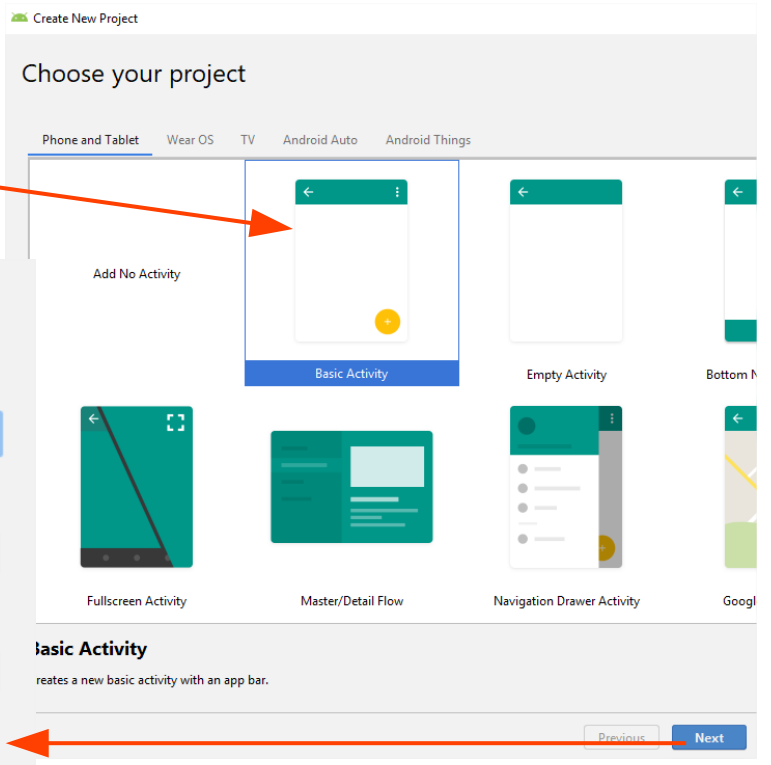
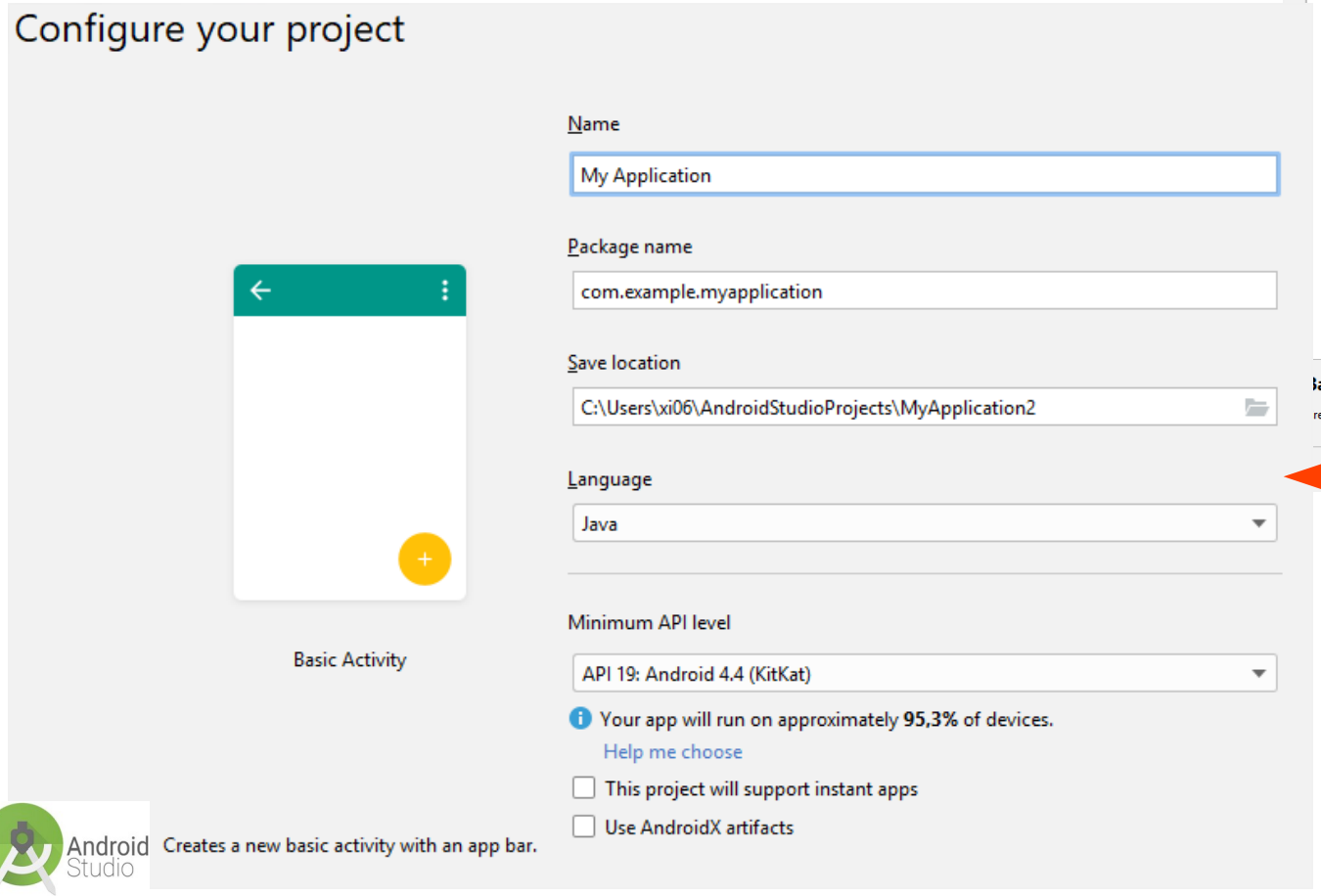
