

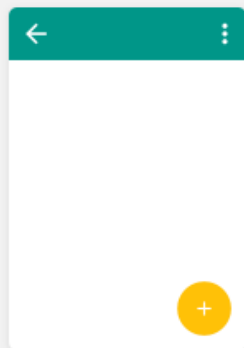


# TD1: Première Appli Hello WORLD



# Créez votre projet

## Configure your project



Basic Activity

### Name

My Application

### Package name

com.example.myapplication

### Save location

C:\Users\xi06\AndroidStudioProjects\MyApplication2

### Language

Java

### Minimum API level

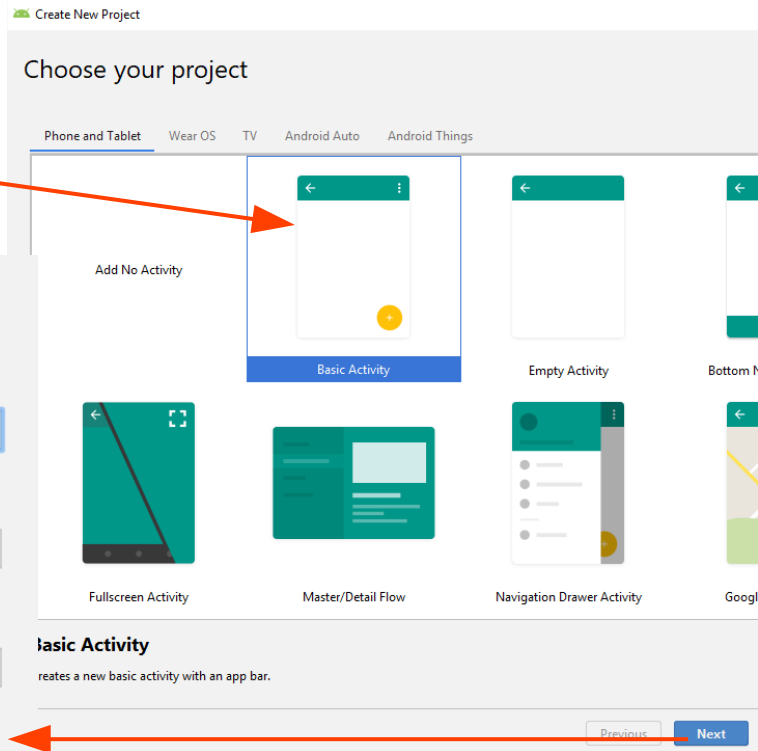
API 19: Android 4.4 (KitKat)

**i** Your app will run on approximately 95,3% of devices.

[Help me choose](#)

- This project will support instant apps
- Use AndroidX artifacts

Creates a new basic activity with an app bar.



MonAppli

Android

MonAppli C:\Users\xi06\AndroidStudioProjects\MonAppli

Gradle Scripts

Search Everywhere [Double Shift](#)

Go to File [Ctrl+Maj+N](#)

Recent Files [Ctrl+E](#)

