MODULE 5

JSON AND XML

- **JSON or JavaScript Object Notation** is a lightweight text-based open standard designed for human-readable data interchange.
- Best alternative for XML.
- Conventions used by JSON are known to programmers, which include C, C++, Java, Python, Perl, etc.
- The format was specified by Douglas Crockford.
- It is used when writing javascript based applications that include browser extension and websites.
- It has been extended from the JavaScript scripting language.
- The filename extension is **.json**.
- JSON Internet Media type is **application/json**.
- The Uniform Type Identifier is public.json.

- JSON is "self-describing" and easy to understand
- **Used for transmitting data between server and web application**.

- This example defines an employees object: an array of 3 employ (objects):

-
  ```
  {
  "employees":[
     {"firstName":"John", "lastName":"Doe"},
     {"firstName":"Anna", "lastName":"Smith"},
     {"firstName":"Peter", "lastName":"Jones"}
  ]
  }
  ```

JSON Syntax Rules
Data is in name/value pairs
Data is separated by commas
Curly braces hold objects
Square brackets hold arrays

JSON Data - A Name and a Value
JSON data is written as name/value pairs, just like JavaScript object properties.
A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:
"firstName":"John"

JSON Objects
Object Syntax
Example
{ "name":"John", "age":30, "car":null }
JSON objects are surrounded by curly braces {}.
JSON objects are written in key/value pairs.
Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
Keys and values are separated by a colon.
Each key/value pair is separated by a comma.

JSON Arrays

JSON arrays are written inside square brackets.

Array values must be of type string, number, object, array, boolean or null.
JSON arrays represent an ordered list of values. values must be seperated by commas.
Just like in JavaScript, an array can contain objects:

```
"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName":"Jones"}
]
```

In the example above, the object "employees" is an array. It contains three objects.
Each object is a record of a person (with a first name and a last name).

JSON ARRAY OF STRINGS

[“sunday”,”Monday”,”Tuesday”,”Wednesday”]

JSON array of numbers

[1,2,3,4,5]

JSON array of booleans

[true,false,false,true]

Converting a JSON Text to a JavaScript Object

A common use of JSON is to read data from a web server, and display the data in a web page.

For simplicity, this can be demonstrated using a string as input.

First, create a JavaScript string containing JSON syntax:

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

Then, use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object:

```
var obj = JSON.parse(text);
```

Finally, use the new JavaScript object in your page:

Example

```
<p id="demo">
</p>

<script>
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

Output

```
Anna Smith
```

JSON Vs XML

Both JSON and XML can be used to receive data from a web server.
The following JSON and XML examples both define an employees object, with an array of 3 employees:

JSON Example

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML Example

```
<employees>
  <employee>
    <firstName>John</firstName>
<lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

JSON is Like XML Because

Both JSON and XML are "self describing" (human readable)
Both JSON and XML are hierarchical (values within values)
Both JSON and XML can be parsed and used by lots of programming languages
Both JSON and XML can be fetched with an XML Http Request

JSON is Unlike XML Because

JSON doesn't use end tag
JSON is shorter
JSON is quicker to read and write
JSON can use arrays

The biggest difference is:
XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

JSON	XML
It is JavaScript Object Notation	It is Extensible markup language
It is based on JavaScript language.	It is derived from SGML.
It is a way of representing objects.	It is a markup language and uses tag structure to represent data items.
It does not provides any support for namespaces.	It supports namespaces.
It supports array.	It doesn’t supports array.
Its files are very easy to read as compared to XML.	Its documents are comparatively difficult to read and interpret.
It doesn’t use end tag.	It has start and end tags.
It is less secured.	It is more secured than JSON.
It doesn’t supports comments.	It supports comments.
It supports only UTF-8	It supports various encoding formats

Features of JSON
Easy to use - JSON API offers high-level facade, which helps you to simplify commonly used use-cases.
Performance - JSON is quite fast as it consumes very less memory space, which is especially suitable for large object graphs or systems.
Free tool - JSON library is open source and free to use.
Doesn't require to create mapping - JSON API provides default mapping for many objects to be serialized.
Clean JSON - Creates clean, and compatible JSON result that is easy to read.
Dependency - JSON library does not require any other library for processing.

