University of Cyprus
Computer Science Department

Homework 1: RESTful API Java Client using the Jersey Framework or the Spring Boot Framework[1]

EPL425: Internet Technologies
Spring 2022

Announced Date: Tuesday, 08/02/2022
Submission Date: Tuesday, 01/03/2022 (16:00)

I. Introduction

The goal of this exercise is to write a **RESTful API client program in Java** through which you will obtain hands-on experience in HTTP request/response message exchange. Specifically, you are about to implement an HTTP message exchange functionality between your program and some RESTful APIs over the Internet. You will use the Jersey framework or the Spring Boot framework which are both open source, production quality, frameworks for developing RESTful Web Services in Java. It is strongly advised that your project is built using Apache Maven software which is project build and management tool. You can find examples of Maven-oriented Jersey-based RESTful API client and Maven-oriented Spring Boot-based RESTful API client in Lab5.

II. Goal

The goal of this assignment is to obtain, use and print:

(a) weather data via the OpenWeatherMap (https://openweathermap.org/) RESTful API

(b) routing data via the Graphhopper (https://www.graphhopper.com) RESTful Routing API

OpenWeatherMap provides a RESTful API service to provide current weather, daily forecast for 16 days, and 3-hourly forecast 5 days for any city on Earth by proving any of the following inputs: city name, city id[2], geographic coordinates or zip code. Graphhopper provides a RESTful Direction API which consists of 6 APIs. More specifically, we will use the Routing API which returns the best path(s) between two or more points.

In this assignment, you will combine weather forecast data from OpenWeatherMap API and navigation data between two points from Graphhopper Routing API to develop a web service.

---

[1] The given source code sample **as1-supplementary-jersey.zip** is based on the Jersey Framework while **as1-supplementary-spring-boot.zip** is based on the Spring Boot Framework (recommended solution).
[2] List of city ID city.list.json.gz can be downloaded here http://bulk.openweathermap.org/sample/

III. Prerequisites

To be able to carry out this assignment you will need to:

- Sign up for a free account with OpenWeatherMap. Once you register, you will receive an **APPID**, essentially your key for using the service. Without an APPID you will not be able to access any API endpoint. It takes up to 1 hour to activate your API key. You will receive a confirmation email as your API key is ready to work. The How to start page tells how to include the APIID and id in a request. Unless you care to convert from Kelvin to degrees Celsius in your code, be sure to request 'metric' units from the server.
- Sign up for a free account with Graphhopper. Once you register, you will receive an **API KEY**, which is required for using the service.

IV. Description of Work

Your java program will offer 4 different options according to given command-line arguments:

1. Return the current weather conditions for a given location
2. Return 5-days weather forecast for a given location
3. Return routing instructions for travelling by car from a given starting point to a given destination point
4. Return information about an excursion you plan to do within the next 5-days from a given starting point to a given destination, given the fact that you want to arrive at the destination when the highest temperature (throughout the whole duration) is reached.

Every option involves one or more requests to appropriate API endpoint(s) from OpenWeatherMap and/or Graphhopper APIs. The next section analyzes all API endpoints involved in this assignment. The requests from you API java client will advertise that the client accepts JSON data format. Upon response arrival, your client will parse the JSON obtained within the body of the response using Gson[3] json parser. In order to use GSON, place the following dependency in your pom.xml file:

```
<dependency>
   <groupId>com.google.code.gson</groupId>
   <artifactId>gson</artifactId>
   <version>2.8.9</version>
</dependency>
```

For the sake of further clarifying and guiding the development process, as1-supplementary-jersey.zip[4] and as1-supplementary-spring-boot.zip[5] files are given. The first zip includes a full Jersey-based Maven project and the second zip includes a full Spring Boot based Maven project (OpenWeatherMap folder) with a properly set pom.xml file (with all dependencies needed for the assignment[6]) as well as source code .java files which implement the first option (returns the

[3] Provides simple toJson() and fromJson() methods to convert Java objects to JSON and vice-versa.
[4] Developed on the basis of the Jersey Framework.
[5] Developed on the basis of the Spring Boot Framework.
[6] You can add mode dependencies in case you want to use any other libraries.

current weather conditions). You will have to **choose one framework** (we suggest Spring Boot which is a more popular Framework), study the given solution and build the rest options using the same approach. You may unzip the folder, get into OpenWeatherMap folder and run "mvn clean package". You can also import to Eclipse as shown in Lab5.

