Cairo University

Faculty of Computers and Information

Software testing

Assignment#2

Assignment Team

Name	ID
Marwa Ahmed Darwish	20185003
Marwa Mohamed Ahmed	20186028
Abdulrahman Mohamed Ahmed	20186013

Class:WeatherService
Modeling the Input Domain WeatherService

Step 1: Identify testable functions with parameter.

- `getCurrentWeather(String cityName)`
- `getCurrentWeather(int cityId)`
- `getCurrentWeather(double latitude, double longitude)`

Step 2: Model the input domain.

A) getcurrentweather(string cityname)

Interface-Based Approach

Characteristic:

1) weather information of cityname.
   Partitions: non-empty, empty.

Characteristics Table:

Method	Parameter	Return	value	Characteristics	Ch.ID
getcurrentweather	String cityname	String	String, null	weather information of cityname	C1

Partition Table:

Characteristics	b1	b2
C1	Non-empty	empty

B) getcurrentweather(int cityId)

Interface-Based Approach

Characteristic:

1) weather information of cityid.
   Partitions: cityid=0,cityid>0,cityid<0.

Characteristics Table:

Method	Parameter	Return	value	Characteristics	Ch.ID
getcurrentweather	Int cityid	String	String, null	weather information of cityname	C1

Partition Table:

Characteristics	b1	b2	b3
C1	cityid=0	cityid>0	cityid<0

C) getcurrentWeather(double lat,double lon) Method

Interface-Based Approach

Characteristic:

1) lat property,
   Partitions: valid, invalid
   Note: valid if the boundry of lat is -90° <=lat at south and 90°>=lat at nourth otherwise is invalid.

2) lon property,
   Partitions: valid, invalid
   Note: valid if the boundry of lon -180°<=lon at west and 180°>=lon at east otherwise is invalid.

Functionality-Based Approach

Characteristic:

3) weather information of coordinate.
   Partitions: valid, invalid.
   Note: valid if the boundry of lon and lat is -90° <=lat at south and 90°>=lat at nourth , -180°<=lon at west and 180°>=lon at east otherwise is invalid.

Characteristics Table:

Method	Parameter	Return	value	Characteristics	Ch.ID	Coverd by
getcurrentweather	double lat,double lon	String	String, null	1) lat property 2)lon property 3)weather information of coordinate	C1 C2	 C1,C2

Partition Table:

Characteristics	b1	b2
C1	valid	invalid
C2	valid	invalid

Step 4: Apply a test criterion to choose combinations of values

We will choose Base choice criteria for getcurrentweather(string cityname) method

The path choice test is: Happy path -> **(non-empty)**.

Reason for choosing these criteria:

We have important scenarios based on knowledge domain and this is the most used scenarios so we must to test this scenarios to get the information.

Method	Characteristics	Test Requirement
getcurrentweather	C1	**(non-empty)**. empty

We will choose Base choice criteria for getcurrentweather(int cityId) method

The path choice test is: Happy path -> **(cityid>0)**.

Reason for choosing these criteria:

We have important scenarios based on knowledge domain and this is the most used scenarios so we must to test this scenarios to get the information.

Method	Characteristics	Test Requirement
getcurrentweather	C1	**(cityid>0)** cityid=0 cityid<0.

We will choose Base choice criteria for getcurrentWeather(double lat,double lon) Method method

The path choice test is: Happy path -> **(valid,valid).**

Reason for choosing these criteria:

We have important scenarios based on knowledge domain and this is the most used scenarios so we must to test this scenarios to get the information.

Method	Characteristics	Test Requirement
getcurrentweather	C1 C2	**(valid, valid)** (valid,invalid) (invalid,valid)

Test case for getcurrentweather(string cityname) method

1) String cityname="London"

This test case it covers blocks non-empty.

2) String cityname=" "

This test case it covers blocks of empty.

Test case for getcurrentweather(int cityId) method

1) Int cityid=26437343

This test case it covers blocks cityid>0.

2) Int cityid=0

This test case it covers blocks cityid=0.

3) Int cityid=-1

This test case it covers blocks cityid<0.

Test case for getcurrentWeather(double lat,double lon)

1) Lat=51.5085 , lon=-0.1257

This test case it covers blocks of valid,valid.

2) Lat=51.5085 , lon=0

This test case it covers blocks of valid,invalid.

3) Lat=0 , lon=-0.1257

This test case it covers blocks of invalid,valid.

Class: GsonService
Modeling the Input Domain GsonService

Step 1: Identify testable functions with parameter.

- `fromJson(String jsonString, Class<T> classOfT)`
- `toJson(Object object)`

Step 2: Model the input domain.

A) fromJson

Interface-Based Approach

Characteristic:

1) content of string.
Partition: emty, non-empty.

2) java object(return class).
Partition: valid, invalid.
Note: valid if the class is actually found.

Functionality-Based Approach

Characteristic:

3) from json to java object
Partition: valid, Invalid.

Characteristics Table:

Method	Parameter	Return	value	Characteristics	Ch.ID	Coverd by
fromjson	String jsonString, Class<T> classof	T	String, null	1)content of string. 2) java object 3)from json to java object	C1 C2	C1,C2

Partition Table:

Characteristics	b1	b2
C1	valid.	invalid
C 2	valid.	invalid

B) A) toJson

Interface-Based Approach

Characteristic:

1) from java object to json

Partitions: valid ,invalid.

