

## Correction TP05

Matière : ATELIER DEVELOPPEMENT MOBILE

Classes : SEM31

## Exercice 1

```
package com.ex1;
//imports
public class MainActivity extends Activity {
    private TextView tvListe;
    private LocationManager lmg;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        init();
    }
    private void init() {
        tvListe = (TextView) findViewById(R.id.tvListe);
        lmg = (LocationManager) getSystemService(LOCATION_SERVICE);
    }
    @Override
    protected void onResume() {
        remplir();
        super.onResume();
    }
    private void remplir() {
        String res = "";
        res = "Liste des fournisseurs:\n";
        List<String> liste = lmg.getAllProviders();
        for (String nom : liste) {
            res+="\t"+nom+"\n";
        }
        res += "Liste des fournisseurs actifs:\n";
        List<String> listeActif = lmg.getProviders(true);
        for (String nom : listeActif) {
            res+="\t"+nom+"\n";
        }
        tvListe.setText(res);
    }
}
```

## Exercice2

```
package com.ex2;
//imports
public class MainActivity extends AppCompatActivity {

    private static final int REQUEST_ID_ACCESS_COURSE_FINE_LOCATION = 1;
    private TextView tvPosition;
    private LocationManager mg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        init();
    }

    private void init() {
        tvPosition = (TextView) findViewById(R.id.tvPosition);
        mg = (LocationManager) getSystemService(LOCATION_SERVICE);
        ajouterEcouteur();
    }

    private void ajouterEcouteur() {
        if (verifierPermissions())
            demanderPosition();
        else {
            if (Build.VERSION.SDK_INT >= 23) {
                int accessCoarsePermission
                    = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION);
                int accessFinePermission
                    = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION);
                if (accessCoarsePermission != PackageManager.PERMISSION_GRANTED
                    || accessFinePermission != PackageManager.PERMISSION_GRANTED) {
                    // The Permissions to ask user.
                    String[] permissions = new String[]{Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION};
                    // Show a dialog asking the user to allow the above permissions.
                    ActivityCompat.requestPermissions(this, permissions,
REQUEST_ID_ACCESS_COURSE_FINE_LOCATION);
                    return;
                }
            }
        }
    }

    private boolean verifierPermissions() {
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
```

```

        // ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        // public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                     int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return false;
    }
    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case REQUEST_ID_ACCESS_FINE_LOCATION:
            // Note: If request is cancelled, the result arrays are empty.
            // Permissions granted (read/write).
            if (grantResults.length > 1
                && grantResults[0] == PackageManager.PERMISSION_GRANTED
                && grantResults[1] == PackageManager.PERMISSION_GRANTED) {
                demanderPosition();
            }
            // Cancelled or denied.
            else {
                Toast.makeText(this, "Permissions non accordées!", Toast.LENGTH_LONG).show();
            }
            break;
    }
}

private void demanderPosition() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        // ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        // public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                     int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }
    mg.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0, new LocationListener() {
        @Override
        public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
        }

        @Override
        public void onProviderEnabled(String arg0) {
        }
    }
}

```

```

@Override
public void onProviderDisabled(String arg0) {
}

@Override
public void onLocationChanged(Location arg0) {
    afficher(arg0);
}
});

}

private void afficher(Location location) {
    String res = "";

    res = "Position:\n";
    res += "\tLongitude: " + location.getLongitude() + "\n";
    res += "\tLatitude: " + location.getLatitude() + "\n";

    tvPosition.setText(res);
}
}

```

## Exercice3

```
package com.ex3;
//imports

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    public static final float ISET_SFAX_LAT=34.756932f;
    public static final float ISET_SFAX_LONG=10.772176f;

    public static final float ISET_RADES_LAT=36.760506f;
    public static final float ISET_RADES_LONG=10.270365f;

    public static final float ISET_MAHDIAT_LAT=35.522674f;
    public static final float ISET_MAHDIAT_LONG=11.030392f;

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        marquerIssets();
    }
    private void marquerIssets() {
        MarkerOptions marker1 = new MarkerOptions().position(new LatLng(ISET_SFAX_LAT,
ISET_SFAX_LONG));
        marker1.title("Iset Sfax");
        marker1.snippet("Sfax");
        mMap.addMarker(marker1);

        MarkerOptions marker2 = new MarkerOptions().position(new LatLng(ISET_SFAX_LAT,
ISET_SFAX_LONG));
        marker2.title("Iset Radès");
        marker2.snippet("Radès");
        mMap.addMarker(marker2);
    }
}
```

```

        MarkerOptions marker3 = new MarkerOptions().position(new LatLng(ISET_SFAX_LAT,
ISET_SFAX_LONG));
        marker3.title("Iset Mahdia");
        marker3.snippet("Mahdia");
        mMap.addMarker(marker3);

        zoomIsetSfax();
    }
    private void zoomIsetSfax() {
        CameraPosition cameraPosition = new CameraPosition.Builder()
            .target(new LatLng(ISET_SFAX_LAT, ISET_SFAX_LONG)).zoom(6).build();
        mMap.animateCamera(CameraUpdateFactory
            .newCameraPosition(cameraPosition));
    }
}

