# Etapes

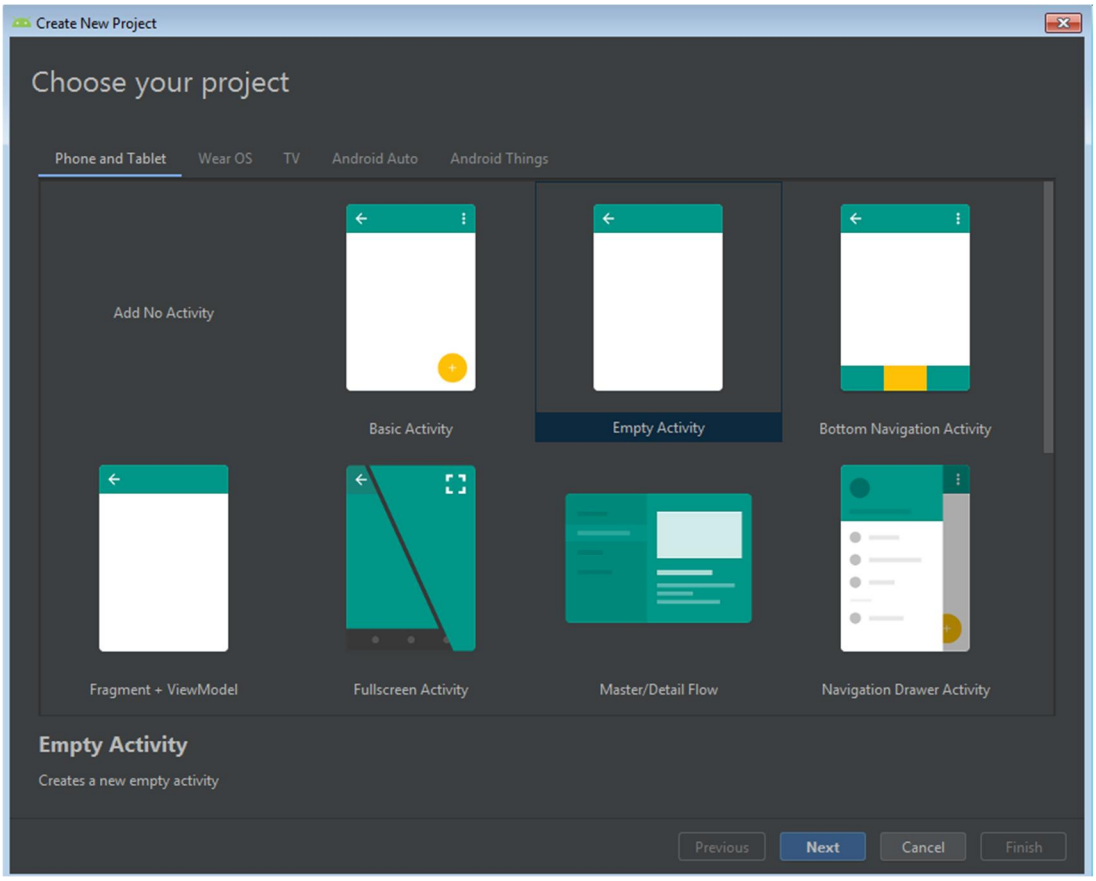
De programmation dans Java

# Etape 1 : Installation et configuration du Android Studio

- https://developer.android.com/studio/ - Un lien vers le tutoriel d'installation de l'Android Studio, ainsi que le téléchargement.
- Lorsque le téléchargement est terminé, lancer l'installateur puis cliquer suivant.
- Sélectionner « Android SDK » et « Android Virtual Device » si cela n'est pas déjà fait.
- Continuer puis installer.