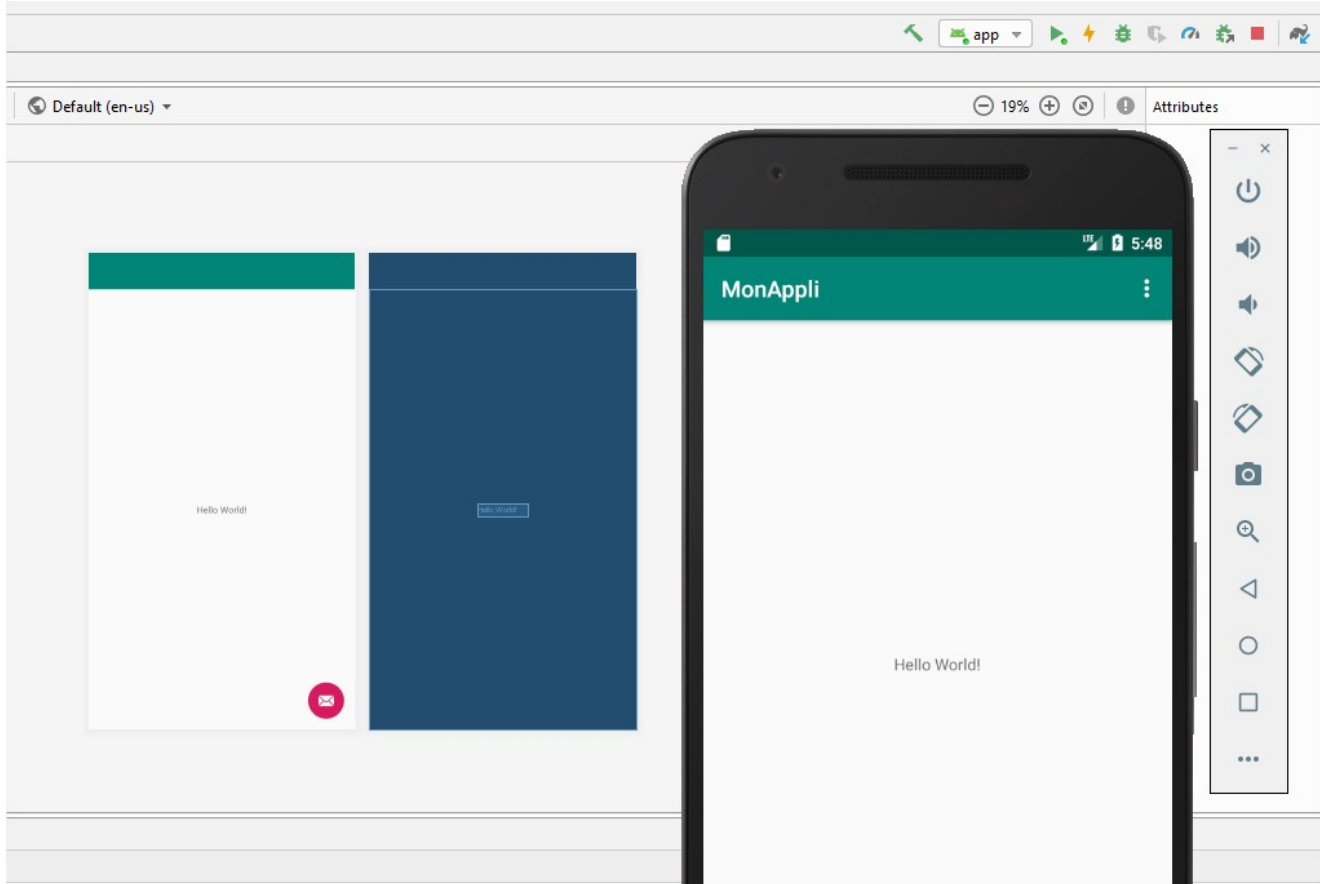
Navigation Bar [Alt+Origine](#)

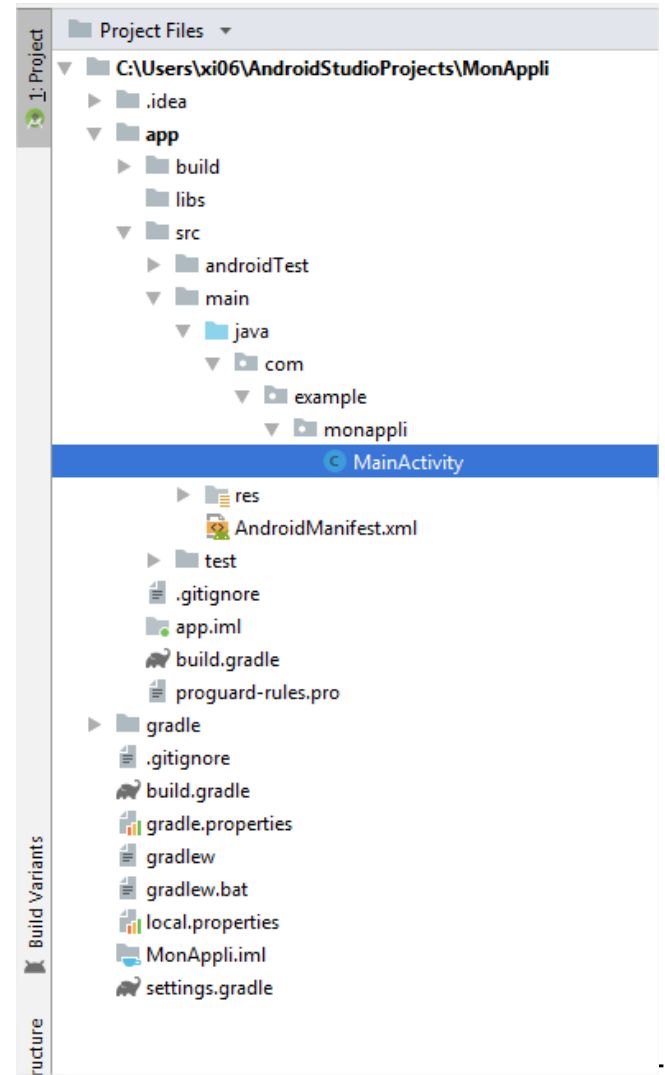
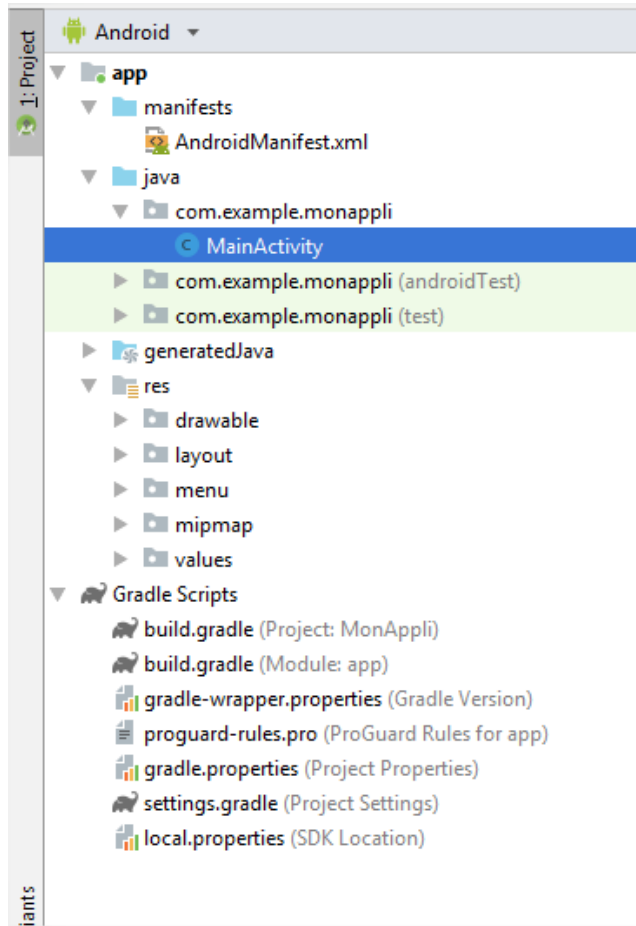
Drop files here to open