```

## Exercice4

```

//imports
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private static final int REQUEST_ID_ACCESS_FINE_LOCATION = 1;
    private LocationManager mg;

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        init();
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
    }
    private void init() {
        mg = (LocationManager) getSystemService(LOCATION_SERVICE);
        ajouterEcouteur();
    }
}

```

```

private void ajouterEcouteur() {
    if (verifierPermissions())
        demanderPosition();
    else {
        if (Build.VERSION.SDK_INT >= 23) {
            int accessCoarsePermission
                = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION);
            int accessFinePermission
                = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION);
            if (accessCoarsePermission != PackageManager.PERMISSION_GRANTED
                || accessFinePermission != PackageManager.PERMISSION_GRANTED) {
                // The Permissions to ask user.
                String[] permissions = new String[]{Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION};
                // Show a dialog asking the user to allow the above permissions.
                ActivityCompat.requestPermissions(this, permissions,
REQUEST_ID_ACCESS_COURSE_FINE_LOCATION);
                return;
            }
        }
    }
}

private boolean verifierPermissions() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //   ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                           int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return false;
    }
    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case REQUEST_ID_ACCESS_COURSE_FINE_LOCATION:
            // Note: If request is cancelled, the result arrays are empty.
            // Permissions granted (read/write).
            if (grantResults.length > 1
                && grantResults[0] == PackageManager.PERMISSION_GRANTED
                && grantResults[1] == PackageManager.PERMISSION_GRANTED) {
                demanderPosition();
            }
    }
}

```

```

        // Cancelled or denied.
        else {
            Toast.makeText(this, "Permissions non accordées!", Toast.LENGTH_LONG).show();
        }
        break;
    }

}

private void demanderPosition() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //    ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //    public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                           int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }
    mg.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0, new LocationListener() {
        @Override
        public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
        }

        @Override
        public void onProviderEnabled(String arg0) {
        }

        @Override
        public void onProviderDisabled(String arg0) {
        }

        @Override
        public void onLocationChanged(Location arg0) {
            marquerPoint (arg0);
            zoomDernierPoint(arg0);
        }
    });
}

private void marquerPoint(Location location) {
    MarkerOptions marker1 = new MarkerOptions().position(new
    LatLng(location.getLatitude(),location.getLongitude()));
    marker1.title("Point");
    marker1.snippet("Point");
    mMap.addMarker(marker1);
}
}

```



```
private void zoomDernierPoint(Location location) {  
    CameraPosition cameraPosition = new CameraPosition.Builder()  
        .target(new LatLng(location.getLatitude(),location.getLongitude())).zoom(6).build();  
    mMap.animateCamera(CameraUpdateFactory  
        .newCameraPosition(cameraPosition));  
}  
  
}
```

## Exercice 5

```
package com.ex5;
//imports
public class Param extends Activity {
    private EditText edTitre;
    private EditText edMessage;
    private Button btnValider;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.param);
        init();
    }
    private void init() {
        edTitre = (EditText) findViewById(R.id.edTitre);
        edMessage = (EditText) findViewById(R.id.edMessage);
        btnValider = (Button) findViewById(R.id.btnValider);
        ajouterEcouteur();
    }
    private void ajouterEcouteur() {
        btnValider.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                valider();
            }
        });
    }
    protected void valider() {
        Intent i = new Intent();
        i.putExtra("titre", edTitre.getText());
        i.putExtra("message", edMessage.getText());
        setResult(RESULT_OK, i);
        finish();
    }
}

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private LatLng dernierLatlong;
    private static final int PARAM=1;

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
}
```

```

/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.
 * This is where we can add markers or lines, add listeners or move the camera. In this case,
 * we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered once the user has
 * installed Google Play services and returned to the app.
 */
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    ajouterEcouteur();
}

private void ajouterEcouteur() {
    googleMap.setOnMapLongClickListener(new OnMapLongClickListener() {

        @Override
        public void onMapLongClick(LatLng arg0) {
            paramettrer(arg0);

        }
    });
}

protected void paramettrer(LatLng arg0) {
    dernierLatlong=arg0;
    Intent i = new Intent(MainActivity.this,Param.class);
    startActivityForResult(i, PARAM);
}

protected void marquer(String titre, String message) {
    MarkerOptions marker = new MarkerOptions().position(dernierLatlong);
    marker.title(titre);
    marker.snippet(message);

    // adding marker
    mMap.addMarker(marker);
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PARAM && resultCode == RESULT_OK) {
        String titre = data.getExtras().getCharSequence("titre").toString();
        String message = data.getExtras().getCharSequence("message").toString();
        marquer(titre, message);
        zoomDernierPoint() ;
    }
    super.onActivityResult(requestCode, resultCode, data);
}

```

```
}  
private void zoomDernierPoint() {  
    CameraPosition cameraPosition = new CameraPosition.Builder()  
        .target(dernierLatlong).zoom(6).build();  
    mMap.animateCamera(CameraUpdateFactory  
        .newCameraPosition(cameraPosition));  
}  
  
}
```