# Etape 2 : Création d'un projet

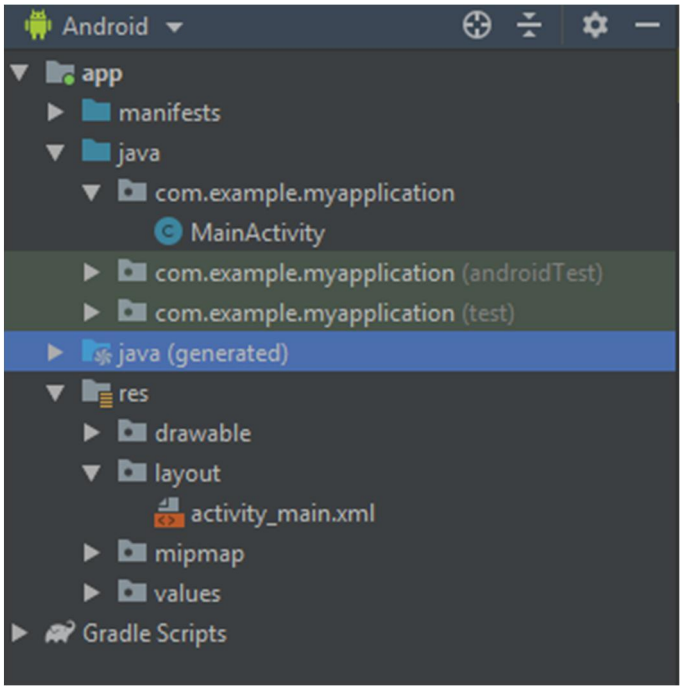
- Cliquer sur « New project »
- Choisir un Template d'activity. (Dans notre cas on va prendre un « Empty Activity », puis cliquer sur suivant.




- Configurer le projet comme vous voulez. Pour que l'application soit accessible au maximum de gens possible, sélectionner la plus ancienne version dans « Minimum API Level ». Par contre en faisant ceci vous pourriez ne pas avoir toutes les fonctionnalités des versions récentes.

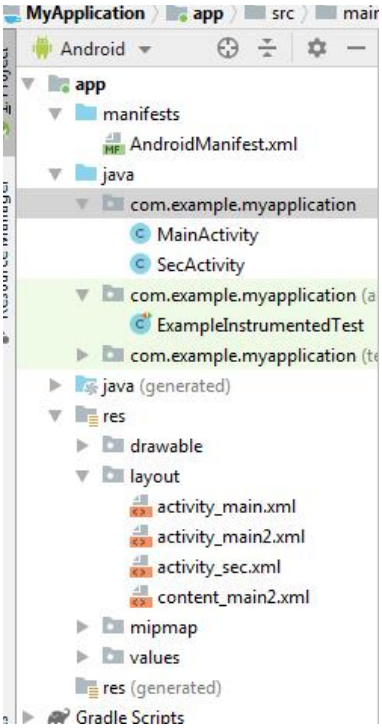
- Ceci sera l'architecture de votre application. Le dossier Java contiendra tous vos fichier java. Les sous-dossiers qui contiennent « (androidTest) » et « (test) » à côté de leurs noms ne doivent pas être modifiées car ils sont utilisés pour les tests.

  Le dossier « res » contiendra toutes les ressources, et c'est dans le dossier « layout » où vous aurez vos fichiers xml pour vos vues.




# Etape 3 : Changer l'activité

1. Dans le dossier Android clique droite sur package




2. New->Activity
3. Fichier Layout pour l'activité est créé automatiquement
4. Il y a 2 manières différentes de créer un layout à partir d'un autre :
   a. `android:onClick="NewEvent"` - Dans le fichier xml ajouter cette ligne où « NewEvent » est le nom de la méthode qui va créer nouvelle activity ;

```
Button test = findViewById(R.id.testBut);
test.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent myIntent = new Intent( packageContext: MainActivity.this, SecActivity.class);
        MainActivity.this.startActivity(myIntent);
    }
});
```

   b. Créer un observateur de bouton (OnClickListener()) et redéfinir la méthode onClick.
5. Intent est un outil de redirection vers une autre activity.

# Etape 4 : Incrémenter un nombre avec un bouton

1. Ajouter un bouton dans layout
2. Ajouter un « TextView » pour afficher le nombre

3. 
```
public class Main2Activity extends AppCompatActivity {
    private Button btnIncrement;
    TextView txtView;
    int counterValue = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        btnIncrement = findViewById(R.id.btnIncrement);
        txtView = findViewById(R.id.textView2);

        btnIncrement.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                counterValue++;
                txtView.setText(String.valueOf(counterValue));
            }
        });
    }
}
```
Voici le code pour incrémenter le nombre.
Il faut créer un « OnClickListener », ensuite dans la méthode onClick incrémenter une variable et copier sa valeur en format « string » dans le « TextView ».