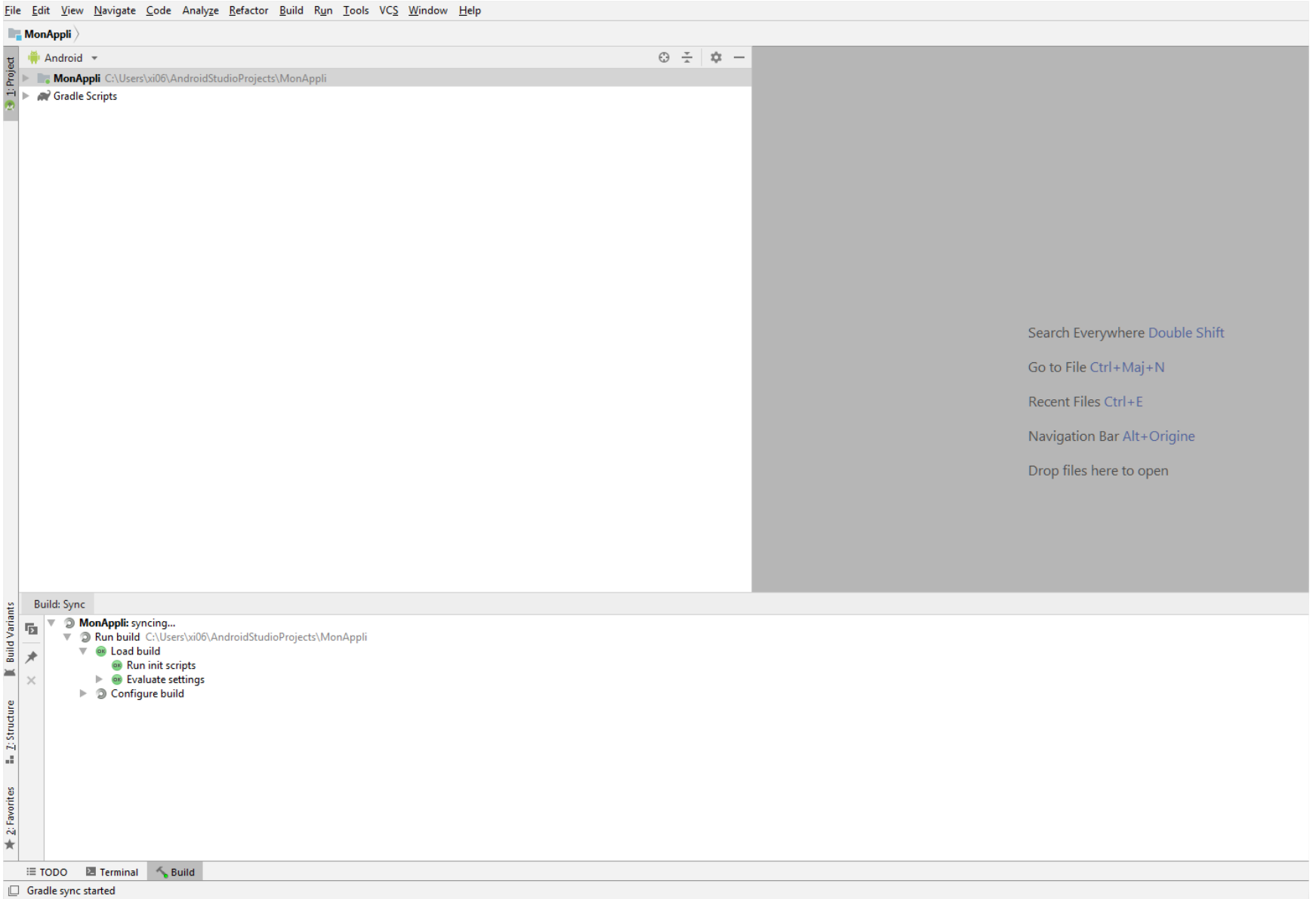