Characteristics Table:

Method	Parameter	Return	value	Characteristics	Ch.ID
toJson	Object object	String	String, null	from java object to json	C1

Partition Table:

Characteristics	b1	b2
C1	valid	invalid

Step 4: Apply a test criterion to choose combinations of values

We will choose Base choice criteria for fromJson method

The path choice test is: Happy path -> **(non-empty, valid)**.

Reason for choosing these criteria:

We have important scenarios based on knowledge domain and this is the most used scenarios so we must to test this scenarios to get the information.

Method	Characteristics	Test Requirement
fromjson	C1 C2	**(non-empty, valid)** (non-empty, invalid) (empty, valid)

We will choose Base choice criteria for toJson method

The path choice test is: Happy path -> **(valid)**.

Reason for choosing these criteria:

We have important scenarios based on knowledge domain and this is the most used scenarios so we must to test this scenarios to get the information.

Method	Characteristics	Test Requirement
tojson	C1	**(valid)** invalid

Test case for fromjson `(String jsonString, Class<T> classOfT)` **method**

1) String jsonString=”{\"coord\":{\"lon\":0.1257,\"lat\":51.5085},\"weather\":[{\"id\":804,\"main\":\"Clouds\",\"description\":\"overcastclouds\",\"icon\":\"04d\"}],\"base\":\"stations\",\"main\":{\"temp\":293.76,\"feels_like\":293.23,\"temp_min\":290.36,\"temp_max\":295.21,\"pressure\":1013,\"humidity\":52,\"sea_level\":1013,\"grnd_level\":1010},\"visibility\":10000,\"wind\":{\"speed\":1.31,\"deg\":113,\"gust\":3.05},\"clouds\":{\"all\":96},\"dt\":1622663119,\"sys\":{\"type\":1,\"id\":1414,\"country\":\"GB\",\"sunrise\":1622605698,\"sunset\":1622664543},\"timezone\":3600,\"id\":2643743,\"name\":\"London\",\"cod\":200}”, Class<T> classofT=`ViewModel`

This test case it covers blocks (non-empty,valid)

2) String jsonString==”{\"coord\":{\"lon\":0.1257,\"lat\":51.5085},\"weather\":[{\"id\":804,\"main\":\"Clouds\",\"description\":\"overcastclouds\",\"icon\":\"04d\"}],\"base\":\"stations\",\"main\":{\"temp\":293.76,\"feels_like\":293.23,\"temp_min\":290.36,\"temp_max\":295.21,\"pressure\":1013,\"humidity\":52,\"sea_level\":1013,\"grnd_level\":1010},\"visibility\":10000,\"wind\":{\"speed\":1.31,\"deg\":113,\"gust\":3.05},\"clouds\":{\"all\":96},\"dt\":1622663119,\"sys\":{\"type\":1,\"id\":1414,\"country\":\"GB\",\"sunrise\":1622605698,\"sunset\":1622664543},\"timezone\":3600,\"id\":2643743,\"name\":\"London\",\"cod\":200}”, Class<T> classofT=WeatherService.class

This test case it covers blocks (non-empty,invalid)

3) String jsonString= `""` , Class<T> classofT=ViewModel. class

This test case it covers blocks (empty,valid)

Test case for tojson `(Object object)` **method**

1) String

```
jsonString="{\"coord\":{\"lon\":0.1257,\"lat\":51.5085},\"weather\":[{\"id\":804,\"main\":\"Clouds\",\"description\":\"overcastclouds\",\"icon\":\"04d\"}],\"base\":\"stations\",\"main\":{\"temp\":293.76,\"feels_like\":293.23,\"temp_min\":290.36,\"temp_max\":295.21,\"pressure\":1013,\"humidity\":52,\"sea_level\":1013,\"grnd_level\":1010},\"visibility\":10000,\"wind\":{\"speed\":1.31,\"deg\":113,\"gust\":3.05},\"clouds\":{\"all\":96},\"dt\":1622663119,\"sys\":{\"type\":1,\"id\":1414,\"country\":\"GB\",\"sunrise\":1622605698,\"sunset\":1622664543},\"timezone\":3600,\"id\":2643743,\"name\":\"London\",\"cod\":200}";
ViewModel vm=new ViewModel();
vm=notnull;
```

This test case it covers blocks (valid)

2)
```
ViewModel vm=new ViewModel();
vm=null;
```

This test case it covers blocks (invalid)

Class:ApplicationService

Test case for getcurrentweather(string cityname) method

1) String cityname="London"

This test case it covers blocks non-empty.

2) String cityname=" "

This test case it covers blocks of empty.

Test case for getcurrentweather(int cityId) method

1) Int cityid=26437343

This test case it covers blocks cityid>0.

2) Int cityid=0

This test case it covers blocks cityid=0.

3) Int cityid=-1

This test case it covers blocks cityid<0.

Test case for getcurrentWeather(double lat,double lon)

1) Lat=51.5085 , lon=-0.1257

This test case it covers blocks of valid,valid.

2) Lat=51.5085 , lon=0

This test case it covers blocks of valid,invalid.

3) Lat=0 , lon=-0.1257

This test case it covers blocks of invalid,valid.