Bonial Technical Challenge

There are some restaurants in the city. The public authority of this city decided to make tourists' and residents' lives easier and create an application to help them find the closest restaurant. Every place has its circle, denoted by its radius, from where people might go to this place.

The development of this application was given to an IT company, your task is to develop the API for it.

First endpoint

Restaurants are placed on the first quadrant of the Cartesian plane, their coordinates are non-negative integers, e.g., `x = 4, y = 5`. A restaurant's radius is a positive integer, e.g., `r = 2`.

You have to create several endpoints **using Java and Spring Boot**. The first endpoint should return restaurants that are visible from the given user coordinates, i.e. the ones, whose distance to the user is not greater than their radius.

For instance, if there are these restaurants in the city:
#1: `(1, 1), radius 1`,
#2: `(2, 2), radius 2`,
#3: `(5, 5), radius 1`,
#4: `(2, 3), radius 5`,
and the user coordinates are `(3, 2)`, then items #2 and #4 should be returned (see the image below).

The endpoint should accept a GET request like this:

```
GET /locations/search?x=3&y=2
```

Where `x` and `y` represent the user location.

A restaurant search view consists of:

- ID (UUID),
- Restaurant name,
- Coordinates,
- Distance from the user (calculated).

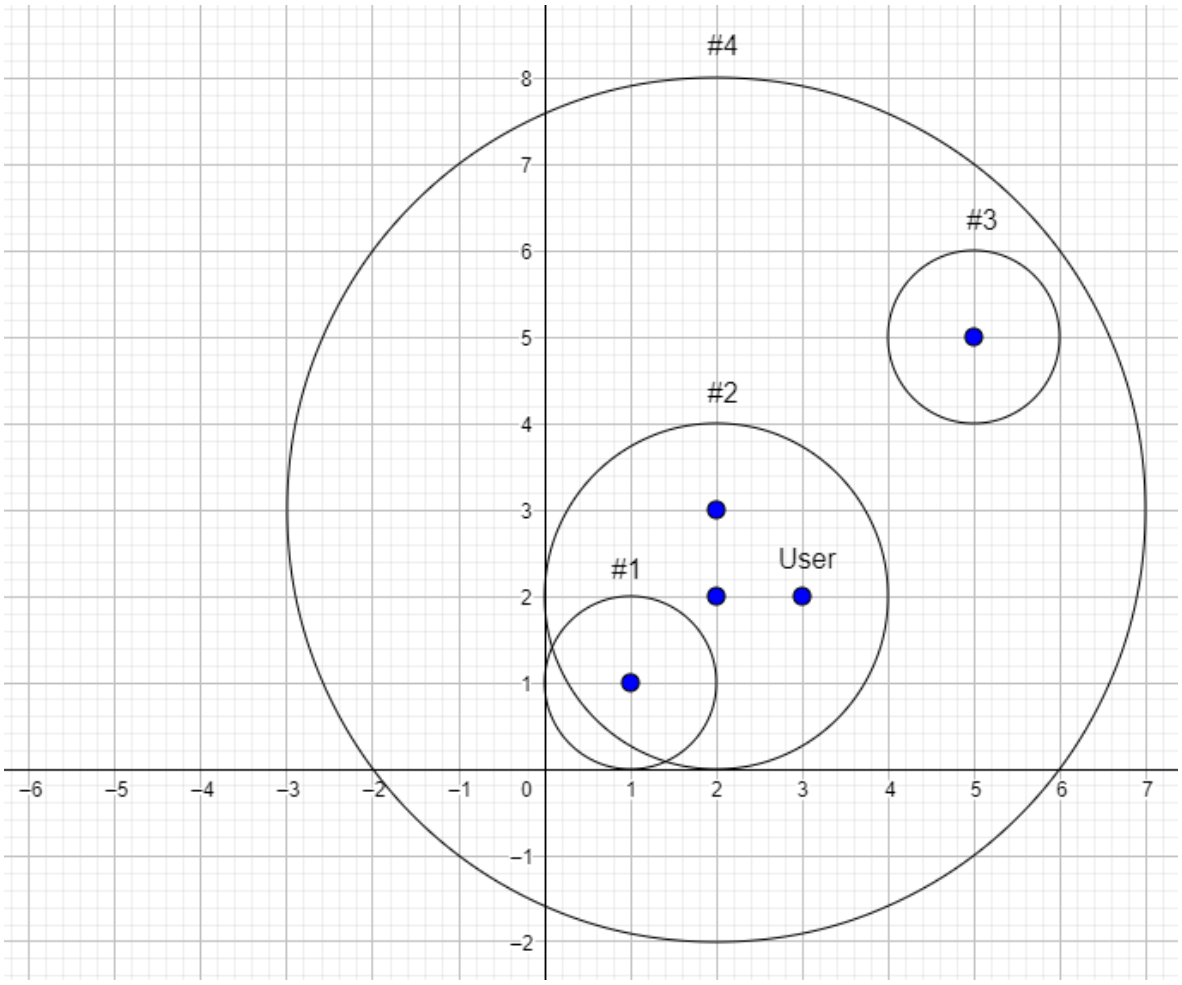
The endpoint should return the user coordinates and a list of restaurants with their distances to the user.