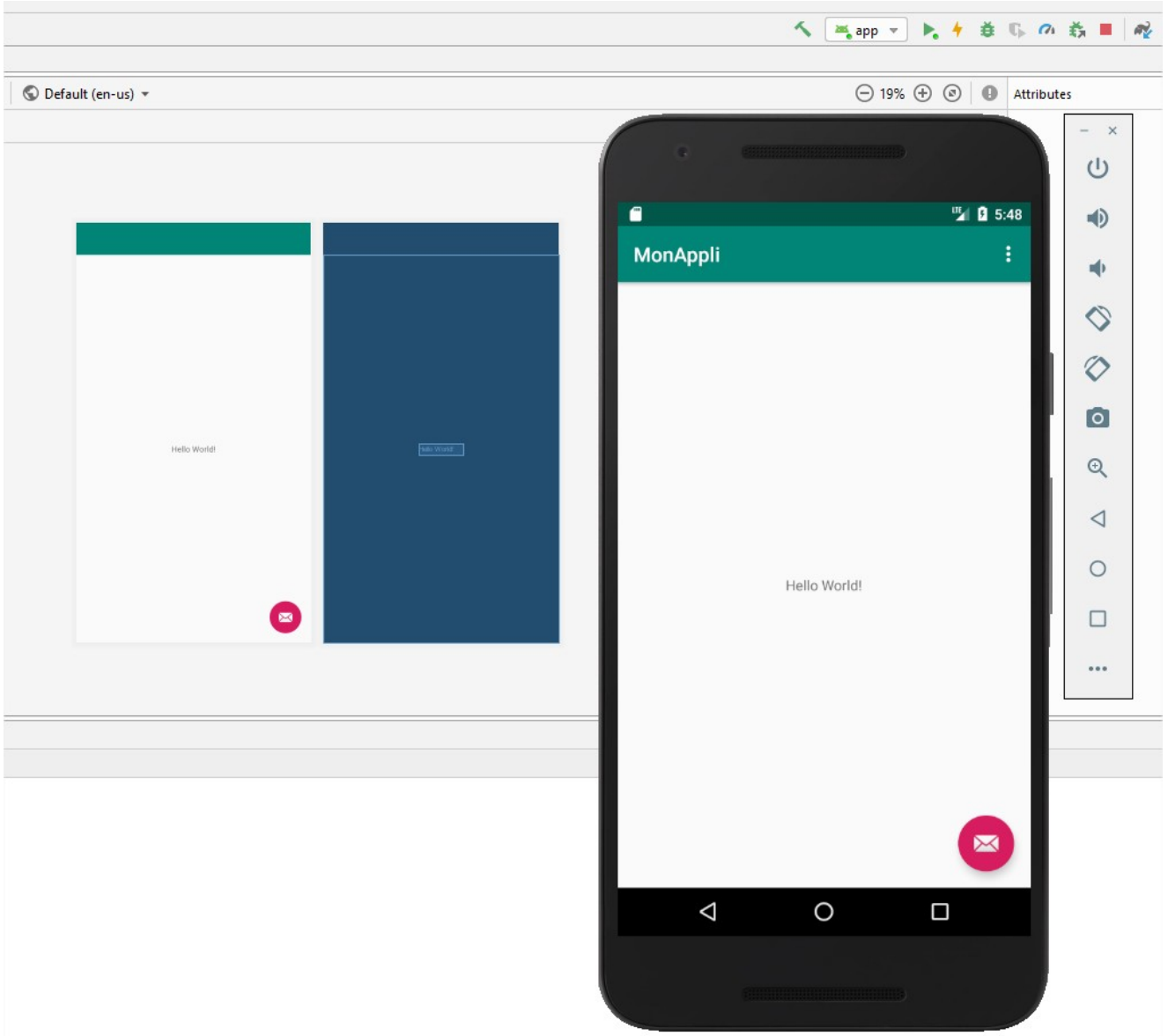
TD1:Première Appli Hello WORLD

