Unit – 4

JSON Concept

4.1 Concept and Features of JSON, Similarities, and difference among JSON and XML

Concept of JSON

JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa)

Features of JSON

- JSON are Scalable. Because of language-independent, it works with most modern programming languages.
- JSON is lightweight.
- JSON is easy to read and write.
- JSON is a text-based, human-readable data exchange format.

Similarities between JSON and XML

- Both JSON and XML are human-readable languages.
- Both JSON and XML support Unicode, thus any human written language can be written in JSON and XML documents.
- Both JSON and XML can be parsed.
- The data in both JSON and XML can be fetched using XMLHttpRequest.

The difference between JSON and XML

JSON	XML
JSON is a data interchange format and only provides a data encoding specification.	XML is a language to specify custom mark-up languages and provides a lot more than data interchange.
JSON type: string, number, array, Boolean	All XML data should be a string
Data is readily accessible as JSON objects	XML data needs to be parsed
JSON is supported by most browsers.	Cross-browser XML parsing is not an easy task

JSON supports arrays thus it is easy to transfer a big amount of data items using JSON.	XML doesn't support arrays
Retrieving value is easy	Retrieving value is difficult
JSON files are easy to read as compared to XML.	XML documents are relatively more difficult to read and interpret.

4.2 JSON objects (with strings and Numbers)

JSON object is a comma-separated collection of key-value pare written between opening and closing curly braces { }.

Example:

```
{
    "course" : "BCA" ,
    "year" : 3 ,
    "subject" : "Advance Mobile Computing"
}
```

Characteristics of JSON Object

- JSON object is written inside opening and closing curly braces { }.
- Each member in the JSON object is a key: value pair. Considering the above example: "course": "BCA" where "course" is a key, "BCA" values and, : (colon) is written between key and value.
- The key in each member is always a string which means the key will be always written inside double quotes " ".
- On the other hand, the Value can be any of the following types
    - String
    - Number
    - Array
    - Object
    - Boolean
    - Null

4.3 JSON Arrays and their examples

JSON array represents an ordered list of values. JSON array can store multiple values. It can store strings, numbers, Boolean or JSON objects in a JSON array.

A JSON array is an ordered collection of values separated by " , " (comma) and it is always written inside [ ] (Square bracket).

4.3.1 Array of string, Array of Numbers, Array of Booleans, Array of objects, Multidimensional Arrays

JSON Array of Strings

JSON Array of String is an ordered collection of comma-separated strings enclosed within the square bracket.

Consider the following example:

```
let arrayOfString = ["One", "Two", "Three", "Four", "Five", "Six"];
```

JSON Array of Numbers

JSON Array of Numbers is an ordered collection of comma-separated numeric values enclosed within the square bracket. Values can be integer numbers or real numbers.

Consider the following example:

```
let arrayOfNumber = [ 1, 2, 3, 4, 5, 6];
```

JSON Array of Booleans

JSON Array of Booleans is an ordered collection of comma-separated Boolean values enclosed within the square bracket.

Consider the following example:

```
let arrayOfBoolean = [true, false, true, false];
```

JSON Array of Objects

JSON Array of Object is an ordered collection of comma-separated JSON objects enclosed within the square bracket.

Consider the following example:

```
let ArrayOfObjects = {
        "courses":[
                {
                        "shortName":"BCA",
                        "duration":3,
                        "fullName": "Bachelor of Computer Application"
                },
                {
                        "shortName":"BBA",
                        "duration":3,
                        "fullName": "Bachelor of Business Administration"
                },
                {
                        "shortName":"BCOM",
                        "duration":3,
                        "fullName": "Bachelor of Commerce"
                },
                {
                        "shortName":"BSC",
                        "duration":3,
                        "fullName": "Bachelor of Science"
                }

        ]
}
```

JSON Multidimensional Array

We can also store JSON array inside JSON array, it is known as array of arrays or multidimensional array. Any type of JSON array mentioned above can be written as member to an JSON array.