In JSON, values must be one of the following data types:

a string.

a number.

an object (JSON object)

an array.

a boolean.

null.

Looping an Object

You can loop through object properties by using the for-in loop:

Example

```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
  document.getElementById("demo").innerHTML += x;
}
```

In a for-in loop, use the bracket notation to access the property *values*:

Example

```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
  document.getElementById("demo").innerHTML += myObj[x];
}
```

Nested JSON Objects

Values in a JSON object can be another JSON object.

Example

```
myObj = {
  "name":"John",
  "age":30,
  "cars": {
    "car1":"Ford",
    "car2":"BMW",
    "car3":"Fiat"
  }
 }
```

You can access nested JSON objects by using the dot notation or bracket notation:

Example

```
x = myObj.cars.car2;
// or:
x = myObj.cars["car2"];
```

Arrays as JSON Objects

Example

["Ford", "BMW", "Fiat"]

Arrays in JSON are almost the same as arrays in JavaScript.

In JSON, array values must be of type string, number, object, array, boolean or *null*.

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and *undefined.*

Arrays in JSON Objects

Arrays can be values of an object property:

Example

```
myObj={
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

Accessing Array Values

You access the array values by using the index number:

Example

```
x = myObj.cars[0];
```

Accessing Object Values

You can access the object values by using dot (.) notation:

Example

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj.name;
```

You can also access the object values by using bracket ([]) notation:

Example

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Looping Through an Array

You can access array values by using a **for-in** loop:

```
myObj={…..,cars:[“fiat”,”bmw”,”indica”]}
```

Example

```
for (i in myObj.cars) {
  x += myObj.cars[i];
}
```

Or you can use a for loop:

Example

```
for (i = 0; i < myObj.cars.length; i++) {
  x += myObj.cars[i];
}
```

Nested Arrays in JSON Objects

Values in an array can also be another array, or even another JSON object:

Example

```
myObj = {
  "name":"John",
  "age":30,
  "cars": [
    { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
    { "name":"BMW", "models":[ "320", "X3", "X5" ] },
    { "name":"Fiat", "models":[ "500", "Panda" ] }
  ]
 }
```

To access arrays inside arrays, use a for-in loop for each array:

Example

```
for (i in myObj.cars) {
  x += "<h1>" + myObj.cars[i].name + "</h1>";
  for (j in myObj.cars[i].models) {
    x += myObj.cars[i].models[j];
  }
}
]
```

PARSING JSON AND XML

JSON PARSER

- Parsing means interpreting.
- Here we are converting the string into a JSON object.
- JSON parser parses a JSON object embedded in a string field and parses the parsed data to an output field in the record.
- Based on the contents of the JSON object, the resulting field is either a string, a map or an array.

```
<!DOCTYPE html>
<html>
<body>
<h1>JSON parse()</h1>
<p>JSON.parse() parses a string, written in
JSON format, and returns a JavaScript
object.</p>
<p>The value of "firstName" is:</p>
<p id="demo"></p>
<script>
var obj = JSON.parse('{"firstName":"John",
"lastName":"Doe"}');

document.getElementById("demo").innerHT
ML = obj.firstName;
</script>
</body>
</html>
```

Output

```
JSON parse()
JSON.parse() parses a string, written in
JSON format, and returns a JavaScript
object.

The value of "firstName" is:
John
```

The JSON.parse() method parses a string and returns a JavaScript object.
The string has to be written in JSON format.

Parsing JSON HTTP Response
If we get HTTP Response as JSON format, then we require to parse it.
The following fragment is used to parse JSON.

