VE472 — Methods and tools for big data

Lab 1

Manuel — UM-JI (Summer 2022)

Goals of the lab

- Write and use packages in Java
- File input and output in Java
- Object-oriented programming in Java

1 Introduction

In this lab we assume a good C++ background and help you turning it into the ability of Java programming. A simple guideline can be found in the appendix A.

Your friend Reapor Rich plans to build a new type of cinema in Singapore. The seats are assigned automatically by a seat management system when customers buy tickets. The structure of a typical cinema hall is shown on figure 1, and we can observe that three seats are selected.

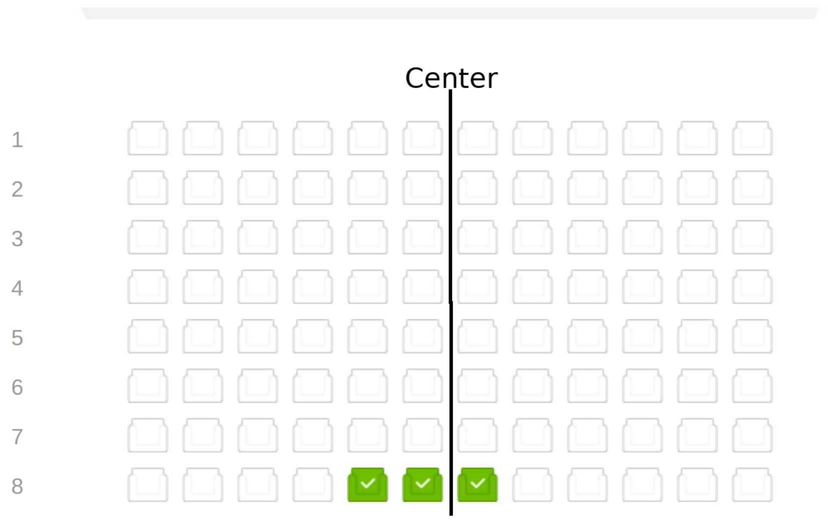

Figure 1: Structure of a typical cinema hall.

When people choose their seat they always want to stay together and prefer the seats in the middle and back of the cinema. In figure 1, your other friends, Krystor, Frank, and Simon selected the three best seats in the hall.

According to this golden rule, the management system will assign the seats within the following rules:

- When a customer wants to buy n tickets, the system always selects n consecutive seats on a row. The seats are "consecutive" means that they are next to each on the same row.
- The system always selects the seats closest to the middle back of the cinema hall. It means that the distance between the centroid of the seats and the middle back point (row 8, column between 6 and 7 on figure 1) should be minimized.
- When two selections on different rows have the same distance, choose the row with the larger row number.

- When two different selections on one row have the same distance, choose the centroid on the left.
- If no selection satisfies the rules above, deny the customer's request of buying tickets.

For example on figure 1, as the centroid is $(8, 6)$, and the middle back point $(8, 6.5)$,then the distance is 0.5, which is the smallest in all selections.

Hint: you are not recommended to use square root when comparing the distance, as it may lose some accuracy. This also applies to the analysis of big data composed of integers.

2 Java environment setup

In this section we install the Java environment and learn how to create and install basic Java packages.

2.1 Installation

First you need a Java SDK (Java Software Development Kit, or JDK) installed on your system. There are two implementations for Java platform programming: Oracle JDK (preferred), the official Oracle version of Java Development Kit; and OpenJDK a free and open-source implementation of the Java SE Platform Edition. For this course, you should install Java 8 or above.

2.1.1 Windows

Windows is **not recommended** in this course. If you really want to use Java on Windows, download Oracle JDK from `https://www.oracle.com/technetwork/java/javase/downloads/index.html` and run the installer.