Créez votre projet

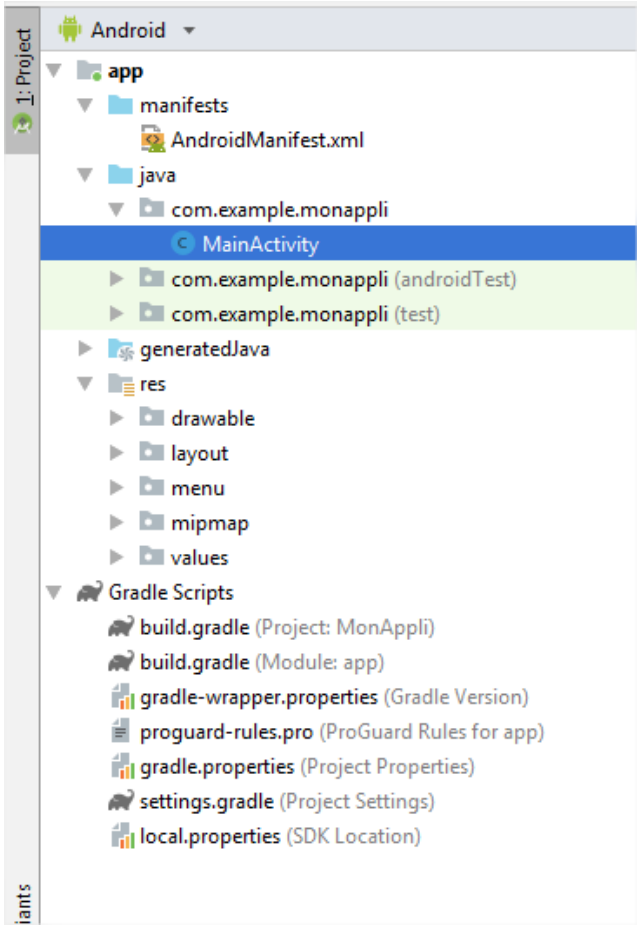


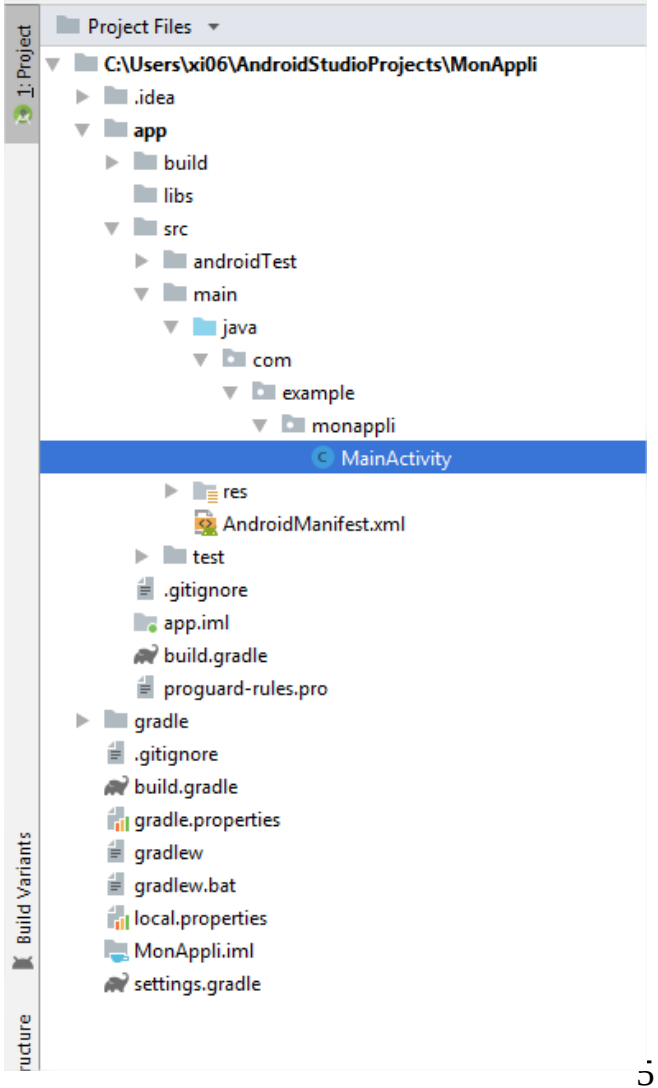



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MonAppli
Android
MonAppli C:\Users\xi06\AndroidStudioProjects\MonAppli
Gradle Scripts
Search Everywhere Double Shift
Go to File Ctrl+Maj+N
Recent Files Ctrl+E
Navigation Bar Alt+Origine
Drop files here to open
Build: Sync
MonAppli: syncing...
Run build C:\Users\xi06\AndroidStudioProjects\MonAppli
Load build
Run init scripts