V. Involved API Endpoints

The API endpoints will be used in this assignment are shown below. The first 2 endpoints are related to OpenWeatherMap API and the third endpoint relates to Graphhopper API.

1. Current weather: http://api.openweathermap.org/data/2.5/weather

Example of API endpoint with parameters:

http://api.openweathermap.org/data/2.5/weather?q={city_name},{country_code}&units=metric&APPID={APPID}

Parameters:

- q : city_name and country_code divided by comma, use ISO 3166 country codes
- units : metric (temperature in Celsius, distance in kilometers), imperial (temperature in Fahrenheit, distance in miles). When you do not use units parameter, format is Standard (temperature in Kelvin, distance in kilometers) by default.

Example of API call that returns the current weather information for Nicosia,cy

http://api.openweathermap.org/data/2.5/weather?q=Nicosia,cy&units=metric&APPID={APPID}

The following request headers and reply headers and body were obtained using the curl command from command line in order to study the messages exchanged. For security reasons, the APPID was replaced by XXXXX from the information below. The curl command syntax is:

curl -v "http://api.openweathermap.org/data/2.5/weather?q=Nicosia,cy&units=metric&APPID=XXXXX"

Request

```
GET /data/2.5/weather?q=Nicosia,cy&units=metric&APPID=XXXXX HTTP/1.1
Host: api.openweathermap.org
User-Agent: curl/7.58.0
Accept: */*
```

Reply

```
HTTP/1.1 200 OK
Server: openresty
Date: Fri, 05 Feb 2021 07:14:14 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 462
```

```
Connection: keep-alive
X-Cache-Key: /data/2.5/weather?APPID=XXXXX&q=nicosia,cy&units=metric
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST

{"coord":{"lon":33.3667,"lat":35.1667},"weather":[{"id":801,"main":"Clouds","d
escription":"few
clouds","icon":"02d"}],"base":"stations","main":{"temp":16.53,"feels_like":14.
47,"temp_min":15,"temp_max":18.33,"pressure":1016,"humidity":72},"visibility":
10000,"wind":{"speed":3.6,"deg":280},"clouds":{"all":20},"dt":1612509092,"sys"
:{"type":1,"id":6370,"country":"CY","sunrise":1612500151,"sunset":1612538315},
"timezone":7200,"id":146268,"name":"Nicosia","cod":200}
```

The json obtained in 200 OK response body can be beautified using online json validators such as https://jsonlint.com/ as follows:

```
{
	"coord": {
		"lon": 33.3667,
		"lat": 35.1667
	},
	"weather": [{
		"id": 801,
		"main": "Clouds",
		"description": "few clouds",
		"icon": "02d"
	}],
	"base": "stations",
	"main": {
		"temp": 16.53,
		"feels_like": 14.47,
		"temp_min": 15,
		"temp_max": 18.33,
		"pressure": 1016,
		"humidity": 72
	},
	"visibility": 10000,
	"wind": {
		"speed": 3.6,
		"deg": 280
	},
	"clouds": {
		"all": 20
	},
	"dt": 1612509092,
	"sys": {
		"type": 1,
		"id": 6370,
		"country": "CY",
		"sunrise": 1612500151,
		"sunset": 1612538315
	},
	"timezone": 7200,
	"id": 146268,
	"name": "Nicosia",
```

```
      "cod": 200
}
```


2. [3-hourly 5-days Weather Forecast](): http://api.openweathermap.org/data/2.5/forecast

Example of API endpoint with parameters:

http://api.openweathermap.org/data/2.5/forecast?q={city_ID},{country_code}&units=metric&APPID={APPID}

Parameters:

- q : city_name and country_code divided by comma, use ISO 3166 country codes
- units : metric (temperature in Celsius, distance in kilometers), imperial (temperature in Fahrenheit, distance in miles). When you do not use units parameter, format is Standard (temperature in Kelvin, distance in kilometers) by default.

