Coding Interview

Requirements:

Xcode 15.0 & above

Swift 5.0/6.0 & above(or SwiftUI)

iOS 16.0 & above

Basic idea of asynchronous coding(eg. Async/await), combine, generics.

Use MVVM/MVC architecture for code design

Exercise:

Create a sample app which displays a list of temperatures for top 5 cities from Accuweather, using accuweather developer api which are provided in the exercise.

Xcode settings before starting:

- In your info.plist please add `App Transport Security Settings` as a dictionary.
- Once added, please add `Allow Arbitary Loads` boolean under transport security settings and set it to be true.

Introduction:

This exercise has 3 parts:

1) Networking using asynchronous tasks.
2) UI - creating tableview to display the data retrieved as part of networking step.
3) Data Updates/Binding(optional):

- Use Observable object or Combine publishers to keep your list view updated, rather than calling tableview.reload

Or

- Use Observation framework or Combine to bind your data source with list view for dynamic updates.

Networking:

Networking step has 3 parts:

1) Use Accuweather developer api to get list of top cities, from the response from the call only decode the city name and city key which we will use in UI display and making later call for temperature.

Note: We only need top 5 cities and rest can be discarded.

API: `"http://dataservice.accuweather.com/locations/v1/topcities/50"`

Query parameter:

1) query item name: `"apikey"` and value: `"kAP3jGb5EV3XvxVxV6xfmXoNSjGovsvD"`

Sample Response:


```
[
  {
    "Version": 1,
    "Key": "28143",
    "Type": "City",
    "Rank": 10,
    "LocalizedName": "Dhaka",
    "EnglishName": "Dhaka",
    "PrimaryPostalCode": "",
    "Region": {
      "ID": "ASI",
      "LocalizedName": "Asia",
      "EnglishName": "Asia"
    }
  },
  {
    "Version": 1,
    "Key": "113487",
    "Type": "City",
    "Rank": 10,
    "LocalizedName": "Kinshasa",
    "EnglishName": "Kinshasa",
    "PrimaryPostalCode": "",
    "Region": {
      "ID": "AFR",
      "LocalizedName": "Africa",
      "EnglishName": "Africa"
    }
  },
  ....
  ....
  ....
  ]
```

Above is the sample response, which is an array of identical items where every item has a `key` and `englishName` (also highlighted) which is what we need for the exercise and rest of the data be discarded.
Create your models and decoding based on in the above response.

2) Using the array of cities from the previous call(only the first 5 cities are needed from the cities array), make another accuweather api call to retrieve temperatures for each of the 5 cities which uses city `key` retrieved for each city from the above call.
   Note: Below call is dependent on the previous call and the call need to be made for each city using the city key.

API: `"http://dataservice.accuweather.com/forecasts/v1/hourly/1hour/\(city.key)"`
Query parameter:
Use the same query parameter `apikey` from previous call.

Sample Response:

```
[
  {
    "DateTime": "2022-08-14T21:00:00-05:00",
    "EpochDateTime": 1660528800,
    "WeatherIcon": 7,
    "IconPhrase": "Cloudy",
    "HasPrecipitation": false,
    "IsDaylight": false,
    "Temperature": {
      "Value": 54,
      "Unit": "F",
      "UnitType": 18
    },
    "PrecipitationProbability": 20,
    "MobileLink": "http://www.accuweather.com/en/co/bogota/107487/hourly-
weather-forecast/107487?day=1&hbhhour=21&lang=en-us",
    "Link": "http://www.accuweather.com/en/co/bogota/107487/hourly-weather-
forecast/107487?day=1&hbhhour=21&lang=en-us"
  }
]
```

Above is the sample response, which is an array of one single item where we only need `Temperature` and nested `Value` in temperature which is what we need for the exercise and rest of the data be discarded.
Create your models and decoding based on in the above response.

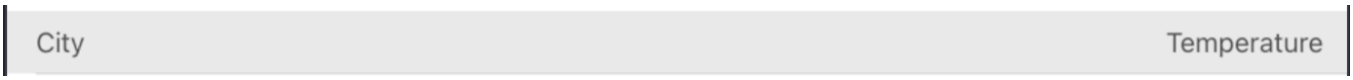
Makes sure to create your models such that every city has its city name, key and temperature which will be display on the tableview

Note: Later call is dependent on the first call, this required for structured concurrency so please prefer to use async/await for both the above steps as response of the first call is needed for the second call.

UserInterface:

1) Create a list UI using table view and UI tableview cells or SwiftUI lists to display the city name and city temperature for the 5 cities retrieved from the Networking part.

Tableview Header:



Fonts: Use native iOS system font style: .subheadline
Header background color - system color: .gray
Font color - system color: .darkgray

TableView Cell:

Fonts: Use native iOS system font style: .body
Cell background color - system color: .white
Font color - system color: .black

Final UI:

Note: All the UI including constraints should be done programmatically without using storyboard. For constraints use 16pts as top, leading, trailing, bottom across the UI for all the labels in both header and cell.

Optional Step: Data Update/Binding

1) Use Observable object or combine publishers (published properties, published subjects) to keep

your list updated with datasource changes or updates across the app.

Or

2) Create a custom generic UI table view data source and using observation object or combine publishers (published properties, published subjects) bind it with tableview to dynamically update tableview as the data come in.