Consider the following example:

```
let multidimensionalArray =  [
                                [ "BCA", "BBA", "BCOM", "BSC" ],
                                [ 1, 2, 3 ],
                                [ true, false, false ]
                             ]
```

4.3.2 JSON comments

JSON doesn't support any type of Comment in it but, there is one way of adding an extra attribute for comment in JSON object where you can have "comment" as a key and you can provide your comment text as a value to it.

Consider the following example:

```
let commentExample = {
                        "courses": {
                          "name":     "BCA",
                          "years":    3,
                          "comments":   "Its a 3 years UG course"
                         }
                     }
```

Here, "comments" attribute can be treated as comment and "its a 3 years UG course" is the actual comment text.

4.4 Building multi-screen apps

As we know we use activity for creating a screen(layout) in any android application by default on creating a new android application it comes with one activity(screen) named "MainActivity" but most android applications have more than one screen, in this topic, we will learn about adding multiple activities (screens) in android application and the method of navigating from one activity(screen) to another activity(screen) using intent.

4.4.1 Intents and their applications, types of intents

Android Intent is the message that is passed between components such as activities, content providers, broadcast receivers, services, etc.

It is generally used with startActivity() method to invoke activity, broadcast receivers, etc.

The dictionary meaning of intent is intention or purpose. So, it can be described as the intention to do action. The Labelled Intent is the subclass of the "android.content.Intent" class.

Android intents are mainly used to

- Start a service
- Launch an activity
- Display a web page
- Broadcast a message
- Dial a phone call etc

There are two primary forms of intents you will use

1. Explicit Intents
    - Explicit intents specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name.
    - You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start.
    - For example, you might start a new activity within your app in response to a user action, or start a service to download a file in the background.

2. Implicit Intents
    - Implicit intents do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.
    - For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

4.4.2 Data exchange from one activity to another using intent

For understanding the data exchange from one activity to another using intent let us consider the following example. In this example, apart from the default MainActivity, we have to create a new activity named SecondaryActivity. This task will be done using Explicit Intent.

The Steps to create a new activity in the android studio:

1. In the **Project** window, right-click the **app** folder and select **New > Activity > Empty Activity**.
2. In the **Configure Activity** window, enter "SecondaryActivity" for **Activity Name(you can use any appropriate name for your activity)**. Leave all other properties set to their defaults and click **Finish.**

Android Studio automatically does three things:

- Creates the SecondaryActivity.kt file.
- Creates the layout file activity_secondary.xml, which corresponds with the SeconderyActivity.kt file.
- Adds the required <activity> element in AndroidManifest.xml.

Code to navigate from one Activity(screen) to another Activity(screen) with data

Consider the following scenario to understand the example

The application has to Activities

1. MainActivity
2. SecondaryActivity

Steps in this example will navigate from MainActivity to SecondaryActivity Along with data that is written inside the edit text control on the MainActivity. That text will be displayed on the TextView on Secondary Activity

Step 1: Add a Button with id btnNavigate in the activity_main.xml file.

Step 2: Add an EditText with id editTextPersonName in activity_main.xml

Step 3: Write Following Code inside onCreate function of mainActivity.kt

```
val btn_click_me = findViewById<Button>(R.id.btnNavigate)
val etName = findViewById<EditText>(R.id.editTextPersonName)
btn_click_me.setOnClickListener {
  val personName = etName.text
  val intent = Intent(this,SecondaryActivity::class.java)
  intent.putExtra("name", personName.toString())
  startActivity(intent)
}
```

Note: You will have to import several classes to make this code working in the import section of MainActivity.kt file to work with Intent.

Explanation of the above code:

- The Intent constructor takes two parameters, a Context, and a Class.
- The Context parameter is used first because the Activity class is a subclass of Context the keyword "this" represents the current application context.
- The Class parameter of the app component, to which the system delivers the Intent, is, in this case, the "SecondaryActivity::class.java" will start and the layout file activity_scecondary.xml will be displayed.

Step 4: Add a TextView with Id textViewGreetings in the activity_secondary.xml file.