Example of API call that returns the 5-day historical weather information for Nicosia,cy

http://api.openweathermap.org/data/2.5/forecast?q=Nicosia,cy&units=metric&APPID={APPID}

The following request headers and reply headers and body were obtained using the curl command from command line in order to study the messages exchanged. The curl command syntax is:

curl -v "http://api.openweathermap.org/data/2.5/forecast?q=Nicosia,cy&units=metric&APPID=XXXXX"

Request

```
GET /data/2.5/forecast?q=Nicosia,cy&units=metric&APPID=XXXXX HTTP/1.1
Host: api.openweathermap.org
User-Agent: curl/7.58.0
Accept: */*
```

Reply

```
HTTP/1.1 200 OK
Server: openresty
Date: Sun, 04 Feb 2021 20:40:20 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 15227
Connection: keep-alive
X-Cache-Key: /data/2.5/forecast?APPID=XXXXX&q=nicosia,cy&units=metric
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST
```


```
{"cod":"200","message":0,"cnt":40,"list":[{"dt":1612472400,"main":{"temp":13.8
6,"feels_like":12.22,"temp_min":12.74,"temp_max":13.86,"pressure":1014,"sea_le
vel":1014,"grnd_level":997,"humidity":83,"temp_kf":1.12},"weather":[{"id":800,
"main":"Clear","description":"clear
```

```
sky","icon":"01n"}],"clouds":{"all":10},"wind":{"speed":2.82,"deg":319},"visib
ility":10000,"pop":0,"sys":{"pod":"n"},"dt_txt":"2021-02-04
21:00:00"},{"dt":1612483200,"main":{"temp":13.19,"feels_like":11.45,"temp_min"
:12.71,"temp_max":13.19,"pressure":1014,"sea_level":1014,"grnd_level":997,"hum
idity":84,"temp_kf":0.48},"weather":[{"id":800,"main":"Clear","description":"c
lear
sky","icon":"01n"}],"clouds":{"all":4},"wind":{"speed":2.77,"deg":317},"visibi
lity":10000,"pop":0,"sys":{"pod":"n"},"dt_txt":"2021-02-05
00:00:00"},...],"city":{"id":146268,"name":"Nicosia","coord":{"lat":35.1667,"l
on":33.3667},"country":"CY","population":200452,"timezone":7200,"sunrise":1612
413801,"sunset":1612451853}}
```


From the body of 40 (see cnt value in json) 3-hourly forecasts above, we show only the first two forecasts (highlighted in yellow and blue color). In order to better preview the json returned from the weather forecast API endpoint you may use the json validator https://jsonlint.com/.

3. Finding the best paths between two or more points: https://graphhopper.com/api/1/route

Example of API endpoint with parameters:

https://graphhopper.com/api/1/route?point={lat},{lon}&point={lat},{lon}&vehicle={type}&key={API_KEY}

Parameter	Default	Description
point	-	Specify multiple points for which the route should be calculated. The order "latitude,longitude" is important. Specify at least two points. The maximum number depends on the selected package.
locale	en	The locale of the resulting turn instructions. E.g. `el_GR` for Greek or `de` for German.
optimize	`false`	Normally, the calculated route will visit the points in the order you specified them. If you have more than two points, you can set this parameter to `"true"` and the points may be re-ordered to minimize the total travel time. Keep in mind that the limits on the number of locations of the Route Optimization API applies, and the request costs more credits..
instructions	`true`	If instruction should be calculated and returned