Evaluate settings
Configure build
1: Project
Build Variants
7: Structure
2: Favorites
TODO
Terminal
Build
Gradle sync started



Android Studio

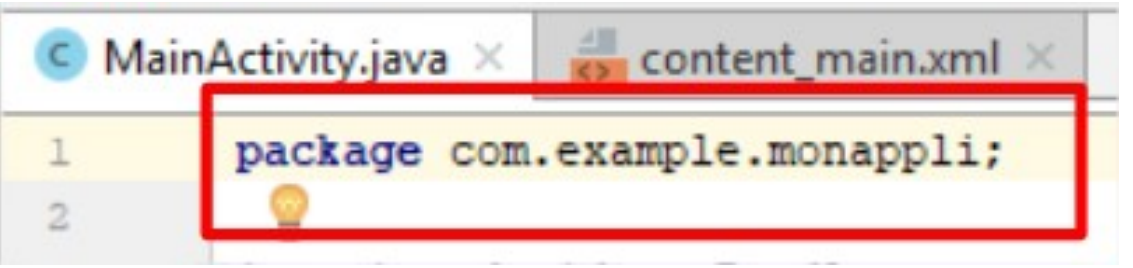

1: Project
Android
app
manifests
AndroidManifest.xml
java
com.example.monappli
MainActivity
com.example.monappli (androidTest)
com.example.monappli (test)
generatedJava
res
drawable
layout
menu
mipmap
values
Gradle Scripts
build.gradle (Project: MonAppli)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)


