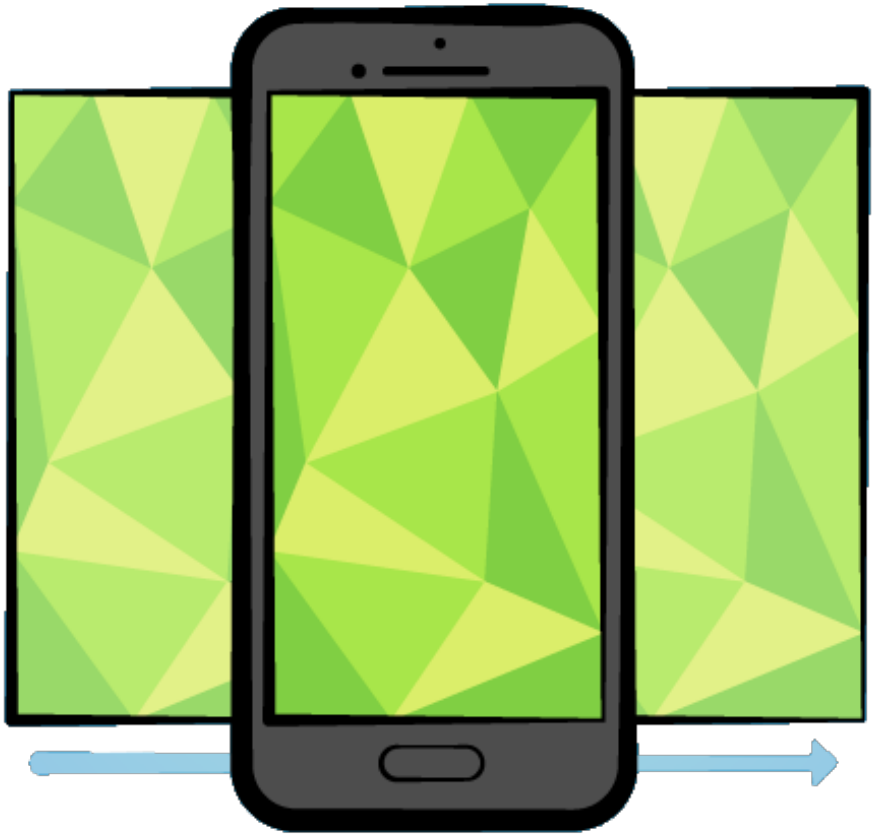
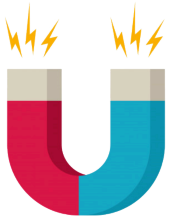
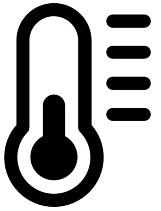
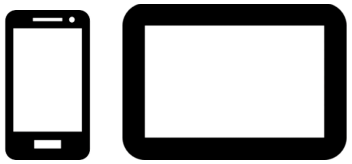
[TD5] ViewPager2 & Capteurs







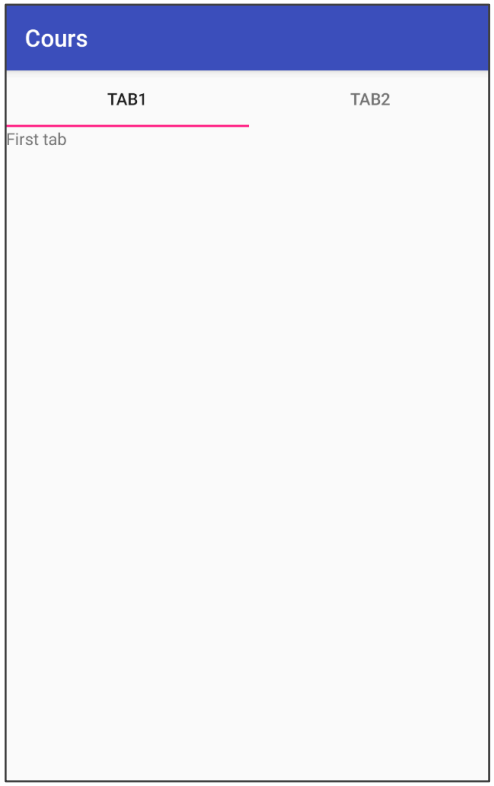


PRÉSENTATION

Nous allons créer des pages gérées par des onglets en nous servant de **Fragments** et des classes **ViewPager2** et **TabLayout**.