Parameter	Default	Description
vehicle	car	Enum: `"car"` `"bike"` `"foot"` `"hike"` `"mtb"` `"racingbike"` `"scooter"` `"truck"` `"small_truck"`. The vehicle profile for which the route should be calculated.
elevation	`false`	If `true` a third dimension - the elevation - is included in the polyline or in the GeoJson. IMPORTANT: If enabled you have to use a modified version of the decoding method or set points_encoded to `false`. See the points_encoded attribute for more details. Additionally a request can fail if the vehicle does not support elevation. See the features object for every vehicle.
points_encoded	true	If `false` the coordinates in `point` and `snapped_waypoints` are returned as array using the order [lon,lat,elevation] for every point. If `true` the coordinates will be encoded as string leading to less bandwith usage. You'll need a special handling for the decoding of this string on the client-side. We provide open source code code in Java and JavaScript, see the clients section. It is especially important to use our official client or code if you set `elevation=true`!
calc_points	`true`	If the points for the route should be calculated at all printing out only distance and time.
debug	`false`	If true, the output will be formated.
type	`json`	Specifies the resulting format of the route, for `json` the content type will be application/json. Or use `gpx`, the content type will be application/gpx+xml, see below for more parameters.
point_hint	-	Optional parameter. Specifies a hint for each `point` parameter to prefer a certain street for the closest location lookup. E.g. if there is an address or house with two or more neighboring streets you can control for which street the closest location is looked up.

Parameter	Default	Description
details	-	Optional parameter to retrieve path details. You can request additional details for the route: `street_name` and `time`. For all motor vehicles we additionally support `max_speed`, `toll` (no, all, hgv), `road_class` (motorway, primary, ...), `road_environment`, and `surface`. The returned format for one details is `[fromRef, toRef, value]`. The `ref` references the points of the response. Multiple details are possible via multiple key value pairs `details=time&details=toll`.

Example of API endpoint that returns the routing information for travelling from Nicosia,cy to Paphos,cy by car

https://graphhopper.com/api/1/route?point=35.1667,33.3667&point=34.7667,32.4167&vehicle=car&key={API_KEY}

The following request headers and reply headers and body were obtained using the curl command from command line in order to study the messages exchanged. The curl command syntax is:

curl -v "https://graphhopper.com/api/1/route?point=35.1667,33.3667&point=34.7667,32.4167&vehicle=car&key=XXXXX"

Request

```
GET
/api/1/route?point=35.1667,33.3667&point=34.7667,32.4167&vehicle=car&key=XXXXX
HTTP/1.1
Host: graphhopper.com
User-Agent: curl/7.58.0
Accept: */*
```

Reply

```
HTTP/2 200
server: nginx
date: Thu, 04 Feb 2021 21:00:48 GMT
content-type: application/json;charset=utf-8
content-length: 6410
access-control-expose-headers:       X-RateLimit-Limit,X-RateLimit-Remaining,X-
RateLimit-Reset,X-RateLimit-Credits
access-control-allow-origin: *
x-ratelimit-limit: 500
x-ratelimit-remaining: 500
x-ratelimit-reset: 16894
x-ratelimit-credits: 1
strict-transport-security: max-age=31536000; includeSubDomains
```

