1. **INTRODUCTION**
2. **OUTILS**
3. **Membres du groupe**
4. **Matériels**
5. **Ordinateur portale <ACER>**

* ***Type de processeur :*** *Intel(R) core (TM) i5-7200U*
* ***Fréquence du processeur :*** *2.5GHz*
* ***Taille de la RAM :*** *8GB*
* ***Adresse Mac :*** *3E-A0-92-60-94-92*

1. **Smartphone <Phantom6(A6)>**

* ***Mémoire interne :*** *32Gb*
* ***Processeur :*** *8x Arm cortex-A53 @ 1,95 GHz*
* ***RAM :*** *3GB*

1. **Logiciels**
2. **APPLICATION**

**EXERCICE 2 : SERVICE ET TACHE PERIODIQUE**

**Les Services  sont les** Applications ne présentant aucun affichage, mais s’exécutant en arrière-plan.

**A : création d’un service**

ClickDroit sur l’app> New > Service

**B : Lancement et arrêt du service**

Le lancement se fait dans la méthode « « OnCreate » »

Ainsi

intent = *new* Intent(MainActivity.*this*, MainService.*class*);  
 startService(intent) ;

L’arrêt du service se fait ainsi

stopService(intent);

**C : démarrage en mode ‘Sticky’**

Il suffit de redéfinir une méthode dans notre service comme ceci

@Override  
*public int* onStartCommand(Intent intent, *int* flags, *int* startId) {  
 *return* START\_STICKY;  
}

D: **Afficher une ligne de log**

L’affichage d’une ligne de log , se fait avec la méthode suivant, exemple

Log.*d*("MainActivity", "Création de l'activité");

**E : Créer une TimerTask**

TimerTask myTimerTask = *() ->* Log.*d*("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"," TimeTask Log : Salut, TheProgrammer My Time est lancé");  
 }.wait(30000);

**F: Lancer une timerTask**

myTimerTask.run();

**3 : Créer une tache asynchrone**

Il suffit juste de créer une classe et la faire hériter de « « AsynTack » », comme suit

*public class* MyAsyncTask *extends* AsyncTask {  
 @Override  
 *protected* Object doInBackground(Object[] objects) {  
  
 Log.*d*("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"," Tache asynchrone Je suis exécuté ");  
 *return null*;  
 }  
}

Ainsi la méthode “doInBackground” Est exécuté a chaque fois, que la tache asynchrone est exécuté .

**Exercice 3 : Conception D’interface utilisateur**

Xml est un langage de balisage, presque Semblable au HTML , Et permet de mettre en forme des information .

Pour ce devoir , Notre interface est géré dans le répertoire « « Ressources/layout/activity \_main.xml» ».

**A : Composant utilisé :**

1 : le « layout  LinearLayout » pour la disposition vertical et horizontal de nos composants.

2 : Le composant « TextView » pour l’affichage de nos texte

3 : Le composant « Button » pour l’affichage des boutons

4 : Le componsant « ToggleButton » pour les bouton « « Off » » Et « On »

**B : Reference de ces composant dans notre code java**

Pour récupérer notre butons textVieuw, nous utilisons la méthode « FindViewById » comme suit :

ToggleButton toggleBtn = findViewById(R.id.toggleButton);  
txtStatus = findViewById(R.id.txtStatus);  
txtAlert = findViewById(R.id.txtAlert);  
txtResult = findViewById(R.id.txtResult);

Sachant que “toggleButton” , est l’identifiant mit dans notre activité;

C : Réagir au click sur le bouton

ToggleButton.OnCheckedChangeListener onCheckListener = *new* CompoundButton.OnCheckedChangeListener() {  
 @Override  
 *public void* onCheckedChanged(CompoundButton buttonView, *boolean* isChecked) {  
 *if* (isChecked) {  
 txtStatus.setText("En Cours");  
 startService(intent);  
  
 }  
 *else* {  
 txtStatus.setText("Arreté");  
 stopService(intent); }}

On crée une variable listener , qui serra notre écoute, sur le bouton , donc, s’il est coché, on démarre le service, s’il n’est pas on arrête le service.

On modifie aussi la valeur du status ainsi

txtStatus.setText("En Cours");

comme vous le voyez ;

Enfin

toggleBtn.setOnCheckedChangeListener(onCheckListener);

qui valide donc notre listener en l’attribuant cette variable.

4 **: Exercice 4 : Accès aux services de localisation**

La localisation est un service utilisé, dans le développement Android,

que ce soit pour trouver un lieu, se rendre dans un lieu, ou regarder sa propre maison..

On trouve deux API de liées au concept de localisation

-- une API qui permet de localiser l'appareil

-- une API qui permet d'afficher les cartes

La localisation

on trouve tous les outils de localisation dans le package "" android.location"

On fais hériter notre classe de l’interface « **LocationListener** »

A: il faut demander la permission dans le manifest pour utiliser les fonctionnalité de localisation comme suit

<uses-permission *android:name=*"android.permission.ACCESS\_COARSE\_LOCATION" />  
<uses-permission *android:name=*"android.permission.INTERNET" /> *<!-- pour avoir une localisation précise -->*<uses-permission *android:name=*"android.permission.ACCESS\_FINE\_LOCATION" />

b : Faire appel a un nouveau service systeme pour acceder a ces fonctionalités ""LocationManager"" que nous avons recupéré ainsi

LocationManager lm = (LocationManager) getSystemService(LOCATION\_SERVICE);

**C : vérification si nos sévices de localisation sont actifs**

On s’abonne aux différents fournisseurs de localisation

*if* (lm.isProviderEnabled(LocationManager.GPS\_PROVIDER)) {  
 lm.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 10000, 0, *this*);  
 }

**D: Test des permissions**

Ici on teste les permissions pour chaque fournisseur , on verifie

Si l’utilisateur l’a activé sur son telephone

*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* ActivityCompat.*requestPermissions*(*this*, *new* String[]{  
 Manifest.permission.ACCESS\_FINE\_LOCATION,  
 Manifest.permission.ACCESS\_COARSE\_LOCATION  
 }, PERMS\_CALL\_ID);  
  
 *return*;  
}

C: Refinir la méthode OnLocationChanged pour avoir a chaque fois la nouvelle localisation

@Override  
*public void* onLocationChanged(Location location) {  
 *double* latitude = location.getLatitude();  
 *double* longitude = location.getLongitude();  
 Toast.*makeText*(*this*,"Location : " + latitude + " /" + longitude, Toast.LENGTH\_LONG).show();  
 *if* (googleMap != *null*) {  
 LatLng googleLocation = *new* LatLng(latitude, longitude);  
 googleMap.moveCamera(CameraUpdateFactory.*newLatLng*(googleLocation));  
 }  
}

**F: Affichage de la carte**

googleMap.moveCamera(CameraUpdateFactory.*zoomBy*(15));  
  
googleMap.setMyLocationEnabled(*true*);  
googleMap.addMarker(*new* MarkerOptions().position(*new* LatLng(33.87,11.48)).title("TheProgrammer Infini"));

La on positionne la camera, le zoom a 15%   
On Ajoute un marquer contenant, notre position , avec un titre ^^

Maintenant, il suffit juste de l’afficher sur notre interface graphique