Get the content from HTTP response

```
HttpEntity httpentity=httpResponse.getEntity()
InputStream is=httpentity.getContent();
```

Read the content stored into JSON Object

```
BufferedReader reader=new BufferedReader(new InputStreamReader is,”iso-8859-1”),8);
StringBuilder sb=new StringBuilder();
String line=null;
while((line=reader.readLine()!=null);
{sb.append(line+”/n”);
}
is.close();
String json=sb.toString();
JSONObject jobj.getJSONArray(“array name”);
List<Map<String,String>>list=new ArrayList<Map<String,String>>();
For(int i=0;i<arr.length();i++)
{JSONObject c=arr.getJSONObject(i);
String att1=c.getString(“attribute name”);
String att2=c.getString(“attribute name”);
String att3=c.getString(“attribute name”);
}
```

PARSING XML HTTP RESPONSE

If we get the HTTPResponse as XML format,then we require to parse it.

Get the content from HTTP Response

```
HttpEntity resEntity = httpresponse.getEntity();
 response = EntityUtils.toString(resEntity);
```

Convert the response data to Document Object

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder=factory.new DocumentBuilder();
 InputSource is=new InputSource();
Is.setCharacterStream(new StringReader(xml));
Document doc=builder.parse(is);
```

Finally get the document from DocumentObject

```
NodeList nl=doc.getElementByTagName(“tagname”);
List <Map<String,String>>list=new ArrayList<Map<String,String>>();
<item>
For(int i=0;i<n1.getlength();i++)
{ Map<String,String>>map=new HashMap<String,String>>();
Element e=(Element)nl.item(i);
Map.put(“att name”,getValue(e,”tag-name”);

Map.put(“att name”,getValue(e,”tag-name”);
Map.put(“att name”,getValue(e,”tag-name”);
List.add(map);
}
```

- **Getting the child element from the Element**

