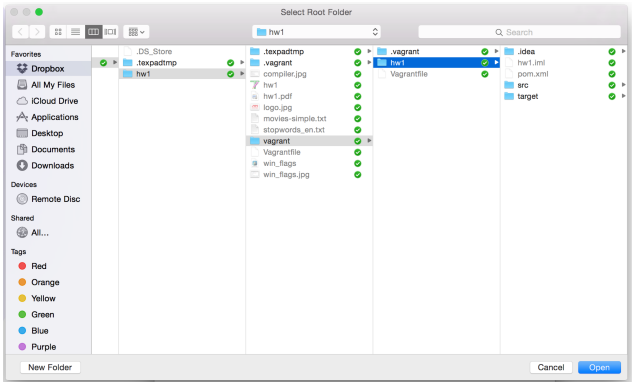
1. Start eclipse.
2. Go to File − > Import..



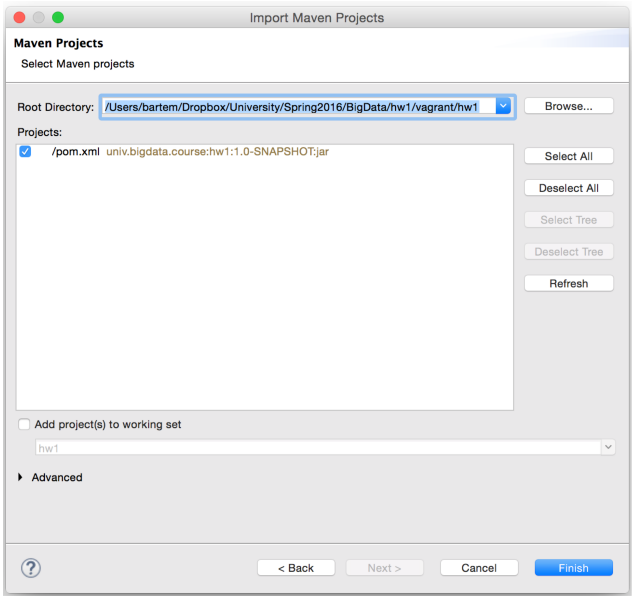
3. Select Existing Maven Projects



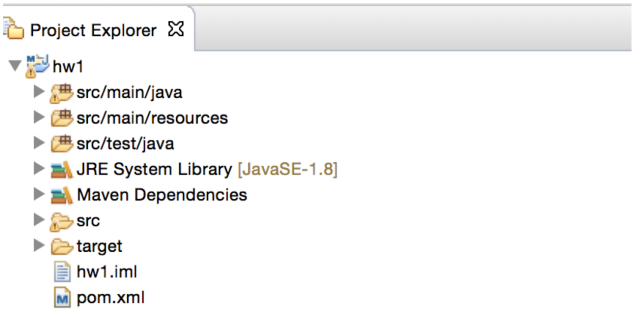
4. Choose the hw1 folder provided in assignment



5. Click on Finish



6. Switch to project explorer view, you should see:



7. You can start developing.

2.5.2 How to assignment will be checked

1. Create a virtual box by:

   ```
   vagrant up
   ```

   It might take a frew minutes for very first time and require connection to the internet. Virtual machine base image will be downloaded and provisioned by vagrant with commands from the **Vagrantfile**.

2. Connect to the virtual box:

   ```
   vagrant ssh
   ```

3. Go to the project folder:

   ```
   cd /homework/vagrant/hw1
   ```

4. Execute maven compilations and build goals:

```
mvn clean install
```

After execution of these command the entire project will be compiled and validated, so you need to wait for **BUILD SUCCESS** message. Once you get the message that means project has been successefully compiled.

5. Execute your implementation on sample input and validate it:

```
mvn exec:java && ./grade.sh
```

You should see the output:

```
Your output is correct. PERFECT MATCH!!!
```

This will indicate correctness of your code otherwise it will present you a differences in the output.

Please note that your assignment will be checked inside the virtual machine using the process described above, although you are free to develop and debug your implementation in the most conviniet way which suits you. Pay attention to validate and verify your project still compiles using virtual box.

Chapter 3

Submission

This is an electronical submission via course portal on Moodle. To complete your submission you have to:

- Provide your students details at the top of each submitted file.
- Add **hw1_sol.pdf** file with short description of the implementation details and description which explains what was the responsibilities of each teammate during the assignment.
  - **NOTE:** Without this files 10pt will be reduced.
- Add **README.txt** file which will also include information in the format:

Listing 3.1: README.txt

```
First and Second Name;email1;ID1
First and Second Name;email2;ID2
First and Second Name;email3;ID3
First and Second Name;email4;ID4
```

- All files should be archived with zip and final name for submission should be **homework1.zip**.
- **NOTE:** submissions which will be missed **README.txt** will not be graded.

GOOD LUCK!