Un **ViewPager** est un composant permettant de naviguer horizontalement entre différentes portions du contenu d'une application sur un seul et même écran.

Dans un ViewPager il y aura différents fragments matérialisés par des pages.



VIEWPAGER : CONFIGURATION

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.cfc.td3"
        minSdkVersion 23
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation "androidx.constraintlayout:constraintlayout:2.0.0-beta4"
    implementation 'com.google.android.material:material:1.2.0-alpha04'
}
```

Permet l'utilisation de JAVA 8

Gérer la dépendance à la bibliothèque des ViewPager2

NB : la version doit être cohérente avec celle de *compileSdkVersion*

VIEWPAGER : UTILISATION

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportFragmentManager().beginTransaction()
                .add(R.id.main, new TabsManager())
                .commit();
    }

}
```

L'activité principale va être en charge de placer un fragment dans un conteneur.

Ce fragment, TabsManager, sera en charge de gérer les deux pages au sein d'un ViewPager et cela grâce à un FragmentStateAdapter qui fera fonctionner l'ensemble correctement.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

</LinearLayout>
```

VIEWPAGER : UTILISATION

```
public class Tab extends Fragment {

    private String name;

    Tab(String name) {
        this.name = name;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_tab, container, false);
        TextView text = view.findViewById(R.id.name);
        text.setText(name);
        return view;
    }
}
```

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.cfc.cours.Tab">

    <TextView
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</FrameLayout>
```

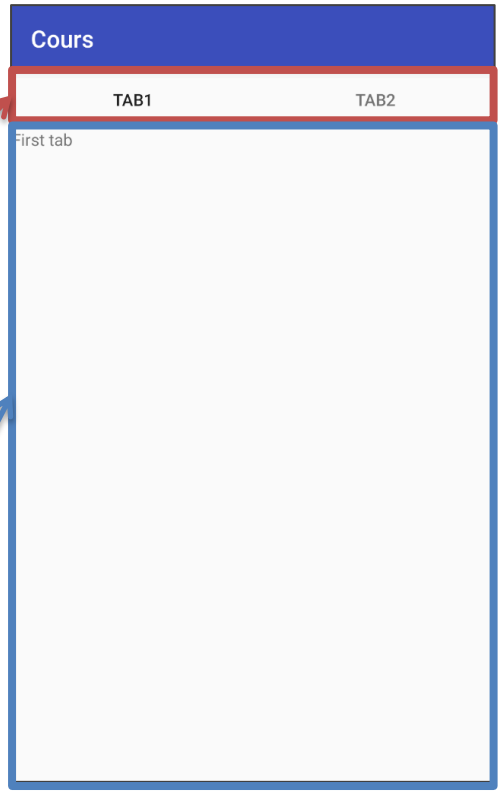
Le fragment Tab permettra de construire les deux pages.

VIEWPAGER : UTILISATION

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.material.tabs.TabLayout
        android:id="@+id/tablayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

La partie statique de TabsManager contiendra :