Build: Sync

Build Variants

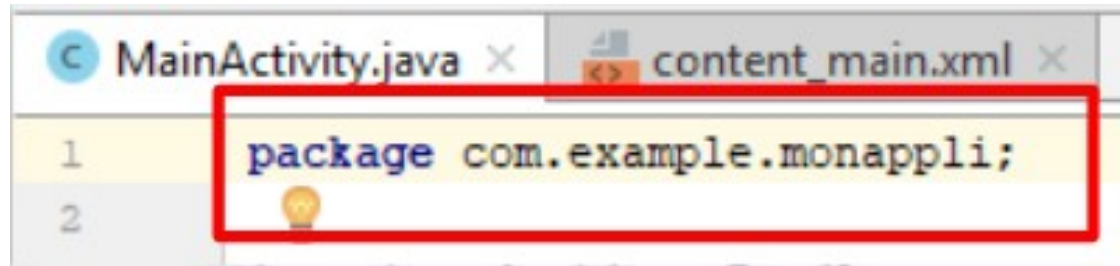
- MonAppli: syncing...
  - Run build C:\Users\xi06\AndroidStudioProjects\MonAppli
    - Load build
      - Run init scripts
      - Evaluate settings
    - Configure build





```
MainActivity.java x content_main.xml x
1 package com.example.monappli;
2
3 import android.os.Bundle;
4 import android.support.design.widget.FloatingActionButton;
5 import android.support.design.widget.Snackbar;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         Toolbar toolbar = findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = findViewById(R.id.fab);
22         fab.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
26                     .setAction(text: "Action", listener: null).show();
27             }
28         });
29     }
30
31     @Override
32     public boolean onCreateOptionsMenu(Menu menu) {
33         // Inflate the menu; this adds items to the action bar if it is present.
34         getMenuInflater().inflate(R.menu.menu_main, menu);
```

# Les packages



```
MainActivity.java x content_main.xml x  
1 package com.example.monappli;  
2
```

## Introduction

Les packages permettent de structurer l'ensemble des classes ou interfaces. Les packages sont similaires aux bibliothèques du langage C/C++. Un package est une unité (un fichier) regroupant des classes.

## Définition d'un package

Pour qu'une classe puisse se retrouver dans un package, il faut commencer en indiquant dans le fichier source, contenant les classes à regrouper, l'instruction `package` suivie du nom que l'on désire donner au package. Ainsi, toutes les classes contenues dans le fichier feront partie du package...

Ex.:

```
package nomPackage;
```

Attention package devra toujours être la première ligne !

## Unicité des noms de package:

Pour éviter des problèmes sur les noms de package (écrasement, ...), il peut être intéressant d'utiliser un identifiant unique dans le nom.

Une solution toute simple pour trouver un identifiant unique, vous pouvez utiliser votre nom de domaine (mais inversé pour trier plus facilement). Par exemple nicolas.free.fr donne fr.free.nicolas

Le nom de mon package serait alors fr.free.nicolas.nomPackage

## L'organisation hiérarchique des packages

Un package est une archive de classes (ZIP). L'extension est .JAR.