Step 5: Write the Following Code inside the OnCreate function of SecondaryActivity.kt

```
val bundle = intent.extras
var personName = ""
if (bundle != null) {
  personName = bundle.getString("name").toString()
  val tvName = findViewById<TextView>(R.id.textViewGreetings)
  tvName.text = personName
}
```

Explanation of the above code:

- In the above code in the onCreate function of SecondaryActivity.kt first of all will store data received via Intent inside the bundle variable.
- In the second line, we will check if the bundle is null or not.
- If the bundle is not null the code will read the string using the label given on the mainActivity and store the value inside the personName variable
- Lastly the data received from the Intent will be displayed on the textView of activity_secondary.xml using the last line.

4.5 Working with implicit intents

Android Implicit Intent invokes the component of another app to handle the request. It does not specify the component name specifically.

For example, if we want to share data using Intent or we want to visit a website in a browser, it invokes the relevant component to fulfil the request.

4.5.1 Opening web URLs through app

For opening a web URL through app Intent.ACTION_VIEW will be used instead of the application context.

Let us consider the following example.

Step 1: Add one button in the layout activity_main.xml with id btnJ2L

Step 2: After that add the following code in MainActivity.kt onCreate function

```
val visitJ2L = findViewById<Button>(R.id.btnJ2L)
visitJ2L.setOnClickListener {
  val intent = Intent(Intent.ACTION_VIEW)
  intent.setData(Uri.parse("https://jump2learn.com/"))
  startActivity(intent)
}
```

Explanation of the above code:

- In the above code an intent has been created using ACTION_VIEW and inside setData function we have passed a Website URL
- When a user clicks on this button android device will look for the default browser in the device and it will open the given URL in the browser window automatically.

4.5.2 Sharing media from our app to other apps

Sharing media from our app to other apps is a part of implicit intent called ACTION_SEND.In ACTION_SEND You need to specify the data and its type. The system automatically identifies the compatible activities that can receive the data and displays them to the user.

To understand this implicit Intent ACTION_SEND we will be considering an example of sending an image from our application to some other application. Let us understand it step by step.

Step 1:

Create a layout having one ImageView Control and One-Button Control. Provide proper id to each control

In This Example

Button id = btImgShare

ImageView id = imageView

Step 2:

Add one image which will be displayed in the ImageView control.

To add the image in to Android Project goto following location in project explorer

App / src / main / res / drawable

Add one png image in the given location. In this example "j2l.png" has been added to given location.

Step 3:

Set property app:srcCompat = "@drawable/j2l" and tools:srcCompat = "@drawable/j2l" of ImageView Control in the code window of layout file. Here j2l is the name of our image.

Step 4:

Create onClickListener for the button btImgShare to sendImage from our app to another app.Add following code to the activity.kt file of your project

```
val btImageShare = findViewById<Button>(R.id.btImgShare)
btImageShare.setOnClickListener {
  val imgView = findViewById<ImageView>(R.id.imageView)
  val bitmap = imgView.drawable.toBitmap()
  val bytes = ByteArrayOutputStream()
  bitmap.compress(Bitmap.CompressFormat.PNG, 100, bytes)
  val path: String = MediaStore.Images.Media.insertImage(
    this.getContentResolver(),
    bitmap,
    "Title",
    null
  )
  val uriToShare = Uri.parse(path)
  val shareIntent: Intent = Intent().apply {
    action = Intent.ACTION_SEND
    putExtra(Intent.EXTRA_STREAM,uriToShare)
    type = "image/png"
  }
  startActivity(Intent.createChooser(shareIntent, null))
}
```

Explanation of the above Code :

- Variable imgmap will store the image from our imageView control into the bitmap format.
- Variable bytes will be used to store the image in the binary format.
- Compress method will convert the image from bitmap to the array of bytes.
- Path variable will store the path of image from android file system.
- Path will be then converted to URI using Uri.parse method so that it can be sent along with intent.
- Variable shareIntent represents the Intent which has 2 properties
    - Action = Intent.ACTION_SEND

- Type = "image/png"

  Here Type must be exactly same to the type of image.

- We will pass the image using its URI with putExtra function. As we are going to send the binary stream of image, Intent.EXTRA_STREAM will be used as first argument and uriToShare as second argument.
- On startActivity function call createChooser will be used as we are goint to send medai to other application.
- createChosser wil open the drawer with all available application on device that supports the media type.
- We have to select the app to which we want to send the image and the task is done.