- un TabLayout qui créera les onglets au-dessus
- un ViewPager permettant la visualisation horizontale des différentes pages.



```
public class TabsManager extends Fragment {

    private TabsStateAdapter tabsStateAdapter;
    private ViewPager2 viewPager;
    List<String> tabs = Arrays.asList("Tab1","Tab2");

    @Override
    public View onCreateView(LayoutInflater inflater,
      ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_tabs_manager, container, false);
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        tabsStateAdapter = new TabsStateAdapter(this);
        viewPager = view.findViewById(R.id.viewpager);
        viewPager.setAdapter(tabsStateAdapter);
        TabLayout tabLayout = view.findViewById(R.id.tablayout);
        new TabLayoutMediator(tabLayout, viewPager,
                (tab, position) -> tab.setText(tabs.get(position))
        ).attach();
    }

    private class TabsStateAdapter extends FragmentStateAdapter {

        public TabsStateAdapter(Fragment fragment) {
            super(fragment);
        }

        @Nullable
        @Override
        public Fragment createFragment(int position) {
            switch (position) {
                case 0:
                    return new Tab("First tab");
                case 1:
                    return new Tab("Second tab");
            }
            return null;
        }

        @Override
        public int getItemCount() {
            return 2;
        }
    }

}
```

createFragment : retourne l'élément à une position donnée.

getItemCount : retourne le nombre de vues disponibles

VIEWPAGER : COMMUNICATION ENTRE LES PAGES

La principale difficulté sera de faire communiquer les pages (e.g. des fragments) entre eux. En effets, ces dernières n'auront pas la connaissance des autres.

Les deux options les plus simples à mettre en œuvre seront :

- Que le Fragment TabsManager joue le rôle de médiateur en gardant une référence sur chaque fragment créé dans createFragment et que les fragments aient une référence sur le TabsManager.

- Que le fragment 1, qui doit communiquer avec le fragment 2, reçoive en paramètre de son constructeur la référence du fragment 2.

LES PERMISSIONS

Depuis Android Marshmallow (API 23) il existe deux catégories de permissions :

- **normales** : elles seront garanties si elles sont demandées dans le manifeste car elles ne concernant pas la vie privée de l'utilisateur (i.e. Internet)
- **sensibles** : nécessitent une demande au runtime via un popup car elles concernent la vie privée de l'utilisateur.

LES PERMISSIONS

Permission normale

Pour demander une permission normale il suffit de modifier le manifeste.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Dans cet exemple nous souhaitons utiliser la connexion internet.

Permission sensible

Une permission sensible doit aussi être déclarée dans le manifeste.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

ACCESS_FINE_LOCATION : position aussi précise que possible à partir des fournisseurs de géolocalisation disponibles (GPS, Wi-Fi et les données des réseaux de téléphonie mobile).

```
public class MainActivity extends AppCompatActivity {

    private static final int MY_PERMISSION = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        int permission = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION);
        if(permission != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    MY_PERMISSION);
        }
    }

    @Override
    public void onRequestPermissionsResult(int code, String perms[], int[] results) {
        switch (code) {
            case MY_PERMISSION :
                if(results.length > 0 && results[0] == PackageManager.PERMISSION_GRANTED)
                    Toast.makeText(getApplicationContext(), "ALLOW", Toast.LENGTH_SHORT).show();
                else
                    Toast.makeText(getApplicationContext(), "DENY", Toast.LENGTH_SHORT).show();
        }
    }
}
```

LES PERMISSIONS

Permission sensible

Il faut vérifier si la permission a été accordée.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    int permission = ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION );
    if(permission != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                MY_PERMISSION);
    }
}
```

Savoir si la permission a été accordée ou non

Demander l'autorisation si la permission n'a pas été accordée.

LES PERMISSIONS

Permission sensible

En redéfinissant la méthode **onRequestPermissionsResult** vous pourrez gérer l'acceptation ou le refus de la permission.