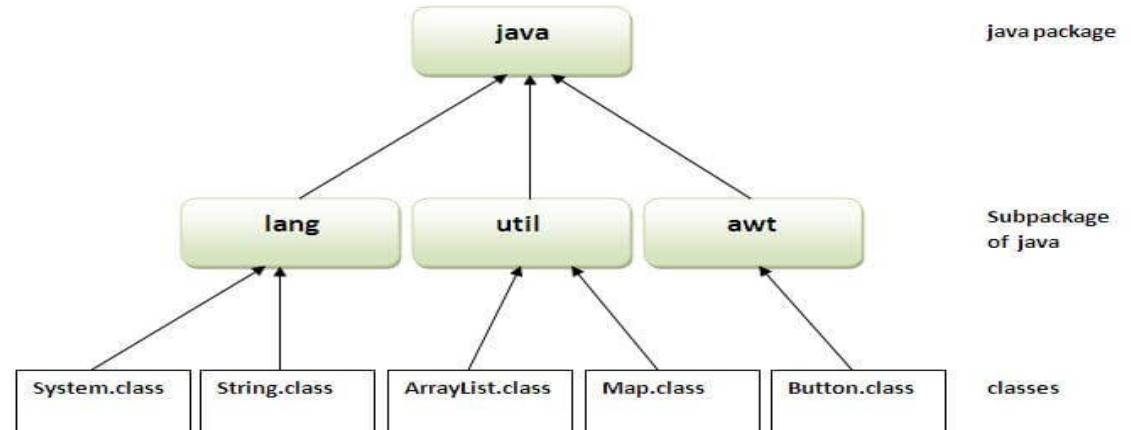
Les packages sont organisés sous forme de hiérarchie:

java->lang->annotation->Annotation.class

Avec java, lang et annotation qui sont des répertoires et Annotation.class qui est le fichier compilé du source correspondant.

Note :

Cela implique que vous retrouverez le source hors package dans la même arborescence que celle définie dans le package !





# Packages de la librairie standard

Les classes de la librairie standard sont organisées en packages. La figure reprend quelques packages de la librairie standard avec une courte description.

Le package `java.lang` est le package de base qui contient des classes qui seront presque toujours utilisées.

Package	Description
<code>java.applet</code>	Créer des programmes exécutables sur des pages web
<code>java.awt</code>	Créer des interfaces graphiques natives ( <u>lourdes</u> )
<code>java.io</code>	Effectuer des opérations d'entrées/sorties
<code>java.lang</code>	Contient les classes de base, par exemple <u>System</u> , <code>String</code> , etc.
<code>java.math</code>	Effectuer des calculs en précision arbitraire
<code>java.net</code>	Créer des programmes réseaux
<code>java.text</code>	Manipulation de texte, dates, nombres indépendamment de la langue
<code>java.util</code>	Contient des classes utilitaires comme <code>Date</code> par exemple
<code>javax.swing</code>	Créer des interfaces graphiques (légères)
<code>javax</code>	Manipulation de documents XML

**Figure 20.** Quelques packages de la librairie standard Java.

```
MainActivity.java x content_main.xml x
1 package com.example.monappli;
2
3 import android.os.Bundle;
4 import android.support.design.widget.FloatingActionButton;
5 import android.support.design.widget.Snackbar;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         Toolbar toolbar = findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = findViewById(R.id.fab);
22         fab.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
26                     .setAction(text: "Action", listener: null).show();
27             }
28         });
29     }
30
31     @Override
32     public boolean onCreateOptionsMenu(Menu menu) {
33         // Inflate the menu; this adds items to the action bar if it is present.
34         getMenuInflater().inflate(R.menu.menu_main, menu);
```

```
1 package com.example.monappli;
2
3 import android.os.Bundle;
4 import android.support.design.widget.FloatingActionButton;
5 import android.support.design.widget.Snackbar;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         Toolbar toolbar = findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = findViewById(R.id.fab);
22         fab.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
26                     .setAction("Action", listener: null).show();
27             }
28         });
29     }
30
31     @Override
32     public boolean onCreateOptionsMenu(Menu menu) {
33         // Inflate the menu; this adds items to the action bar if it is present.
34         getMenuInflater().inflate(R.menu.menu_main, menu);
35         return true;
36     }
37
38     @Override
39     public boolean onOptionsItemSelected(MenuItem item) {
40         // Handle action bar item clicks here. The action bar will
41         // automatically handle clicks on the Home/Up button, so long
42         // as you specify a parent activity in AndroidManifest.xml.
43         int id = item.getItemId();
44
45         //noinspection SimplifiableIfStatement
46         if (id == R.id.action_settings) {
47             return true;
48         }
49         return super.onOptionsItemSelected(item);
50     }
51 }
```

