まずは課題1を一緒に
やってみようと思います

プログラムを開く

/java-lecture/blob/master/src/jp/co/interfactory/lecture/Task1.java

を好きなテキストエディタで開く

```
10  /**
11   * 2019/07/01 北九州高専 プログラミング講座
12   *
13   * 【課題1】
14   * APIを使って好きな都市の3時間ごとの天気予報を出力してください。
15   */
16  public class Task1 {
17      @SuppressWarnings("unused")
        Run | Debug
18      public static void main(String[] args) throws Exception {
19          // 天気予報を取得するWebAPIを実行
20          Forecast forecast = new WeatherApiClient("037d75dd9c0210aa6fe2e5692fc278ea")
21                                  .hasShowResponseBody(true)
22                                  .get("forecast", Map.of("id", "5128638"), Forecast.class);
23
24          // ここから下にプログラムを書いていく
25
26
27      }
28  }
```

プログラムの説明

```
10  /**
11   * 2019/07/01 北九州高専 プログラミング講座
12   *
13   * 【課題1】
14   * APIを使って好きな都市の3時間ごとの天気予報を出力してください。
15   */
16  public class Task1 {
17      @SuppressWarnings("unused")
        Run | Debug
18      public static void main(String[ {
19          // 天気予報を取得するWebAPIを実行
20          Forecast forecast = new WeatherApiClient("037d75dd9c0210aa6fe2e5692fc278ea")
21                                  .hasShowResponseBody(true)
22                                  .get("forecast", Map.of("id", "5128638"), Forecast.class);
23
24          // ここから下にプログラム
25
26
27      }
28  }
```

WebAPIを実行するクラス

APIの認証キー

標準出力にWebAPIの
レスポンスを表示するか

都市のID
GitHubのページから好きな都市を
選んでIDを設定してください

GitHubのページに記載している都市以外がいい場合は、

https://github.com/if-matsumoto/java-lecture/tree/master/doc/city.list.json.gz

にIDの一覧がありますので、ファイルを解凍して好きな都市のIDを取得してください！

まずは実行してみる

Windows

```
> cd C:¥work¥java-lecture

> javac -encoding UTF-8 -sourcepath src -cp lib/* -d bin src¥jp¥co¥interfactory¥lecture¥Task1.java

> java -cp bin;lib/* jp.co.interfactory.lecture.Task1
```

Mac / Linux

```
$ cd C:¥work¥java-lecture

$ javac -encoding UTF-8 -sourcepath src -cp lib/* -d bin src/jp/co/interfactory/lecture/Task1.java

$ java -cp bin:lib/* jp.co.interfactory.lecture.Task1
```

WebAPIのレスポンスが出力されればOK

```
C:¥work¥java-lecture>java -cp bin;lib/* jp.co.interfactory.lecture.Task1
connect to http://api.openweathermap.org/data/2.5/forecast?appid=037d75dd9c0210aa6fe2e5692fc278ea&units=metric&id=1850147
Warning: Nashorn engine is planned to be removed from a future JDK release
{
    "cod": "200",
    "message": 0.011,
    "cnt": 40,
    "list": [
        {
            "dt": 1561820400,
            "main": {
                "temp": 21.02,
                "temp_min": 21.02,
```

レスポンスの確認


```
{
    "cod": "200",
    "message": 0.011,
    "cnt": 40,
    "list": [
        {
            "dt": 1561820400,
            "main": {
                "temp": 21.02,
                "temp_min": 21.02,
                "temp_max": 22.86,
                "pressure": 1002.79,
                "sea_level": 1002.79,
                "grnd_level": 999.35,
                "humidity": 77,
                "temp_kf": -1.83
            },
            "weather": [
                {
                    "id": 500,
                    "main": "Rain",
                    "description": "light rain",
                    "icon": "10n"
                }
            ],
            "clouds": {
                "all": 100
            },
            "wind": {
                "speed": 1.05,
                "deg": 85.832
            },
            "rain": {
                "3h": 0.188
            },
            "sys": {
                "pod": "n"
            },
            "dt_txt": "2019-06-29 15:00:00"
        },
        {
            "dt": 1561831200,
            "main": {
                "temp": 20.62,
                "temp_min": 20.62,
                "temp_max": 21.99,
                "pressure": 1001.11,
                "sea_level": 1001.11,
```