```
public void onRequestPermissionsResult(int code, String perms[], int[] results) {
      switch (code) {
         case MY_PERMISSION :
            if(results.length > 0 && results[0] == PackageManager.PERMISSION_GRANTED)
               Toast.makeText(getApplicationContext(), "ALLOW", Toast.LENGTH_SHORT).show();
            else
               Toast.makeText(getApplicationContext(), "DENY", Toast.LENGTH_SHORT).show();
      }
   }
```

CLASSE SENSORMANAGER

Tous les capteurs disponibles sur un appareil Android peuvent être gérés grâce à la classe **SensorManager**.

En fonction du matériel que vous utilisez, différents types de capteurs seront disponibles : accéléromètre, gyroscope, capteur de proximité, …

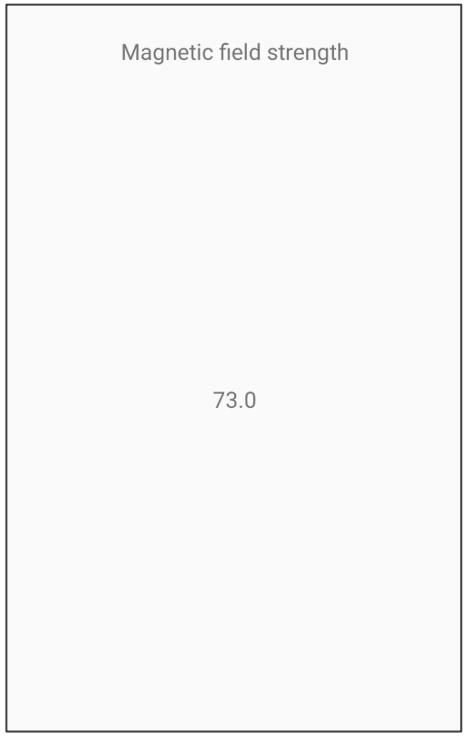
Vous pourrez connaître la liste des capteurs disponibles avec :

```
final SensorManager sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
List<Sensor> list = sensorManager.getSensorList(Sensor.TYPE_ALL);
for(Sensor sensor : list)
    Log.i("Sensor",sensor.getName());
```

MAGNÉTOMÈTRE

Ce capteur permet de connaître le champs magnétique s'appliquant sur l'appareil.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="30dp"
        android:text="Magnetic field strength"
        android:gravity="center"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/strength"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/message"
        app:layout_constraintBottom_toBottomOf="parent"
        android:gravity="center"
        android:textSize="20sp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



MAGNÉTOMÈTRE

```
public class MainActivity extends Activity {

    private TextView strength;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        strength = findViewById(R.id.strength);
    }

    @Override
    protected void onResume() {
        super.onResume();
        final SensorManager sensorManager =
        (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorManager.registerListener(magneticEventListener,
                sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    private final SensorEventListener magneticEventListener =
     new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
                double value = 0;
                for(int i=0;i<3;i++) {
                    value += event.values[i]*event.values[i];
                }
                strength.setText(Double.toString(Math.round(Math.sqrt(value))));
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {}
    };
}
```

MAGNÉTOMÈTRE

```
public class MainActivity extends Activity {

    private TextView strength;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        strength = findViewById(R.id.strength);
    }

    @Override
    protected void onResume() {
        super.onResume();
        final SensorManager sensorManager =
        (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorManager.registerListener(magneticEventListener,
                sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    private final SensorEventListener magneticEventListener =
     new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
                double value = 0;
                for(int i=0;i<3;i++) {
                    value += event.values[i]*event.values[i];
                }
                strength.setText(Double.toString(Math.round(Math.sqrt(value))));
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {}
    };
}
```

Récupérer une instance du gestionnaire de capteurs.

MAGNÉTOMÈTRE

```
public class MainActivity extends Activity {

    private TextView strength;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        strength = findViewById(R.id.strength);
    }

    @Override
    protected void onResume() {
        super.onResume();
        final SensorManager sensorManager =
        (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorManager.registerListener(magneticEventListener,
                sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    private final SensorEventListener magneticEventListener =
     new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
                double value = 0;
                for(int i=0;i<3;i++) {
                    value += event.values[i]*event.values[i];
                }
                strength.setText(Double.toString(Math.round(Math.sqrt(value))));
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {}
    };
}
```

Déclarer et implémenter une instance de la classe **SensorEventListener**.

MAGNÉTOMÈTRE