```
public String getValue(Element e, String tagname){
NodeList n=e.getElementByTagName("tag-name");
Node element=n.item(0);
Node child;
if(element!=null){
if(element.hasChildNodes()){
for (child=element.getFirstChild();child!=null;child=child.getNextSibling()){
if(child.getNodeType()==Node.TEXT_NODE)
String att1=child.getNodeValue();
}
}}
```

GOOGLE PLAY SERVICES AND MAPS

With google play services,your app takes advantage of the latest google powered features such as maps,cast and more with automatic platform updates distributed as APK through the Google play store.

Google play services

- Layer of software that connects your apps, google services and android together.
- It runs in the background and manages notifications, location etc.
- Google is not putting all updates into the android os but has been pushing updates through google apps and services.

Overview of Google Play services

Google Play services powers a broad set of SDKs on Android to help you build your app, enhance privacy and security, engage users, and grow your business.

These SDKs are unique in that they only require a thin client library to be included in your app, as shown in figure 1.

At runtime, the client library communicates with the bulk of the SDK's implementation and footprint in Google Play services.

By providing shared, client-side implementations, Google Play services:

- Helps to optimize on-device resources, such as storage and memory, to improve your users' overall experience.
- Receives automatic updates—independent of OS, OEM, or app updates—so your users receive improvements and bug fixes more quickly.
- Powers SDKs that are backward compatible to Android 4.1 (API level 16), enabling you to reach more users with less effort.

How it works?

The client library contains interfaces to the individual Google services and allows you to obtain authorization from the users to gain access to these services with their credentials.

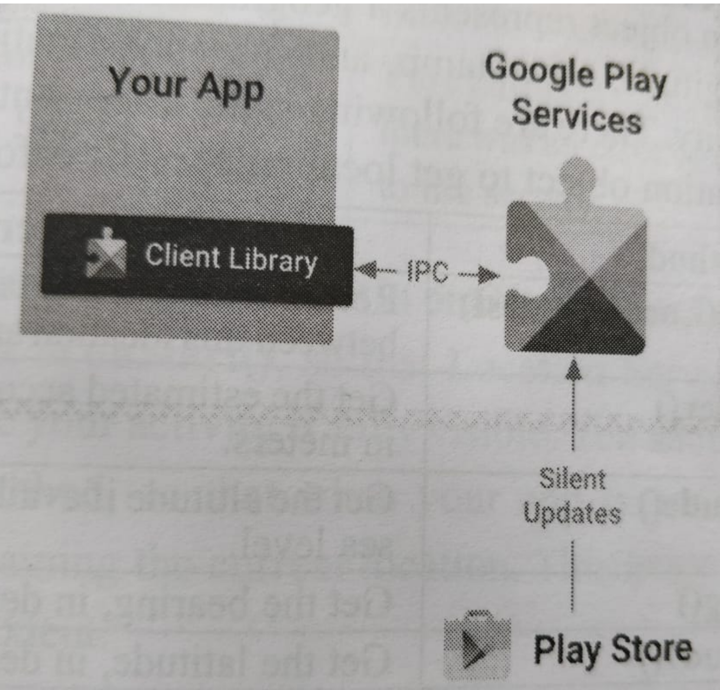


Fig 11.2 The Google Play Services

It also contains API to resolve any issues at runtime such as missing, disabled or out of date Google play services(Android package).

Benefits for your APP

Google play services provides freedom to use the newest APIs for popular Google play services without worrying about device support.

Updates to Google play services are distributed automatically by Google playstore and new versions of client library are delivered through Google MAVEN Repository.

LOCATION SERVICES

Android location APIs make it easy to **build location-aware applications** without needing to focus on the details of the **underlying location technology**.

This becomes possible with the help of **Google play services**, which facilitates adding location awareness to your app with **automated location tracking, geofencing, activity recognition.**

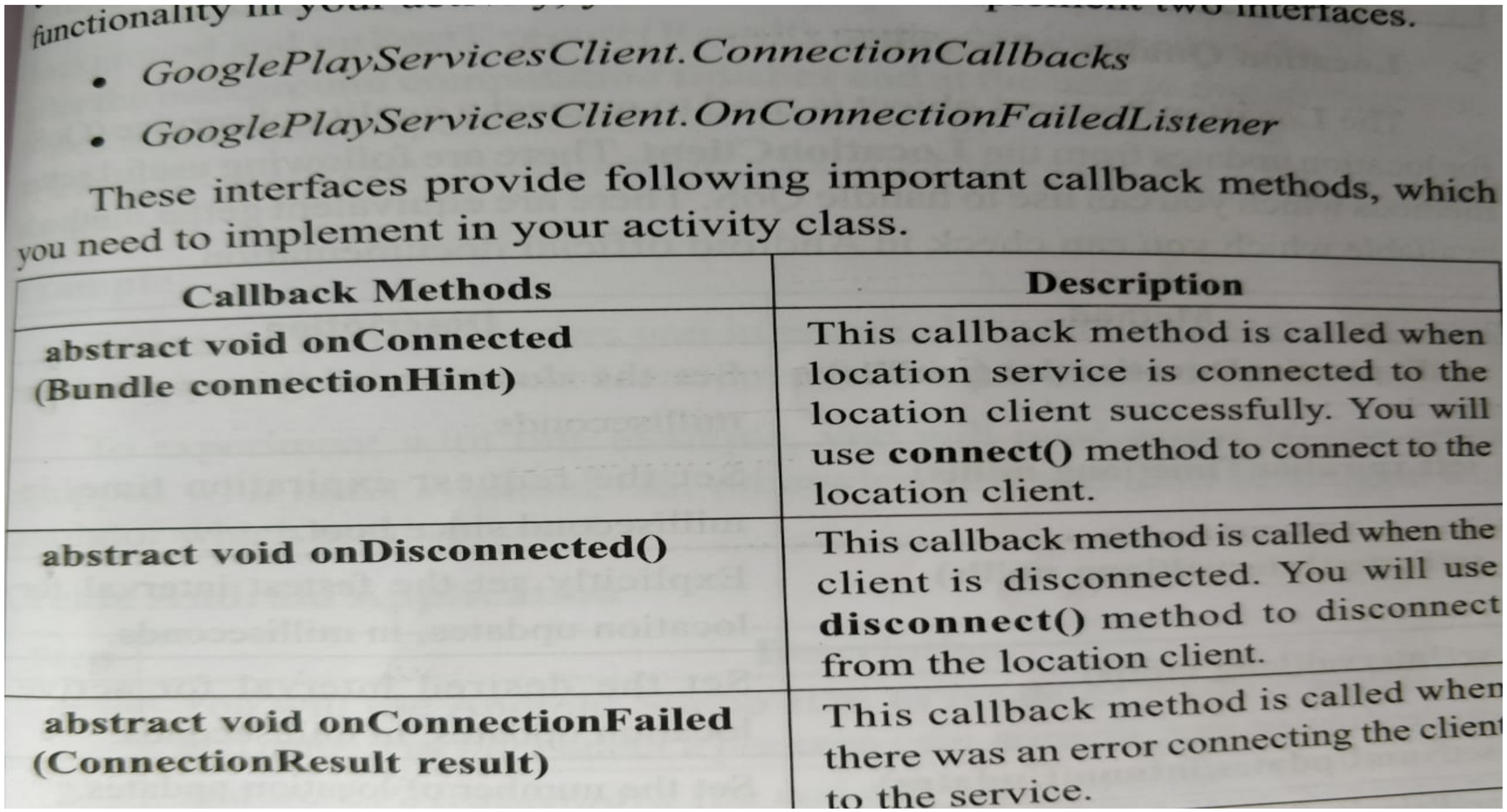
The location Object
The location object represents a geographic location which can consist of a lattitude, longitude, timestamp and other information like velocity, bearing,altitude etc,

Method	Description
float distanceTo(Location dest)	Returns the approximate distance in meters between this location and the given location.
float get Accuracy()	Get the estimated accuracy of this location, in meters.
double getAltitude()	Get the altitude if available, in meters above sea level.
float getBearing()	Get the bearing, in degrees.
double getLatitude()	Get the latitude, in degrees.
double getLongitude()	Get the longitude, in degrees.
float getSpeed()	Get the speed if it is available, in meters/ second over ground.
void reset()	Clears the contents of the location.

Get the current location

To get the current location, create a **location client** which is **Location client object**, connect it to **Location Services** using **connect()** method and then call its **getLastLocation()** method.This method returns the most recent location in the form of location object that contains latitude and longitude coordinates and other info.

2 interfaces to be implemented



functionality in your ... two interfaces.

- *GooglePlayServicesClient.ConnectionCallbacks*