2.1.2 Linux

- Debian and derived distribution, e.g. Ubuntu, Linux Mint.

  ```
  sh $ apt-get update
  sh $ apt-get install software-properties-common
  sh $ add-apt-repository <repository>
  sh $ apt-get update
  sh $ apt-get install oracle-java<X>-installer
  ```

  Replace `<repository>` with a third-party repository, and `<X>` with the version number. Third-party repositories can be found at `https://launchpad.net/`, e.g. Java 8: `ppa:webupd8team/java`, or Java 12: `ppa:linuxuprising/java`[1].

- Arch Linux and derived distributions, e.g. Manjaro: first install yay.

  ```
  sh $ sudo pacman -S git
  sh $ git clone https://aur.archlinux.org/yay.git
  sh $ cd yay
  sh $ makepkg -si
  ```

[1]Reference: `https://www.linuxuprising.com/2019/03/how-to-install-oracle-java-12-jdk-12-in.html`.

Display all available Oracle JDK and JRE versions and select whichever you want.

```
sh $ sudo pacman -Syyu
sh $ yay jdk
```

- Other: use a proper search engine and share your findings such that we can update this guide.

2.1.3 Mac OS

Download Oracle JDK from `https://www.oracle.com/technetwork/java/javase/downloads/index.html` and run the installer.

2.2 Java package creation

In this part, you need to create a simple Java command line program. Different from C++, Java programs are managed in packages, for example, for this lab, a typical package structure is as follows.

```
Current directory
 └─ com
     └─ ve472
         └─ l1
             └─ Main.java
```

Content of `Main.java`[2].

```java
package com.ve472.l1;

public class Main {

    public static void main(String[] args) {
        // write your code here
    }
}
```

2.3 Third-party package installation

The built-in support for third-party packages is one of the most fascinating features of Java compared to C++. The `org.apache.commons.cli` package is used in this lab to parse command line arguments. It is also the `cli` parser of *Hadoop*, which will be used in future labs.

There are two ways to install packages: direct install or through a package manager, such as `maven`. We recommend you to latter option.

2.3.1 Direct package installation

If you use common IDEs, e.g. Eclipse or Intellij Idea, you can manually add third-party libraries. First, download the binary file of the third-party libraries. Apache Commons CLI can be downloaded from `http:`

[2]Reference: `https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html`.

//commons.apache.org/proper/commons-cli/download_cli.cgi. Then add *.jar to your modules in the modules settings.

To include third-party libraries using the command line.

```sh
sh $ javac -classpath <path_to_libraries> <path_to_program>
```

Replace <path_to_libraries> with your own path to the third party libraries and <path_to_program> with your program. For more information refer to https://docs.oracle.com/javase/8/docs/technotes/tools/windows/classpath.html.

2.3.2 Maven package installation

Most IDEs, e.g Eclipse or Intellij Idea, officially support maven. Upon creating a maven project, a pom.xml file is generated. You can add dependencies and define the JDK version to use.

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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>ve472</groupId>
  <artifactId>ve472l1</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/commons-cli/commons-cli -->
    <dependency>
      <groupId>commons-cli</groupId>
      <artifactId>commons-cli</artifactId>
      <version>1.4</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.7.0</version>
        <configuration>
          <source>8</source>
          <target>8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

For more information on how to use maven from the command line refer to `https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html`.

3 Java programming

Now that is environment is ready we start solving our initial problem using a Java program.

3.1 Command Line Arguments

Import the `org.apache.commons.cli` package[3].

```java
import org.apache.commons.cli.*;
```

This package provides several useful classes such as `Options`, `CommandLineParser`, and `HelpFormatter`.

Task

Use them to write a command line program which takes two normal arguments (`--hall` and `--query`), as well as a help argument (`-h/--help`).

When one of the two arguments is missing, or `-h/--help` is used, output a help information generated by `HelpFormatter`. The format is

```
usage: cinema
 -h,--help          print this message
    --hall <arg>    path of the hall config directory
    --query <arg>   query of customer orders
```

3.2 Configuration files

The information about the halls in the cinema is stored in a configuration directory specified using the `--hall` flag (section 3.1). This directory contains several files but the their names are not fixed.