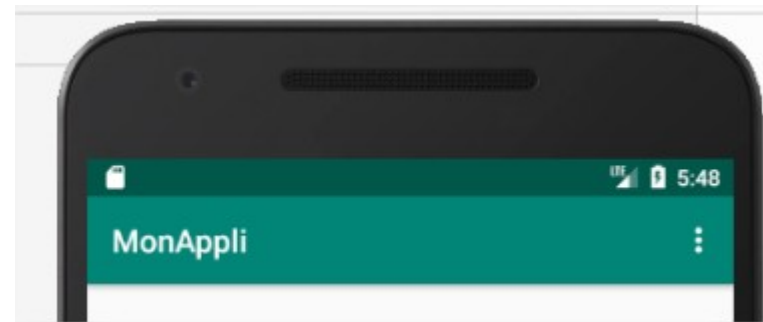
Android studio a généré automatiquement des objets lors de la création de notre projet.

Il y a beaucoup de choses et certaines seront expliquées plus tard.

Pour l'instant nous allons faire un peu de ménage dans tout cela. Vous allez modifier le code.

Supprimez les méthodes :

`onCreateOptionsMenu` et `onOptionsItemSelected` afin d'obtenir :



```
package com.example.monappli;

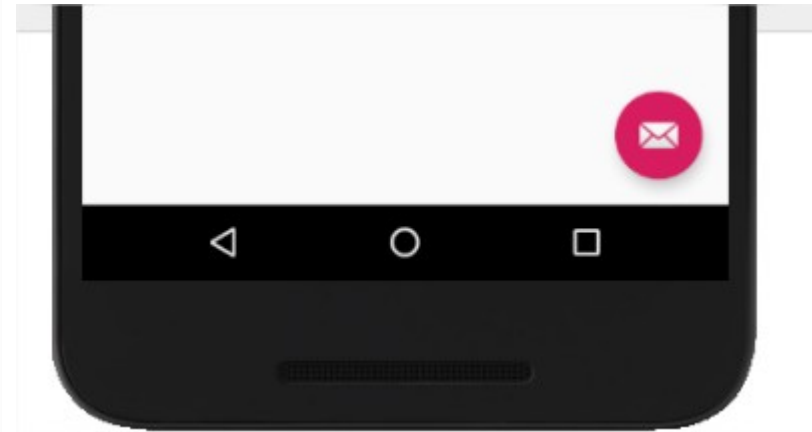
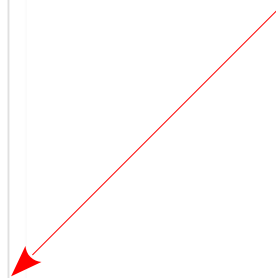
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

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

        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction(text: "Action", listener: null).show();
            }
        });
    }
}
```

On garde cette partie pour l'instant



## En reprenant ce que nous avons vu plus haut :

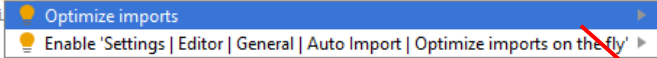
Ici on déclare que notre programme se situe dans le package `com.example.td1helloworld`.  
Si on veut faire référence à notre application, il faudra faire référence à ce package.

Ici on importe des classes qui se trouvent dans des packages différents : les classes.  
Certaines de ces classes sont maintenant devenues inutiles et se retrouvent grisées car nous avons supprimé des fonctions dans le code, comme le montre l'image suivante.