- *GooglePlayServicesClient.OnConnectionFailedListener*

These interfaces provide following important callback methods, which you need to implement in your activity class.

Callback Methods	Description
abstract void onConnected (Bundle connectionHint)	This callback method is called when location service is connected to the location client successfully. You will use **connect()** method to connect to the location client.
abstract void onDisconnected()	This callback method is called when the client is disconnected. You will use **disconnect()** method to disconnect from the location client.
abstract void onConnectionFailed (ConnectionResult result)	This callback method is called when there was an error connecting the client to the service.

You should create the location client in **onCreate()** method of your activity class, then connect it in **onStart()**, so that Location Services maintains the current location while your activity is fully visible.

You should disconnect the client in **onStop()** method, so that when your app is not visible, Location Services is not maintaining the current location. This helps in saving battery power up-to a large extent.

Get the Updated Location
If you are willing to have location updates, then apart from above mentioned interfaces, you will need to implement **LocationListener** interface as well. This interface provide following callback method, which you need to implement in your activity class −

abstract void onLocationChanged(Location location)
This callback method is used for receiving notifications from the LocationClient when the location has changed.

Location Quality of Service
The **LocationRequest** object is used to request a quality of service (QoS) for location updates from the **LocationClient**.

Now for example, if your application wants high accuracy location it should create a location request with **setPriority(int)** set to **PRIORITY_HIGH_ACCURACY** and **setInterval(long)** to 5 seconds. You can also use bigger interval and/or other priorities like **PRIORITY_LOW_POWER** for to request **"city" level accuracy** or **PRIORITY_BALANCED_POWER_ACCURACY** for "**block"** level accuracy.

Some useful setter methods to handle QoS

Sr.No.	Method & Description
1	**setExpirationDuration(long millis)** Set the duration of this request, in milliseconds.
2	**setExpirationTime(long millis)** Set the request expiration time, in millisecond since boot.
3	**setFastestInterval(long millis)** Explicitly set the fastest interval for location updates, in milliseconds.
4	**setInterval(long millis)** Set the desired interval for active location updates, in milliseconds.
5	**setNumUpdates(int numUpdates)** Set the number of location updates.
6	**setPriority(int priority)** Set the priority of the request.

Displaying a Location Address

Once you have **Location** object, you can use **Geocoder.getFromLocation()** method to get an address for a given latitude and longitude. This method is synchronous, and may take a long time to do its work, so you should call the method from the **doInBackground()** method of an **AsyncTask** class.

Step	Description
1	You will use Android studio IDE to create an Android application and name it as *pgm* under a package *com.example.pgm.myapplication*.
2	add *src/GPSTracker.java* file and add required code.
3	Modify *src/MainActivity.java* file and add required code as shown below to take care of getting current location and its equivalent address.
4	Modify layout XML file *res/layout/activity_main.xml* to add all GUI components which include three buttons and two text views to show location/address.
5	Modify *res/values/strings.xml* to define required constant values
6	Modify *AndroidManifest.xml* as shown below
7	Run the application to launch Android emulator and verify the result of the changes done in the application.

Google Maps

Android allows us to integrate google maps in our application. You can show any location on the map , or can show different routes on the map e.t.c. You can also customize the map according to your choices.

Google Map - Layout file

Now you have to add the map fragment into xml layout file.
Its syntax is given below −