Write a class `Hall` which reads a cinema hall configuration files and saves all of the information as attributes. It is recommended to use a two-dimension `List<Boolean>` to store the seats.

Format of a cinema hall configuration file.

- Hall name: unique, on the first line;
- Movie name: not unique, on the second line;
- m boolean values on each of the following n lines: a location with a seat shows a 1 and a 0 otherwise; m is the maximum length over all the rows;
- Center line: between column $m/2$ and $m/2+1$, if m is even and on column $(m+1)/2$, otherwise; The column number starts at 1;

[3]Reference: `https://commons.apache.org/proper/commons-cli/`.

Task

Write a class `Cinema` which parses the configuration files in the directory, using the `--Hall` flag, and saves the information as attributes. It is recommended to use a `HashMap<String, Hall>` to store the mapping between hall names and `Hall`s, and a `HashMap<String, List<Hall>>` to store the mapping between movie names and `Hall`s.

Example

A valid configuration file looks as follows.Note that spaces are allowed in the hall and movie names.

```
apple
The Wandering Earth
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
```

3.3 Answer queries

The tickets purchase requests are stored in a query file defined using the command line argument `--query` (section 3.1). The format of each line composing the query file is

```
customer name,movie name,ticket number n
```

We assure $n > 0$, and no comma "," in the customer and movie names. However spaces are allowed. Omit all trailing spaces around a comma.

Task

Read the file line by line and answer each query to `stdout` using the following format:

```
customer name,movie name,hall name,row number,column number 1,...,column number n
```

First assign customers to halls, sorted in alphabetical order; start with the first hall. If no seats can be assigned, or the movie cannot be found, then only print the customer's name and movie.

Warning: do not print any unnecessary spaces, or some tests might fail on JOJ.

Example

For the input file,

```
Krystor He , The Wandering Earth, 3
Manuel,The Wandering Earth,20
Jing,unknown,1
```

The output should be

```
Krystor He,The Wandering Earth,apple,8,5,6,7
Manuel,The Wandering Earth
Jing,unknown
```

4 JOJ submissions

Compress your source code into a `tar` or `zip` archive following the structure below. Then submit it on JOJ (`https://joj.sjtu.edu.cn`).

```
└─ com
   └─ ve472
      └─ l1
         ├─ Cinema.java
         ├─ Hall.java
         └─ Main.java
```

Note: only the correctness of the implementation will be checked, i.e. its efficiency will not be evaluated.

A The C++ Hitchhiker's Guide to the Javanian Galaxy

Java is a compiled language, yet it runs on the Java Virtual Machine (JVM).

Java programs need to be compiled and run on a runtime called Java Runtime Environment (JRE). The Java development environment requires a package called Java Development Kit (JDK).

Java is fully object-oriented.

Java programs are organized in terms of classes. All variables or functions must reside within the scope of a class. All classes always inherit a common base class "Object".

The Java memory management features a garbage collector.

Object are automatically deleted and recycled when no longer needed. However, in some cases, you have to pay a significant performance price for that feature. Stay alert.

Java is statically and weakly typed.

Like C++, all Java variables are typed, and their type must be known at compile time. If necessary variables can be cast.

Java objects are always passed by reference.

Java contains a few built-in types that are passed by value. However all objects are passed by value-references. Think of them as pointers that automatically dereferences themselves.

Java is polymorphic by default.

By default, all methods in a class are “virtual”. Subclasses always override base class objects.

Java allows only single Inheritance and features Interfaces.

Only single inheritances are allowed. Interfaces in Java are similar to pure abstract base classes in C++.

Generics are done by type-erasure.

Java also supports Generic containers, similar to `std::vector<>` or `std::list<>` in C++. However, unlike C++, Java uses type-erasure to handle generics. In contrast, C++ uses type specialization.

Java supports reflection and introspection.

It is possible to ask a class to print out its supported methods, or look up a class by string, at runtime.

Have fun with Java.

Yep. Enjoy Coding!