```
public class MainActivity extends Activity {

    private TextView strength;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        strength = findViewById(R.id.strength);
    }

    @Override
    protected void onResume() {
        super.onResume();
        final SensorManager sensorManager =
        (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorManager.registerListener(magneticEventListener,
                sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    private final SensorEventListener magneticEventListener =
     new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
                double value = 0;
                for(int i=0;i<3;i++) {
                    value += event.values[i]*event.values[i];
                }
                strength.setText(Double.toString(Math.round(Math.sqrt(value))));
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {}
    };
}
```

Lier l'instance du **SensorManager** au listener.

CAPTEURS DE POSITION

Exemple de code permettant de récupérer périodiquement la localisation d'un utilisateur et de l'afficher.

En fonction des besoins de l'application vous pourriez aussi :

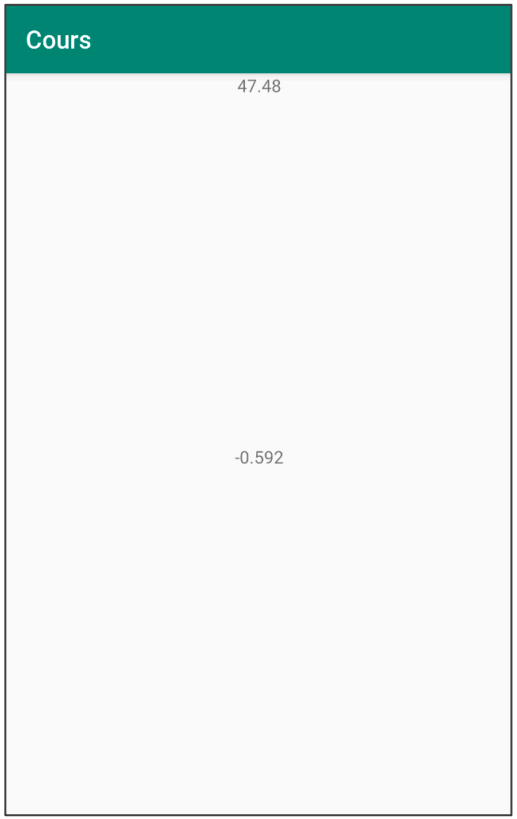
- Récupérer depuis un cache la dernière localisation

  Méthode : **getLastKnownLocation**

- Obtenir une seule fois la localisation

  Méthode : **requestSingleUpdate**

```
public class MainActivity extends AppCompatActivity implements LocationListener {
    private static final int MY_PERMISSION = 20;
    private TextView latitude;
    private TextView longitude;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        latitude = findViewById(R.id.latitude);
        longitude = findViewById(R.id.longitude);
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) !=
                PackageManager.PERMISSION_GRANTED ) {
            ActivityCompat.requestPermissions( this, new String[] {
                    Manifest.permission.ACCESS_FINE_LOCATION  },
                    MY_PERMISSION );
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) ==
                PackageManager.PERMISSION_GRANTED ){
            LocationManager locationManager =
              (LocationManager) getSystemService(Context.LOCATION_SERVICE);
            if(locationManager != null)
               locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER
               , 1000, 1, this);

        }
    }

    @Override
    public void onLocationChanged(Location location) {

latitude.setText(String.valueOf(Math.round(location.getLatitude()*1e3)/1e3));

longitude.setText(String.valueOf(Math.round(location.getLongitude()*1e3)/1e3));
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {}

    @Override
    public void onProviderEnabled(String provider) {}

    @Override
    public void onProviderDisabled(String provider) {}

}
```



```
public class MainActivity extends AppCompatActivity implements LocationListener {
    private static final int MY_PERMISSION = 20;
    private TextView latitude;
    private TextView longitude;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        latitude = findViewById(R.id.latitude);
        longitude = findViewById(R.id.longitude);
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) !=
                PackageManager.PERMISSION_GRANTED ) {
            ActivityCompat.requestPermissions( this, new String[] {
                    Manifest.permission.ACCESS_FINE_LOCATION  },
                    MY_PERMISSION );
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) ==
                PackageManager.PERMISSION_GRANTED ){
            LocationManager locationManager =
              (LocationManager) getSystemService(Context.LOCATION_SERVICE);
            if(locationManager != null)
               locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER
               , 1000, 1, this);