Example response:

```
{
```

```
 "user-location": "x=3,y=2",
 "locations": [
   {
     "id": "21e1545c-8b65-4d83-82f9-7fcad4a23114",
     "name": "Deseado Steakhaus",
     "coordinates": "x=2,y=2",
     "distance": 1.0
   },
   {
     "id": "20e1545c-8b65-4d83-82f9-7fcad4a23114",
     "name": "Fire Tiger",
     "coordinates": "x=2,y=3",
     "distance": 1.41421
   }
 ]
}
```

The locations in the response **should** be sorted by distance.



Second endpoint

The second endpoint should provide restaurant details. For the given restaurant ID, the endpoint returns more detailed restaurant information as in the given JSON format.

```
GET /locations/51e1545c-8b65-4d83-82f9-7fcad4a23111

Response:
{
 "name": "Da Jia Le",
 "type": "Restaurant",
 "id": "51e1545c-8b65-4d83-82f9-7fcad4a23111",
 "opening-hours": "10:00AM-11:00PM",
 "image": "https://tinyurl.com",
 "coordinates": "x=5,y=5"
}
```

Limitations

The restaurant coordinates are always non-negative, although they might not always be as small as 1, 2 or 5. There might be lots of data points (thousands, …) with much bigger values (up to millions).
The same applies to the user coordinates, but radii won't be too big not to return lots of items at once.

Data

The locations data will be provided in JSON format in a separate file. The choice of the datasource is yours, although note that it will also be evaluated.

It would also be **nice-to-have** an endpoint that is able to add restaurants dynamically:

```
PUT /locations/51e1545c-8b65-4d83-82f9-7fcad4a23111
{
 "name": "Da Jia Le",
 "type": "Restaurant",
 "id": "51e1545c-8b65-4d83-82f9-7fcad4a23111",
 "opening-hours": "10:00AM-11:00PM",
 "image": "https://tinyurl.com",
 "coordinates": "x=5,y=5",
 "radius": 1
}
```

Acceptance criteria

Your application **SHOULD**:

- Be implemented in Java and built on top of the Spring Boot framework.
- Implement the `GET /locations/search` endpoint. The locations in the response **should** be sorted by distance, ascending.
- Implement the `GET /locations/{id}` endpoint.
- Be able to read and process locations from a provided JSON file.
- Have some tests.

It would be **NICE**:

- To provide instructions on how to run the application in readme.
- To have an endpoint `PUT /locations/{id}` to dynamically add locations.
- To have proper error handling and reporting along with proper HTTP codes returned from your endpoints.
- To consider a high amount of data and/or load on your API and prepare accordingly.
- To provide a brief rationale explaining your technical decisions (e.g. choice of the tech stack, choice of algorithms) in the readme file.

You will be provided with a sample data JSON, although we may test your solution against bigger datasets.