```
<fragment android:id="@+id/map"
android:name="com.google.android.gms.maps.MapFragment"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

Google Map - AndroidManifest file

The next thing you need to do is to add some permissions along with the Google Map API key in the AndroidManifest.XML file.
Its syntax is given below −

```
<!--Permissions-->
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permissionandroid:name="com.google.android.providers.gsf.permission
. READ_GSERVICES" />
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!--Google MAP API key-->
<meta-data android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyDKymeBXNeiFWY5jRUejv6zItpmr2MVyQ0" />
```

Customizing Google Map

You can easily customize google map from its default view , and change it according to your demand.

Adding Marker

You can place a maker with some text over it displaying your location on the map. It can be done by via **addMarker()** method.

Its syntax is given below –

```
final LatLng myprogram = new LatLng(21 , 57);
Marker TP = googleMap.addMarker(new MarkerOptions()
.position(myprogram).title(“Map"));
```

Changing Map Type

You can also change the type of the MAP.

There are four different types of map and each give a different view of the map. These types are Normal,Hybrid,Satellite and terrain. You can use them as below

```
googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Enable/Disable zoom

You can lso enable or disable the zoom gestures in the map by calling the **setZoomControlsEnabled(boolean)** method. Its syntax is given below –

```
googleMap.getUiSettings().setZoomGesturesEnabled(true);
```

Apart from these customization, there are other methods available in the GoogleMap class , that helps you more customize the map. They are listed below –

Sr.No	Method & description
1	**addCircle(CircleOptions options)** This method add a circle to the map
2	**addPolygon(PolygonOptions options)** This method add a polygon to the map
3	**addTileOverlay(TileOverlayOptions options)** This method add tile overlay to the map
4	**animateCamera(CameraUpdate update)** This method Moves the map according to the update with an animation
5	**clear()** This method removes everything from the map.

6	**getMyLocation()** This method returns the currently displayed user location.
7	**moveCamera(CameraUpdate update)** This method repositions the camera according to the instructions defined in the update
8	**setTrafficEnabled(boolean enabled)** This method Toggles the traffic layer on or off.
9	**snapshot(GoogleMap.SnapshotReadyCallback callback)** This method Takes a snapshot of the map
10	**stopAnimation()** This method stops the camera animation if there is one in progress