Pour résoudre ce problème, on va utiliser le raccourci clavier le plus magique d'Android Studio : **ALT + ENTREE**. Ce raccourci permet d'effectuer des corrections d'erreurs rapidement, ainsi que d'autres actions rapides. Ici, on vous propose de : "*Optimize Imports*". Appuyez sur **ENTREE** pour valider votre sélection.

```
1 package com.example.td1helloworld;
2
3 import android.os.Bundle;
4
5 import com.google.android.material.floatingactionbutton.FloatingActionButton;
6 import com.google.android.material.snackbar.Snackbar;
7
8 import androidx.appcompat.app.AppCompatActivity;
9 import androidx.appcompat.widget.Toolbar;
10
11 import android.view.View;
12 import android.view.Menu;
13 import android.view.MenuItem;
14
15 public class MainActivity extends AppCompatActivity {
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_main);
21         Toolbar toolbar = findViewById(R.id.toolbar);
22         setSupportActionBar(toolbar);
23
24         FloatingActionButton fab = findViewById(R.id.fab);
25         fab.setOnClickListener((view) -> {
26             Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
27                 .setAction(text: "Action", listener: null).show();
28         });
29     }
30
31 }
32
33
34
35 }
```

```
1 package com.example.tdlhelloworld;
2
3 import android.os.Bundle;
4
5 import com.google.android.material.floatingactionbutton.FloatingActionButton;
6 import com.google.android.material.snackbar.Snackbar;
7
8 import androidx.appcompat.app.AppCompatActivity;
9 import androidx.appcompat.widget.Toolbar;
10
11 import android.view.View;
12 import android.view.Menu;
13 import android.view.MenuItem;
14
15 public class MainActivity extends AppCompatActivity {
16
```



```
1 package com.example.tdlhelloworld;
2
3 import android.os.Bundle;
4 import android.view.View;
5
6 import com.google.android.material.floatingactionbutton.FloatingActionButton;
7 import com.google.android.material.snackbar.Snackbar;
8
9 import androidx.appcompat.app.AppCompatActivity;
10 import androidx.appcompat.widget.Toolbar;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         Toolbar toolbar = findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = findViewById(R.id.fab);
22         fab.setOnClickListener((view) -> {
23             Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
24                 .setAction(text: "Action", listener: null).show();
25         });
26     }
27 }
28
29
30
31
32
```

Android Studio



# Les concepts de l'Orienté objet?

Un programme orienté objet est uniquement constitué de classes interagissant par envoi de messages

L'intégralité du code d'un programme orienté objet se trouve donc à l'intérieur de classes

# La création d'objets

## Écriture d'une classe

```
MainActivity.java x content_main.xml x
package com.example.monappli;
2
3 import android.os.Bundle;
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
```

Un constructeur est une méthode qui doit respecter quelques contraintes :

elle doit porter le même nom que la classe ;

Ici notre Constructeur dérive de AppCompatActivity, ce qui signifie qu'il s'approprie les caractéristiques/propriétés de ce composant.

```
public class MainActivity extends AppCompatActivity {
```

Documentation: AppCompatActivity x

AppCompatActivity (androidx.appcompat.app)

```
androidx.appcompat.app
public class AppCompatActivity
extends androidx.fragment.app.FragmentActivity
implements androidx.appcompat.app.AppCompatActivity, androidx.core.app.TaskStackBuilder.SupportParentable, androidx.a
```

Base class for activities that use the [support library](#) action bar features.

You can add an [ActionBar](#) to your activity when running on API level 7 or higher by extending this class for your activity and setting the activity theme to [Theme.AppCompat](#) or a similar theme.

### Developer Guides

For information about how to use the action bar, including how to add action items, navigation modes and more, read the [Action Bar API guide](#).

Gradle: androidx.appcompat:appcompat:1.1.0@aar



Nous allons modifier notre constructeur.  
On déclare ici une nouvelle classe, MainActivity,  
et on la fait dériver de Activity, puisqu'il s'agit d'une  
activité.

Remplacer AppCompatActivity par simplement

Activity

```
package com.example.tdlhelloworld;

import android.os.Bundle;
import android.view.View;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public class MainActivity extends AppCompatActivity {

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

        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener((view) -> {
            Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction(text: "Action", listener: null).show();
        });
    }
}
```

Android Stu



```
package com.example.tdlhelloworld;
```

```
import android.os.Bundle;  
import android.view.View;
```

```
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import com.google.android.material.snackbar.Snackbar;
```

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

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        Toolbar toolbar = findViewById(R.id.toolbar);
```

```
        setSupportActionBar(toolbar);
```

```
        FloatingActionButton fab = findViewById(R.id.fab);
```

```
        fab.setOnClickListener((view) -> {
```

```
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
```

```
                .setAction("Action", listener: null).show();
```

```
        });
```

```
package com.example.tdlhelloworld;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import com.google.android.material.floatingactionbutton.FloatingActionButton;
```

```
import com.google.android.material.snackbar.Snackbar;
```

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

```
import androidx.appcompat.widget.Toolbar;
```

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        Toolbar toolbar = findViewById(R.id.toolbar);
```

```
        setSupportActionBar(toolbar);
```

```
        FloatingActionButton fab = findViewById(R.id.fab);
```

```
        fab.setOnClickListener((view) -> {
```

```
            Snackbar.make
```

```
                .setA
```

```
        });
```

- Activity (android.app)
- ActivityChooserView (android.app)
- ActivityInfo (android.content)
- ActivityManager (android.app)
- ActivityMonitor (android.app)
- ActivityNotFoundException (android.app)
- ActivityOptions (android.app)
- ActivityResult (android.app)
- ActivityCompat (androidx.appcompat)
- ActivityManagerCompat (androidx.appcompat)
- ActivityOptionsCompat (androidx.appcompat)
- ActivityResultRegistry (androidx.appcompat)



```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public class MainActivity extends Activity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener((view) -> {
            Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction(text: "Action", listener: null).show();
        });
    }
}
```