# Etape 5 : Dynamiquement créer un nouveau layout

1. Ajouter un bouton dans le layout qui va servir de changer de layout.
2. Refaire un événement click comme dans les étapes précédentes.
3. Créer une linearlayout comme ceci :

```
LinearLayout newLayout = new LinearLayout( context: this);
newLayout.setOrientation(LinearLayout.VERTICAL);
newLayout.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT, LinearLayout.LayoutParams.WRAP_CONTENT));
```

4. Ensuite pour afficher le nouveau layout il faut écrire :
```
setContentView(newLayout);
```

# Etape 5 (SUITE) : Un nombre aléatoire

1. Importer la bibliothèque : `import java.util.Random;`
2. Définir la marge des nombre (Max et Min) ;
3. En suite initialiser un objet random :

```
final Random r = new Random();
```

4. Et faire next sur un integer :

```
int slctBtn = r.nextInt(max)+1;
```

5. (max)+1 signifie qu'on va commencer de 1 inclusive et finir par valeur de « max » inclusive. :

# Etape 6 : Stockage des données

1. Il existe de différents types de stockage sur java android :
    a. Internal file storage : stocke les fichiers privés sur l'appareil ;
    b. External file storage : stocke les fichiers sur un système de fichiers partagé ;
    c. Shared preferences : stocke les données primitives sous forme des clés-valeurs ;
    d. Database : base de données privé
2. Dans notre cas on va utiliser la base de données, plus précisément SQLite.
    a. Créez une classe statique « myDbHelper » qui hérite la classe **SQLiteOpenHelper**

```
static class myDbHelper extends SQLiteOpenHelper
```

    b. Ajoutez des packages :

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
```

    c. Créez le constructeur :

```
public DatabaseHelper(Context context) {
    super(context,DATABASE_NAME, factory: null, version: 2);
}
```

    d. Implémentez les fonctions « onCreate » et « onUpgrade ». La méthode « onCreate » sera appelée automatiquement lorsque la base de donnée va être crée pour la première fois. La méthode « onUpgrade » est appelée lorsque la version définit dans le « super » du constructeur est incrémenté.

```
@Override
public void onCreate(SQLiteDatabase db) {

}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

}
```

e. Créez une table en ajoutant ce code dans la méthode « onCreate » :

```
db.execSQL("CREATE TABLE " + EVALUATIONS_TABLE_NAME +" ( id INTEGER PRIMARY
KEY AUTOINCREMENT, Nom VARCHAR, Valeur INTEGER)");
```

f. Créez la méthode pour l'insertion des données :

```
public long insert(String name, Integer value) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("Nom", name);
    contentValues.put("Valeur", value);
    long id = db.insert(EVALUATIONS_TABLE_NAME, nullColumnHack: null, contentValues);
    return id;
}
```

g. Créez la méthode pour récupérer les 5 meilleures résultats :

```
public HashMap getTopFive(){
    SQLiteDatabase db = this.getWritableDatabase();
    HashMap<String, Integer> hashMap = new HashMap<>();
    Cursor res = db.rawQuery( sql: "SELECT * FROM "+ EVALUATIONS_TABLE_NAME + " ORDER BY Valeur DESC LIMIT 5", selectionArgs: null);
    res.moveToFirst();
    while(!res.isAfterLast()) {
        hashMap.put(res.getString(res.getColumnIndex( columnName: "Nom")), res.getInt(res.getColumnIndex( columnName: "Valeur")));
        res.moveToNext();
        Log.d( tag: "rows", msg: "newrow");
    }
    return hashMap;
}
```

# Etape 7 : Senseurs

1. Utiliser tous ces imports :

```
import androidx.appcompat.app.AppCompatActivity;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
```

2. Danss l'activité dans la laquelle vous voulez faire votre senseur, il faut que la classe hérite de AppCompatActivity et implémente

```
public class MainActivity extends AppCompatActivity implements SensorEventListener {
```

SensorEventListener

3. Définir ces variables :

```
private SensorManager sensorManager;
private Sensor accelerometer;

private long lastUpdate = 0;
private float last_x, last_y, last_z;
private static final int SHAKE_THRESHOLD = 600;
```

4. Il faut initialiser le senseur. Du coup dans la méthode onCreate mettre :

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

    sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    sensorManager.registerListener( listener: this, accelerometer , SensorManager.SENSOR_DELAY_NORMAL);
}
```

5. Créer la méthode onSensorChanged qui va être appelé à chaque fois que l'accéléromètre à détécté du changement. Dans la méthode, mettre :

```
@Override
public void onSensorChanged(SensorEvent event) {
    Sensor mySensor = event.sensor;

    if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];

        long curTime = System.currentTimeMillis();

        if ((curTime - lastUpdate) > 100) {
            long diffTime = (curTime - lastUpdate);
            lastUpdate = curTime;

            float speed = Math.abs(x + y + z - last_x - last_y - last_z)/ diffTime * 10000;

            if (speed > SHAKE_THRESHOLD) {
                Log.d( tag: "shake",  msg: "device shaked");
            }

        last_x = x;
        last_y = y;
        last_z = z;
        }
    }
}
```