{"hints":{"visited_nodes.average":"172.0","visited_nodes.sum":"172"},"info":{"copyrights":["GraphHopper","OpenStreetMap contributors"],"took":11},"paths":[{"distance":149564.451,"weight":8515.483725,"time":6271857,"transfers":0,"points_encoded":true,"bbox":[32.416931,34.667021,33.411997,35.166632],"points":"mnsuE_}sjEDr@xInCZBb@rGZb@VB~D}Cx@u@^a@xDuGPa@vAuDl@kBb@qBP_APmAj@wGPcBh@qA^c@f@[~Ai@`DmAlC{@fAm@zAq@dAy@zDeDf@_@t@a@x@WpAKtA@bN`C~DZ~Jf@`D?zGJ~AHjHl@d@@nAEfH{@bEo@|Bi@dAQbCKhDCjEK|A@lAF|ANzCd@~Ab@pC`Az@\\xAv@`BhApAbAtAnAvB~BbBbCx@xAfCbFhBfElB`EnBhDbAvA|@fA~@bAjAfA`BrArA`AlEdCn@XzDpA`Cl@|Ez@vALxCHrE?dCO~CYnCi@~Ae@`c@{M|OoFtl@aQbkAm^vJuCjJ_ChHmA|De@lNy@jFA|\\DhDErMNvICxBMbCY~Ba@lBe@`DgApDcBjBgAzEoDfKwIbCiBpGaEvDqBbFqBtCcAhCy@fDy@zBa@zEu@~BU~Fc@vAG`DEnD@|CHnJr@|nBxWzDf@jHn@fKh@hOAdDItEWpFe@jFo@hDk@pJsBlIcC|JsDzCsA~F{CpGaElJ{GfBsAnBgB`H_HrFqGbCcD|BcDzJ_PnL_RzD}FnFwGpEgEhE_Dp_A_h@zGeD~DuAdE_AvAQxCQd{@mAtKQlKW~DJvDd@lCd@|CbA`DvA`HdDtD|AtDjAbEt@fEVnE@|BO~B[bB_@nV{GlJmBtCa@|BWbFc@nGShE?hGNrHf@lKtAvqC|`@lNnCrIxBpOnF|i@zRtE|AlFjAhEVhDC`D]lH}A~H}AfDUzDBfEh@nF`BnNvExLrDlHrAlIx@vcCvLxGEhHw@hEcA`DiAbCyAzEcDnDqDrBeChBuCbN{Uz@gAnBmBtCmBpDsAlCi@xBU`B@nADbAJ~AXjC~@fCrA~AfAnAtAzCpEn@bBfAzDbLjb@dMxe@zAvFx@`CvAvDpDfIhBjDvBtDlDdF~CzDtC|C|HpHtPnOzDjE|F`IvAfCxBhEfb@r`Ad@lApM`ZnBtEvArCjCrEhChD|DxDzEnDpEzBnBp@lBh@~Cr@bEd@~DPvPf@vDPrDZdCb@tElAfCv@jA`@fJhEtOfHnG`B~Cb@~DZhEBhHSnXc@`Eb@|Bh@xBr@lCrApBtAfB~A|KpMlE~DrHfE`KfElIvDzCjBxC|Bt[t]nXxZlD|EjE`HxBdErBhEzBlF|BtGrErOpHfWpRlq@l[ngA~AjE|@vBdArBnAnBpA`BrDfErBfBbDxB~BnAnEjBnUlIbFpBvEbC`DpBjBzA`CxBfBjBlDlEzXxc@pIvMtKbQ~AvCdA`C`AhCzA`Gx@pFXhDP`F?rEKdDWdD[tCi@zC_AzDaMxe@mAjEoHtYsC|IaDnIgZzu@y@zBg@dBcAvDk@bCo@tD[~B]nDSrCoClm@GjD@hDL`D\\jEp@xE|@vDhAxDzAnDfB|CjAhB~BpCxBpB|BbBhDlBxEfB~IpDpAp@pAx@xAlApApA|@fAbA`Br@pAz@vBVv@r@nCXzARbBThDF|CCfQ?fF@dCHbC`@xExS`zAvCrRdGfc@d@|DVrCXdEL~CJ~DBdGGrFQvF[`F]vDeA`I}EtWc@pCc@nDWhDQzCMjEChFHzEPpETfDZ~Cl@bExDjTbDdQ`DlOdEnQvDpNbIbXnAtElAjFlAdGdAvGl@xEjEvb@fCrUdArKjAhKbDv[h@hGZ`GHxCDvGAvFGpFMtBe@xGa@tDg@bEi@lDk@zCqM|n@[fCKrAKjBCtCBzBHlBrQrrBjB~TTrBhBrTh@tIR|FTbL@tEQh`@A~GJ`Mh@~RdA~b@PfDVlCh@xCl@tCrAjE`BpDvA`CpAjBzBhC`F~EzLhLfq@lp@fApAl@~@v@zAr@fBj@pBVvA`Ep\\tA|N\\zGVpHFlE?rDKjHSvGWjEa@bGm@~Fs@zFcA~Fy@lEmAdFmHdYaHnXeGlU}H|ZwDxNu@bDsAnHc@bDY|CQpCObDIlD?