On ne dérive pas sur la même classe dans les 2 cas. Certaines méthodes ne sont pas définies dans Activity. On retrouve en rouge l'erreur de code.

```
1 package com.example.tdlhelloworld;
2
3 import android.os.Bundle;
4 import android.view.View;
5
6 import com.google.android.material.floatingactionbutton.FloatingActionButton;
7 import com.google.android.material.snackbar.Snackbar;
8
9 import androidx.appcompat.app.AppCompatActivity;
10 import androidx.appcompat.widget.Toolbar;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         Toolbar toolbar = findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = findViewById(R.id.fab);
22         fab.setOnClickListener((view) -> {
23             Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
24                 .setAction(text: "Action", listener: null).show();
25         });
26     }
27 }
28
29
30
31
32 }
```

A

```
package com.example.tdihelloworld;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
public class MainActivity extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        // Toolbar toolbar = findViewById(R.id.toolbar);
```

```
        //setSupportActionBar(toolbar);
```

```
        /* FloatingActionButton fab = findViewById(R.id.fab);
```

```
        fab.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
```

```
            public void onClick(View view) {
```

```
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
```

```
                    .setAction("Action", null).show();
```

```
            }
```

```
        }); /*
```

```
    }
```

```
}
```

**MOST IMPORTANT**

```
package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

public class MainActivity extends Activity {

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

}
```

Le `@Override` permet d'indiquer que l'on va redéfinir une méthode qui existait auparavant dans la classe parente, ce qui est logique puisque nous savons qu'une activité a une méthode `void onCreate()` et que notre classe hérite de `Activity`.

Heu en français s'il vous plait...

Pour faire court : `@Override` c'est le mot clef pour indiquer une méthode redéfinie. Toujours pas ?  
Exemple :

```
1 public abstract class Felin extends Animal {
2
3     @Override
4     void deplacement() {
5         System.out.println("Je me déplace seul !");
6     }
7
8 }
```

```
1 public abstract class Felin extends Animal {  
2  
3     @Override  
4     void deplacement() {  
5         System.out.println("Je me déplace seul !");  
6     }  
7  
8 }
```

La classe Felin étend la classe Animal. (ou ... dérive de ...)

D'après ce bout de code, la classe Animal définit déjà une méthode deplacement() mais la classe Felin veut aussi la définir mais différemment.

C'est pourquoi on met le mot clef @Override avant la définition de la méthode dans la classe Felin qui indique qu'on va redéfinir la méthode qui normalement aurait été héritée de la classe Animal.

Quand on utilisera une instance de Felin genre :

```
Felin chat = new Felin();
```

Si on fait :

```
chat().deplacement();
```

C'est la méthode deplacement() de Felin qui sera appelée et non celle de Animal on dit que la méthode redéfinie, masque la version précédente de la méthode.

L'intérêt de ça, c'est que tous les animaux se déplacent (donc héritent d'une méthode déplacement) mais certains se déplacent différemment des autres animaux donc il faut redéfinir la méthode deplacement().

L'instruction @Override est facultative. Cependant, elle permet au compilateur d'optimiser le bytecode, alors faites y attention.

le mot-clé `@override` est également utilisé pour une méthode qui implémente une interface.

Le compilateur affichera également un avertissement si le prototype de la méthode originale n'est pas respecté.

## Continuons :

```
package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

public class MainActivity extends Activity {

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

}
```

Qui a accès à la méthode - `protected`. Le mot-clé `protected` signifie que la méthode est visible dans les classes héritées

On peut trouver `private` : visible seulement dans la même classe

On peut trouver `public` : visible dans toutes les classes

La méthode retourne-t-elle des données ? Le mot-clé `void` signifie que la méthode `onCreate()` ne retourne aucune donnée au programme appelant, qui en l'occurrence le terminal Android. Mais si on prenait l'exemple d'une méthode effectuant des calculs, celle-ci pourrait retourner un résultat à son appelant.

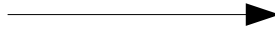
## OnCreate, c'est le nom de la méthode.

Cette méthode est appelée dès la construction de l'activité, elle sert à initialiser ce contrôleur. Cette méthode n'est appelée qu'une seule fois. C'est ici qu'il faudra définir la vue qui sera utilisée, en utilisant `setContentView(int layoutId)` (on verra cela dans un moment. Noter qu'elle reçoit en paramètre un `Bundle savedInstanceState` qui contiendra l'état sauvegardé de la dernière exécution de l'activité (par exemple lorsque nous effectuons une rotation de l'écran).