1: Project
Project Files
C:\Users\xi06\AndroidStudioProjects\MonAppli
.idea
app
build
libs
src
androidTest
main
java
com
example
monappli
MainActivity
res
AndroidManifest.xml
test
.gitignore
app.iml
build.gradle
proguard-rules.pro
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
MonAppli.iml
settings.gradle
Build Variants

MainActivity.java | content_main.xml

```java
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();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
```

Les packages



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;

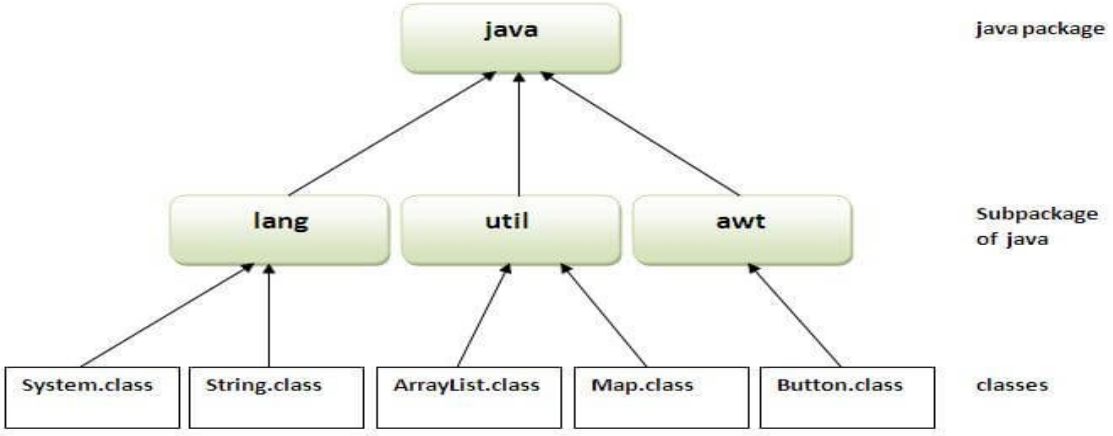
Attention package devra toujours être la première ligne !

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

Figure 20. Quelques packages de la librairie standard Java.

MainActivity.java | content_main.xml

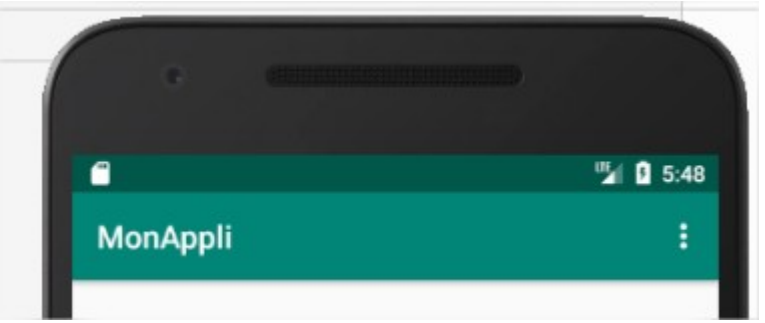
```java
package com.example.monappli;

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();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
```