        }
    }

    @Override
    public void onLocationChanged(Location location) {

latitude.setText(String.valueOf(Math.round(location.getLatitude()*1e3)/1e3));

longitude.setText(String.valueOf(Math.round(location.getLongitude()*1e3)/1e3));
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {}

    @Override
    public void onProviderEnabled(String provider) {}

    @Override
    public void onProviderDisabled(String provider) {}

}
```

La permission ACCESS_FINE_LOCATION doit être obtenue (droit d'utiliser les composants de localisation les plus précis, cette permission inclut donc la permission ACCESS_COARSE_LOCATION).

NB : pensez à faire les modifications dans le Manifest.

```
public class MainActivity extends AppCompatActivity implements LocationListener {
    private static final int MY_PERMISSION = 20;
    private TextView latitude;
    private TextView longitude;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        latitude = findViewById(R.id.latitude);
        longitude = findViewById(R.id.longitude);
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) !=
                PackageManager.PERMISSION_GRANTED ) {
            ActivityCompat.requestPermissions( this, new String[] {
                    Manifest.permission.ACCESS_FINE_LOCATION  },
                    MY_PERMISSION );
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) ==
                PackageManager.PERMISSION_GRANTED ){
            LocationManager locationManager =
              (LocationManager) getSystemService(Context.LOCATION_SERVICE);
            if(locationManager != null)
               locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER
               , 1000, 1, this);

        }
    }

    @Override
    public void onLocationChanged(Location location) {

latitude.setText(String.valueOf(Math.round(location.getLatitude()*1e3)/1e3));

longitude.setText(String.valueOf(Math.round(location.getLongitude()*1e3)/1e3));
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {}

    @Override
    public void onProviderEnabled(String provider) {}

    @Override
    public void onProviderDisabled(String provider) {}

}
```

Récupérer une instance de LocationManager qui permet de gérer le service système de géolocalisation.

La méthode requestLocationUpdates va nous permettre de récupérer la localisation en prenant en paramètre :

- que l'on utilise le GPS
- délai minimum en millisecondes entre deux notifications de localisation
- la distance minimale en mètres qui doit avoir été parcourue entre deux notifications
- le listener (implémenté par la classe elle-même ici)

```
public class MainActivity extends AppCompatActivity implements LocationListener {
    private static final int MY_PERMISSION = 20;
    private TextView latitude;
    private TextView longitude;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        latitude = findViewById(R.id.latitude);
        longitude = findViewById(R.id.longitude);
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) !=
                PackageManager.PERMISSION_GRANTED ) {
            ActivityCompat.requestPermissions( this, new String[] {
                    Manifest.permission.ACCESS_FINE_LOCATION  },
                    MY_PERMISSION );
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        if (ContextCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_FINE_LOCATION ) ==
                PackageManager.PERMISSION_GRANTED ){
            LocationManager locationManager =
              (LocationManager) getSystemService(Context.LOCATION_SERVICE);
            if(locationManager != null)
               locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER
               , 1000, 1, this);

        }
    }

    @Override
    public void onLocationChanged(Location location) {

latitude.setText(String.valueOf(Math.round(location.getLatitude()*1e3)/1e3));

longitude.setText(String.valueOf(Math.round(location.getLongitude()*1e3)/1e3));
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {}

    @Override
    public void onProviderEnabled(String provider) {}

    @Override
    public void onProviderDisabled(String provider) {}

}
```

La méthode onLocationChanged est appelée quand la localisation de l'utilisateur est mise à jour. Elle nous permet ici de changer l'affichage de la localisation de l'utilisateur.

PM_EI5_TD5.pdf