# Revenons sur Android Studio

???



```
public class Activity extends ApplicationContext {  
    protected void onCreate(Bundle savedInstanceState);  
  
    protected void onStart();  
  
    protected void onRestart();  
  
    protected void onResume();  
  
    protected void onPause();  
  
    protected void onStop();  
  
    protected void onDestroy();  
}
```

Revenons à notre ligne de code :

```
package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

}
```

Cette méthode est donc la première qui est lancée au démarrage d'une application, mais elle est aussi appelée après qu'une application a été tuée par le système en manque de mémoire ! C'est à cela que sert le paramètre de type Bundle :

S'il s'agit du premier lancement de l'application ou d'un démarrage alors qu'elle avait été quittée normalement, il vaut null.

Mais s'il s'agit d'un retour à l'application après qu'elle a perdu le focus et redémarré, alors il se peut qu'il ne soit pas null si vous avez fait en sorte de sauvegarder des données dedans, mais nous verrons cela plus tard.

Dans cette méthode, vous devez définir ce qui doit être créé à chaque démarrage, en particulier l'interface graphique.

```
package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

public class MainActivity extends Activity {

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

L'instruction `super` signifie qu'on fait appel au constructeur parent, à une méthode ou un attribut qui appartient à la superclasse de la méthode actuelle. Autrement dit la classe juste au-dessus dans la hiérarchie de l'héritage — la classe parente, c'est-à-dire la classe `Activity`. Ainsi, `super.onCreate` fait appel au `onCreate` de la classe `Activity`, mais pas au `onCreate` de `MainActivity`. Il gère bien entendu le cas où le `Bundle` est `null`.

Cette instruction est obligatoire.  
Android Studio - Nicolas Guizard

```
package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

public class MainActivity extends Activity {

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

}
```

*Nous verrons plus en détails ultérieurement cette instruction.*

Mais nous avons vu que la méthode `onCreate` est appelée à la création de l'activity, elle va nous permettre de relier l'activity avec sa vue.

Comment ? Avec justement la méthode `setContentView`. On voit que la méthode prend le paramètre :

`R.layout.activity_main`

Pour schématiser, cela veut dire que l'activity va afficher la vue contenue dans le fichier `activity_main.xml` du dossier `res/layout`.

Continuons,

Pour l'instant, vous pouvez remplacer le contenu du fichier par celui-ci :

On a commenté :

```
setContentView(R.layout.activity_main);
```

```
package com.example.tdlhelloworld;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
public class MainActivity extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        //setContentView(R.layout.activity_main);
```

Lancer votre AVD pour voir ce qui se passe à l'écran...

## Rajoutons du code Java...

```
package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView
```

- TextView (android.widget)
- TextureView (android.view)
- TextViewCompat (android.support.v4.widget)

android.widget  
public class **TextView**  
extends android.view.  
implements android.vi

A user interface element that  
The following code sample  
view:

```
<LinearLayout
    xmlns:android=
    android:layout
    android:layout
    <TextView
        android:id=
        android:lay
        android:lay
        android:tex
    </LinearLayout>
```

```

package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView text = new TextView(context: this);

        text.setText("Enfin un Hello WORLD");

        se

```

```

m ▶ setContentView(View view) void
m ▶ setContentView(int layoutResID) void
m ▶ setContentView(View view, LayoutParams params) void
m ▶ sendBroadcast(Intent intent) void
m ▶ sendBroadcast(Intent intent, String receiverPermission) void
m ▶ sendBroadcastAsUser(Intent intent, UserHandle user) void
m ▶ sendBroadcastAsUser(Intent intent, UserHandle user, Strin.. void
m ▶ sendOrderedBroadcast(Intent intent, String receiverPermis.. void
m ▶ sendOrderedBroadcast(Intent intent, String receiverPermis.. void
m ▶ sendOrderedBroadcastAsUser(Intent intent, UserHandle user.. void
m ▶ setActionBar(ToolBar toolbar) void

```

android.app.Activity

public void setContentView(android.view.View

Set the activity content to an explicit view. This view is placed directly into the activity's view hierarchy. It can itself be a view hierarchy. When calling this method, the layout parameters of the specified view are ignored. Both the width and the height of the view are set by default to `ViewGroup.LayoutParams.MATCH_PARENT`. To use your own layout parameters, invoke `setContentView(View, ViewGroup.LayoutParams)`.

Params: view – The desired content to display.



```
package com.example.monappli;

import android.app.Activity;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView text = new TextView( context: this);

        text.setText("Enfin un Hello WORLD");


        setContentView(text);
    }
}
```

**Select Deployment Target** [X]

No USB devices or running emulators detected [Troubleshoot](#)

Connected Devices —  
< none >

Available Virtual Devices —

-  Nexus 5X API 24

[Don't see your device?](#)

Use same selection for future launches