~DBhDPpFVzD\\bDp@vE^zB`A`EdA|D~BxGbAzBpB|DrBjDxBzCbDdEzAvB`BlCv@dBp@jBfAtEXlBTzBJ|BFtDq@vr@QjWBfCNdCNlBrA|GzJf^xBnI~B~JjDrPhDxPbM`q@n@nCxBvNl@nFj@zHHdBLtDHvDAxHUrJYrE_@rEe@vDg@tDyB`L{BzHyBxG_BzDgBxD{GfLyIrL{P|TaA~A{@~AiB|EqC|HyBdHgAdCkAtBcAtAoAvAwAlAcBlA_L`GmAf@gFhCaDvByCpCiB|BoAjBy@vAs@`BoA`Do@dC_@fBc@tCSlCQnCAjD@zBRlD~@dGbAdEnArDpE`LxApE~DnJdQbe@pA~D~A|H|@rHZ`EJ`CL~ERxFLtARtAr@fDVbAj@~Ar@fBvAjCjF~GjElGvAdCp@|Al@jBt@rCb@fC`@~DpCb[NpARvAVnAbAbEb@xAn@~AhBlDpAdBnArAz@r@bAr@nDlB`AZvFrApFxApC|@pB`A`BfArAlAhAlAlBbCj@~@j@rAv@pBf@~Af@xBp@rEJ`AH~ABjBAlBGnBMjBShB]lBe@pBe@~Au@jBw@`BgAfBmA~A}@bA_A|@q@h@}CrBeHjEi@`@{AvAyA~A{@jAu@nAy@bBq@jBo@`Cc@jC[xBQbCC~@EbC@|BDfAFx@VhCTtAfCvLnF~YpFtY\\jB^vC^jDP|DBtBC|BMzDUjDmEzi@WpEiDxc@aAlLUfEO~DMnGCbDNlh@BzARnHNtDJvB\\zElFng@n@jHHbGIvFQtDgGxn@KzBAzADdCPlCLtAThBnAxHJjAFnEEzCs@lLEpCJfCTxCh@lFfAhIl@zCtBrJdDrKnAbDfBpFpArEz@hG\\vEFpBGbn@HrET`E\\dDb@|Cr@dDr@jClBfFrAtCxGvLjDpG`BpDjBtFv@rCz@pEp@vER|Br@pKZnCXdBl@rCf@bBp@bBdAxBz@pAjKpMlAjBhBlDp@zAbBxEpA|EhA|Fb@zDXrEJ|K_@~J[zDc@tD_AxE{AzEyArC{A|B}@nAsCvC_BxB}AfCeBjDwBxFi@fBUdAWfBO`BKnBBlBPlCP~Af@hDl@dDdAzDvAnDzFrLfAnCn@nBr@vCd@rC^fDPhCFlBA`DSlFSnCYhC}@zEa@~A_ArCiAjCiA|BuBbDwCtDcMvPcAbBgAnBwGtOaEjIeRx[qC|FiNl\\iBfEkDnJ}Mhb@oA`DsAbDgE~IuIjNgDxEgArAwF`G_DzCkZhXuJhIiEtDcEfEmC|CwGxI}C`F{E|I_BnDsBdF{BdGw`AvqC}A~DqBxDoBlCkCpCmB|AkAv@oC|AcBn@iCz@_NpDyDhAcG`BsLxDcGfCkGfDkDtB}DlCuEtDmD~CyF~FgBtBwC|DeEdGyD~GkCnFqClGaEjL{BhImAxFqApGuKxr@w@dEw@jDm@tBu@~Bm@dBoA`DaBjDqBjDmAbBwA~AqB`B_Af@yAh@sBh@yAXmCRyDCqDUiEGuCFiCRiC`@aB\\eBd@iDrAmDjB_C|AeBrAeB~AeDdEyCdEsK~MiAhBeAnB{@nBeAzCy@xCy@tEi@rEU|CAvD@|BHbDT`FBnDCjFWnF]xDc@bDa@bCy@tDw@rCeAxC_A~Bw@`BgAvBoBzC}b@zk@eU|ZkAnB{@dBw@vBi@dBm@fDYzBSpCIzC@rE?THl@T`AL\\D\\?rBApAYhAIJaEjFV`CPr@d@hAnC`Ft@dBj@zA|@dDjAvGz@vFHt@Bl@VvGXpFHp@P~@pApE@zAG|AaAhO[xAATn@tE@`B`AXnAd@~CxArAf@rBtALB~@l@zCtBh@b@f@p@~@jBNTb@d@XLTF~@?\\GFXN`@LP`AlBp@zC`@tAb@dAzAp@nEzBhAd@zAt@t@RjAKJGXWv@uAd@m@^UNCv@GbJYbHDEzAvBHPJBF?Na@lBkAbE","instructions":[{"distance":23.789,"heading":254.97,"sign":0,"interval":[0,1],"text":"Continue onto Pindarou","time":3058,"street_name":"Pindarou"},{"distance":218.734,"sign":-

