Observables Exercise

0. Start by creating a new application → ng new observableTraining. Then run the command 'ng serve -o' to open the application.

1. HTTP Get Request from Employee Service
    a. Add HttpClientModule into your app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

    b. Generate an employee.service.ts under the app folder → ng g s employee
    c. Make you that you have included EmployeeService in the providers in app.module.ts

```
import { EmployeeService } from './employee.service';
[..]
providers: [EmployeeService],
[..]
```

    d. In the EmployeeService constructor, type 'private http: HttpClient'. This allows us to make a get request to fetch data using http.
    e. Place the employees.json file into src>app>assets folder. Make a folder called assets if you do not have one.
    f. Create a getEmployees() function to retrieve the data from employees.json using the 'http' declaration you have made earlier.

```
import { HttpClient } from '@angular/common/http';
```

```
import { Injectable } from '@angular/core';
@Injectable({
  providedIn: 'root'
})
export class EmployeeService {

  private urlData = 'assets/employees.json';

  constructor(private http: HttpClient) { }

  getEmployees(): any{
    return this.http.get(this.urlData);
  }
}
```

2. Receive the Observable and cast it into an employee array
    a. If you hover over the this.http.get() method, you can see that it returns an Observable.
    b. Create an employee.model.ts interface to represent the employee data.

```
export interface EmployeeModel{
    id: number;
    name: string;
    age: number;
}
```

    c. Add the EmployeeModel type to the http.get request. getEmployees() will need to also return an observable of type EmployeeModel array.

```
  getEmployees(): Observable<EmployeeModel[]>{
    return this.http.get<EmployeeModel[]>(this.urlData);
  }
```

3. Subscribe to the observable & assign the employee array to a local variable
    a. Create a new EmployeeList Component under src> app → ng g c employeeList
    b. Subscribe to the observable you have created in EmployeeService to receive data.
    c. Once you have subscribed, you will receive the data and then assign it to a local variable

```
import { Component, OnInit } from '@angular/core';
import { EmployeeService } from '../employee.service';
@Component({
  selector: 'app-employee-list',
  templateUrl: './employee-list.component.html',
  styleUrls: ['./employee-list.component.scss']
```

```
})
export class EmployeeListComponent implements OnInit {

  public employees = [];
  constructor(private employeeService: EmployeeService) { }

  ngOnInit(): void {
    this.employeeService.getEmployees().subscribe(
      data => {
        this.employees = data;
      }
    );
  }
}
```

d. Display the data in employee-list.component.html

```
<h3>Employee Details</h3>
<div *ngFor="let employee of employees">
  <p>Name: {{ employee.name }} ; Age: {{ employee.age }}</p>
</div>
```

e. Remember to display the EmployeeList Component in app.component.html

```
<div style="text-align: center">
  <h1>Welcome to Observables Training 2020!</h1>
  <br />
</div>
<app-employee-list></app-employee-list>
```

Results

Welcome to Observables Training 2020!

Employee Details

Name: Ang Chee Guan ; Age: 18

Name: Chow Choong Hoe ; Age: 18

Name: Irman Jamil ; Age: 18

Name: Kerk Hui Hong ; Age: 18

Name: Lim Teck Chuan ; Age: 18

Name: Yim Kient Tat ; Age: 18