list：3時間ごとの天気予報（全部で5日分）

main

temp：平均気温

weather

main：天気予報（Clear、Clouds、Rainなど）

dt_txt：天気予報の時刻（UTC）

プログラムを書いていきましょう

```
10  /**
11   * 2019/07/01 北九州高専 プログラミング講座
12   *
13   * 【課題1】
14   * APIを使って好きな都市の3時間ごとの天気予報を出力してください。
15   */
16  public class Task1 {
17      @SuppressWarning
        Run | Debug
18      public static vo
19          // 天気予報を取得す
20          Forecast forecast = new WeatherApiClient("037d75dd9c0210aa6fe2e5692fc278ea")
21                                  .hasShowResponseBody(true)
22                                  .get("forecast", Map.of("id", "5128638"), Forecast.class);
23
24          // ここから下にプログラムを書いていく
25
26
27      }
28  }
```

WebAPIのレスポンスには3時間ごとの天気が設定されている。

取得した天気予報は変数：forecastに格納されている

Forecastのクラス構造

https://github.com/if-matsumoto/java-lecture/blob/master/src/jp/co/interfactory/lecture/model/Forecast.java

```
9   /**
10   * 2019/07/01  北九州高専　プログラミング講座
11   * 3時間ごと5日分の天気データを格納するためのモデルクラス
12   */
13  public class Forecast {
14      // リスト
15      public List<ListElement> list;
16      // リストの要素
17      public class ListElement {
18          // メインの情報
19          public Main main;
20          public class Main {
21              // 平均気温
22              public float temp
23              // 最低気温
24              public float temp_min;
25              // 最高気温
26              public float temp_max;
27          }
28          // 天気情報
29          public List<Weather> weather;
30          // 天気情報
31          public class Weather {
32              // 種別(Rain, Snow, Extreme etc.)
33              public String main;
34          }
35          // 日時
36          public long dt;
37
```

3時間ごとの天気予報がListElementというクラスに設定され、リスト化されている

Forecastのクラス構造

https://github.com/if-matsumoto/java-lecture/blob/master/src/jp/co/interfactory/lecture/model/Forecast.java

```
38          /**
39           * システムのデフォルトタイムゾーンへ変換した日時を文字列で取得します。
40           * @return yyyy-MM-dd HH:mm形式
41           */
42          public String getDtTxt() {
43              // dtが秒単位なのでミリ秒
44              long dtMills = this.dt * 10
45              var formatter =
46              return Instant.o
47          }
48
49          /**
50           * 天気の取得
51           * @return
52           */
53          public String getWeather() {
54              // 確認した範囲では1要素ず
55              return this.weather
56          }
57      }
```

3時間ごとの時刻は、getDtTxt()メソッドで取得可能。
（このメソッド内でシステムのタイムゾーンに変換される）

3時間ごとの天気は、getWeather()メソッドで取得可能。

プログラムを書いていきましょう

```
10  /**
11   * 2019/07/01 北九州高専 プログラミング講座
12   *
13   * 【課題1】
14   * APIを使って好きな都市の3時間ごとの天気予報を出力してください。
15   */
16  public class Task1 {
17      @SuppressWarnings("unused")
        Run | Debug
18      public static void main(String[] args) throws Exception {
19          // 天気予報を取得するWebAPIを実行
20          Forecast forecast = new WeatherApiClient("037d75dd9c0210aa6fe2e5692fc278ea")
21                                  .hasShowResponseBody(true)
22                                  .get("forecast", Map.of("id", "5128638"), Forecast.class);
23
24          // ここから下にプログラムを書いていく
25
26
27      }
28  }
```

listをfor文で繰り返し、
getDtTxt()メソッドとgetWeather()メソッドで
時刻と天気予報を出力する。
出力はSystem.out.println();

Javaのfor文（基本的なfor文）

```
for (初期化式; 条件式; 変化式) {
    // 条件式がtrueのときに繰り返す処理
}
```

```
List<String> list = new ArrayList<>();
list.add("a");
list.add("b");
list.add("c");
for (int i = 0; i < list.size(); i++) {
    System.out.println(list.get(i));
}
```

Javaのfor文（拡張for文）

配列またはコレクションの全ての要素を順に取り出していく。

コレクション：データの集合体のようなもの

```
for (データ型 変数名 : 配列やコレクション) {
    // コレクションの要素数だけ繰り返す処理
}
```

```
List<String> list = new ArrayList<>();
list.add("a");
list.add("b");
list.add("c");
for (String str : list) {
    System.out.println(str);
}
```