MainActivity.java × content_main.xml ×

```java
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();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

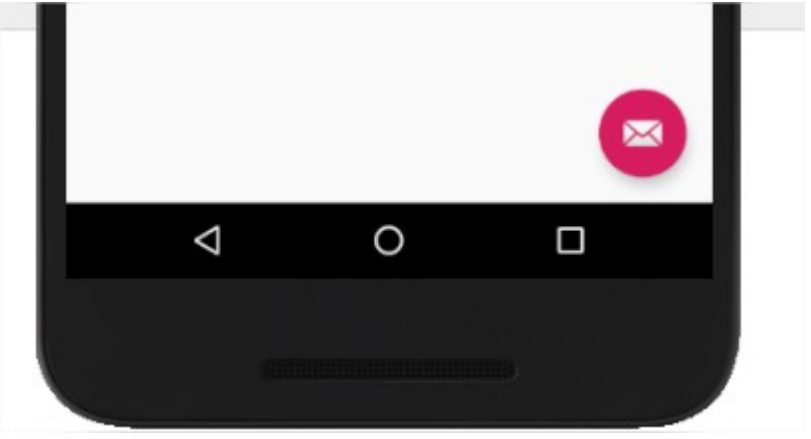
Android studio a généré automatiquement des objets lors de la création de notre projet.
Il y a beaucoup de choses et certaines seront expliquées plus tard.
Pour l'instant nous allons faire un peu de ménage dans tout cela. Vous allez modifier le code.
Supprimez les méthodes :

onCreateOptionsMenu et onOptionsItemSelected afin d'obtenir :



```java
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 devenus 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.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
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) → {
28                 Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
29                         .setAction( text: "Action",  listener: null).show();
30         });
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.vi
14
15 public class MainActivity extends AppCompatActivity {
16
```

Optimize imports

Enable 'Settings | Editor | General | Auto Import | Optimize imports on the fly'

```
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) → {
25                 Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
26                         .setAction( text: "Action",  listener: null).show();
27         });
29     }
30
31
32 }
```

Android Studio

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 | content_main.xml

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

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 AppCompatAc {
```

AppCompatActivity (androidx.appcompat.app)

Documentation: AppCompatActivity

```
androidx.appcompat.app
public class AppCompatActivity
extends androidx.fragment.app.FragmentActivity
implements androidx.appcompat.app.AppCompatCallback, 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();
        });
```

```java
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 {

    @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();
        });
```

```java
package com.example.tdlhelloworld;

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

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

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

public class MainActivity extends Activ {

    @Override
    protected void onCreate(B
        super.onCreate(savedI
        setContentView(R.layo
        Toolbar toolbar = fin
        setSupportActionBar(t

        FloatingActionButton
        fab.setOnClickListene
                Snackbar.make
                        .setA
        });
```

```
Activity (android.app)
ActivityChooserView (andro
ActivityInfo (android.cont
ActivityManager (android.a
ActivityMonitor (android.a
ActivityNotFoundException
ActivityOptions (android.a
ActivityResult (android.ap
ActivityCompat (androidx.
ActivityManagerCompat (an
ActivityOptionsCompat (an
```

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.

```java
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 ac
                        .setAction( text: "Action", listener: null).sho
        });
```

```java
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();
        });
    }


}
```

```
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);

        // 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();
            }
        });      */
    }


}
```

Android Studio

```java
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 :

```java
public abstract class Felin extends Animal {

        @Override
        void deplacement() {
                System.out.println("Je me déplace seul !");
        }

}
```

```java
public abstract class Felin extends Animal {

	@Override
	void deplacement() {
		System.out.println("Je me déplace seul !");
	}

}
```

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

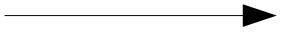
La méthode retourne-t-elle des données ? Le mot-clé void signifie que la méthode onCreate() ne retourne aucune donnée au programme appelant, qui en l'occurrence le terminal Android. Mais si on prenait l'exemple d'une méthode effectuant des calculs, celle-ci pourrait retourner un résultat à son appelant.

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 Activity {

    @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.

```java
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.

```java
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 ficher activity_main.xml du dossier res/layout.