```
2,"interval":[1,3],"text":"Turn left onto Androkleous Street","time":49214,"street_name":"Androkleous Street"},{"distance":127.402,"sign":2,"interval":[3,4],"text":"Turn right onto Digeni Akrita Avenue","time":12739,"street_name":"Digeni Akrita Avenue"},{"distance":35.044,"sign":-7,"interval":[4,6],"text":"Keep left","time":3942,"street_name":""},{"distance":18958.484,"sign":-1,"interval":[6,141],"text":"Turn slight left onto Archbishop Makarios III Avenue, B1","time":944584,"street_name":"Archbishop Makarios III Avenue, B1"},{"distance":126382.421,"sign":7,"interval":[141,964],"text":"Keep right onto Nicosia-Limassol Highway, A1","time":4807520,"street_name":"Nicosia-Limassol Highway, A1"},{"distance":1674.303,"sign":-2,"interval":[964,987],"text":"Turn left onto Eleutheriou Venizelou Avenue, E710","time":188346,"street_name":"Eleutheriou Venizelou Avenue, E710"},{"distance":704.723,"sign":-2,"interval":[987,1004],"text":"Turn left onto Evagora Pallikaridi, B7","time":74852,"street_name":"Evagora Pallikaridi, B7"},{"distance":262.599,"sign":2,"interval":[1004,1011],"text":"Turn right onto Gladstonos, B20","time":27152,"street_name":"Gladstonos, B20"},{"distance":321.615,"sign":-1,"interval":[1011,1016],"text":"Turn slight left onto Apostolou Pavlou, B20","time":32161,"street_name":"Apostolou Pavlou, B20"},{"distance":68.077,"sign":-7,"interval":[1016,1019],"text":"Keep left","time":7658,"street_name":""},{"distance":502.177,"sign":-1,"interval":[1019,1026],"text":"Turn slight left onto Αγαπήνωρος","time":56490,"street_name":"Αγαπήνωρος"},{"distance":42.207,"sign":2,"interval":[1026,1027],"text":"Turn right","time":9496,"street_name":""},{"distance":242.876,"sign":-2,"interval":[1027,1033],"text":"Turn left onto Iliados","time":54645,"street_name":"Iliados"* Connection #0 to host graphhopper.com left intact

},{"distance":0.0,"sign":4,"last_heading":295.73789553472574,"interval":[1033,1033],"text":"Arrive at destination","time":0,"street_name":""}],"legs":[],"details":{},"ascend":1889.9599685668945,"descend":2007.3599681854248,"snapped_waypoints":"mnsuE_}sjEpbmAdoxD"}]}
```