Javaのfor文（forEachメソッド）

Java8からforEachメソッドが追加され、

引数にラムダ式（関数）を渡すことができる

```
List<String> list = new ArrayList<>();
list.add("a");
list.add("b");
list.add("c");
list.forEach(str -> System.out.println(str));
```

拡張forを使ってみましょう

```
17  */
18  public class Task1 {
19      @SuppressWarnings("unused")
        Run | Debug
20      public static void main(String[] args) throws Exception {
21          // 天気予報を取得するWebAPIを実行
22          Forecast forecast = new WeatherApiClient("037d75dd9c0210aa6fe2e5692fc278ea")
23                                  .hasShowResponseBody(true)
24                                  .get("forecast", Map.of("id", "5128638"), Forecast.class);
25
26          // ここから下にプログラムを書いていく
27          for (Forecast.ListElement listElement : forecast.list) {
28
29          }
30      }
31  }
```

拡張for文でlistの要素数だけ繰り返す

時刻と天気予報を出力

```
17  */
18  public class Task1 {
19      @SuppressWarnings("unused")
        Run | Debug
20      public static void main(String[] args) throws
21          // 天気予報を取得するWebAPIを実行
22          Forecast forecast = new WeatherApiClient("037c[illegible]c0210aab1e2e5692fc278ea")
23                                  .hasShowResponseBody(false)
24                                  .get("forecast", Map.of("id", "5128638"), Forecast.class);
25
26          // ここから下にプログラムを書いていく
27          for (Forecast.ListElement listElement : forecast.list) {
28              System.out.println(listElement.getDtTxt());
29              System.out.println(listElement.getWeather());
30          }
31      }
32  }
```

WebAPIのレスポンスを
出力しないように**false**に変更

getDtTxt()メソッドとgetWeather()メソッドで
時刻と天気予報を出力する。

実行してみる（コンパイル→実行）

Windows

```
> cd C:¥work¥java-lecture

> javac -encoding UTF-8 -sourcepath src -cp lib/* -d bin src¥jp¥co¥interfactory¥lecture¥Task1.java

> java -cp bin;lib/* jp.co.interfactory.lecture.Task1
```

Mac / Linux

```
$ cd C:¥work¥java-lecture

$ javac -encoding UTF-8 -sourcepath src -cp lib/* -d bin src/jp/co/interfactory/lecture/Task1.java

$ java -cp bin:lib/* jp.co.interfactory.lecture.Task1
```

→ 3時間ごとの時刻と天気が出力されればOK

実行してみる（コンパイル→実行）

```
C:¥work¥java-lecture>javac -encoding UTF-8 -sourcepath src -cp lib/* -d bin src¥jp¥co¥interfactory¥lecture¥Task1.java

C:¥work¥java-lecture>java -cp bin;lib/* jp.co.interfactory.lecture.Task1
connect to http://api.openweathermap.org/data/2.5/forecast?appid=037d75dd9c0210aa6fe2e5692fc278ea&units=metric&id=5128638
2019-07-01 00:00
Rain
2019-07-01 03:00
Rain
2019-07-01 06:00
Rain
2019-07-01 09:00
Rain
2019-07-01 12:00
Clouds
2019-07-01 15:00
Clear
2019-07-01 18:00
Clear
2019-07-01 21:00
Clear
2019-07-02 00:00
Clouds
2019-07-02 03:00
Clouds
2019-07-02 06:00
Clear
2019-07-02 09:00
Clear
2019-07-02 12:00
Clouds
2019-07-02 15:00
Clouds
2019-07-02 18:00
Clouds
2019-07-02 21:00
```

こんな感じで出力されればOK

foreachを使ってみましょう

```
27        // ここから下にプログラムを書いていく
28        for (Forecast.ListElement listElement : forecast.list) {
29            System.out.println(listElement.getDtTxt());
30            System.out.println(listElement.getWeather());
31        }
32        forecast.list.forEach(listElement -> {
33                System.out.println(listElement.getDtTxt());
34                System.out.println(listElement.getWeather());
35        });
36    }
37 }
```

Foreach同じ出力結果が得られるように実装

拡張for文の時と同じようにコンパイルして実行