The output format and values of the above API endpoint is described here. For example, distance is measured in meters and time (duration) of the route in milliseconds.

VI. Examples of Execution

In this section, we provide execution examples for the 4 program options. Your program must return the same output for each option as the examples shown below. Please make sure that you print appropriate error messages: (a) if the number of arguments is invalid, (b) if the latitude (-90 to +90) and longitude (-180 to +180) values are within the limits (related to option 3) and (c) if the name of the location is invalid. The latter problem will emerge when you send a request to OpenWeatherMap and you receive a 404 Not Found response. Error messages will be taken into account in the assignment grade.

OPTION 1: The output of the first option is the beautified value of the section “weather” (the first element of the list) and “main” of the json response as shown below. The source code for that option is given. Thus, you do not have to re-implement it but you have to incorporate it in your Maven project.

```
java -cp <Classpath> <packages>.<class> -currentWeather Nicosia,cy
{
  "id": 521,
  "main": "Rain",
  "description": "shower rain",
  "icon": "09d"
}
{
  "temp": 11.52,
  "pressure": 1009,
  "humidity": 87,
  "temp_min": 11.0,
  "temp_max": 12.0
}
```

OPTION 2: The output of the second option is the beautified value of the first element of the forecast list within the “list” section of the json response as shown below:

```
java -cp <Classpath> <packages>.<class> -weatherForecast Nicosia,cy

{
  "dt": 1550145600,
  "main": {
    "temp": 14.36,
    "temp_min": 12.44,
    "temp_max": 14.36,
    "pressure": 1013.41,
    "sea_level": 1022.54,
    "grnd_level": 1013.41,
    "humidity": 100,
    "temp_kf": 1.92
  },
    "weather": [{
    "id": 501,
    "main": "Rain",
    "description": "moderate rain",
    "icon": "10d"
    }],
    "clouds": {
      "all": 80
    },
    "wind": {
      "speed": 1.27,
      "deg": 285.006
    },
    "rain": {
      "3h": 3.4375
    },
    "sys": {
      "pod": "d"
    },
    "dt_txt": "2021-02-05 12:00:00"
}
```

OPTION 3: The output of the third option is the beautified value of the first two elements of the instructions list within the "instructions" section of the json response as shown below. The four command line arguments after the -routing option are: source latitude, source longitude, destination latitude, and destination longitude respectively.

```
java -cp <Classpath> <packages>.<class> -routing 35.1667 33.3667 34.7667 32.4167
```


```
{
  "distance": 23.789,
  "heading": 254.97,
  "sign": 0,
  "interval": [0, 1],
  "text": "Continue onto Pindarou",
  "time": 3058,
  "street_name": "Pindarou"
}
{
  "distance": 220.244,
  "sign": -2,
  "interval": [1, 3],
  "text": "Turn left onto Androkleous Street",
  "time": 49554,
  "street_name": "Androkleous Street"
}
```


OPTION 4: The output of the fourth option is information about an excursion you plan to do within the next 5-days from a given starting point to a given destination, given the fact that you want to arrive at the destination when the highest temperature (throughout the whole duration) is reached. When searching for the highest temperature in the next 5 days, use the "temp" field in "main" section of the OpenStreetMap response, and the dt_txt field. The geographic coordinations (latitude, longitude) of both locations will be found on weather forecast responses.

```
java -cp <Classpath> <packages>.<class> -excursion -from Nicosia,cy -to
Paphos,cy

Highest temperature at destination is 15.69 on 2021-02-19 12:00:00
Trip distance: 149.581 km
Time to reach destination: 104.49 minutes
```

Important Notes for OpenWeatherMap API

- In the free plan, there is a limit of 60 requests per minute. If you do not receive a response from the API, please, wait at least for 10 min and then repeat your request.
- The server name is **api.openweathermap.org**. Please, never use the IP address of the server.

Important Notes for Graphhopper Routing API

- In the free plan, there is a limit of 500 requests per day and up to 5 locations and 1 vehicle per request. The latter restriction is out of the scope of this assignment, since you will use the routing API with two locations only (start, stop) for one vehicle.

VII. Submission

Run your program three times, one for obtaining the weather forecast, one for obtaining the routing information and one for running scenario of option 4 and put all the commands along with the results (output) of your program in a word document.

Create a .zip file that will contain:

1. your whole Maven project (the folder created by Maven). Before you zip please run the command "mvn clean" to remove target folder containing large .jar files and minimize Maven project folder size.
2. your document as described above

and submit it to Moodle.

APPENDIX

For Jersey implementation

In rain section in OpenStreetMap weather forecast, there is a json string 3h starting with number. We cannot define a variable name starting with number so we can use annotations:

```
import com.google.gson.annotations.SerializedName;

public class Rain {

    @SerializedName(value = "3h")
    private float threeHours;
```