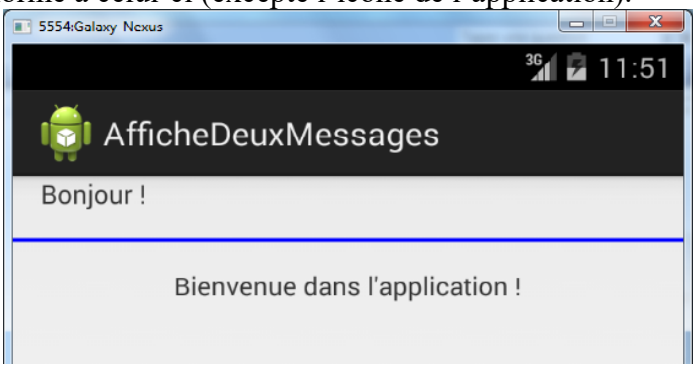
Utilisation de TextView, ImageView, LinearLayout

Exercice 1 - Une application qui affiche 2 messages

Le but de ce premier exercice est de développer une application qui affiche deux messages séparés par une barre horizontale.

Le rendu devra être conforme à celui-ci (excepté l'icône de l'application).



On constate que 3 *widgets* sont présents sur l'interface (2 textes affichés + une barre horizontale) et qu'ils sont placés les uns en dessous des autres : un ***LinearLayout*** sera bien adapté pour obtenir cette présentation. Vous serez donc amené à remplacer le *ConstraintLayout* utilisé par défaut dans le fichier xml décrivant la vue par un ***LinearLayout***. Vous penserez à ajouter une propriété spécifiant l'orientation des *widgets* contenus dans le ***LinearLayout***.

Pour afficher la barre horizontale, on utilisera le code XML suivant :

```
<View
    android:layout_width="match_parent"
    android:layout_height="2dip"
    android:layout_marginTop="15dip"
    android:layout_marginBottom="15dip"
    android:background="#0000FF" />
```

1) Questions en lien avec ce code :

a. Quelles sont les propriétés qui permettent de définir les marges à laisser autour d'un *widget* ?
b. Que signifie l'abréviation `dip`, utilisée comme unité de mesure ? Rechercher la réponse sur le site Internet dédié à Android, si nécessaire. Quel est l'avantage d'utiliser cette unité plutôt que `mm` (millimètre) ou `px` (pixel) ?

2) Développer l'application Android permettant d'effectuer l'affichage demandé. Tester l'application sur l'émulateur.
Indications - Voir l'exemple du cours pages 20 à 23 et remarquer les 2 propriétés suivantes :

✓ pour centrer le *widget* ***TextView***, il faut utiliser la propriété :

```
android:gravity="center"
```

✓ pour laisser une marge à gauche du *widget* ***TextView***, il faut utiliser la propriété :

```
android:layout_marginLeft="15dip"
```

Exercice 2 - Application pour générer un nombre aléatoire

Le but de l'exercice est de développer une application qui affiche un nombre aléatoire compris entre 0 et 99. Une classe outil est disponible sur le serveur : `OutilAleatoire.java`. L'affichage devra être approximativement :

La différence avec l'exercice précédent est qu'une partie du texte à afficher n'est pas statique. En effet, le nombre aléatoire doit être créé dynamiquement au lancement de l'application.

Dans la classe principale (celle qui hérite de ***Activity***), il sera nécessaire d'accéder à l'identifiant du deuxième ***widget*** qui affiche du texte, en appelant la méthode `findViewById` :

```
// on récupère un accès au widget TextView destiné à afficher le résultat
TextView resultat = findViewById(R.id.etiquette_resultat);
```

puis de modifier son contenu, en appliquant la méthode ***setText***.

Pour récupérer une chaîne de caractères définie en tant que ressource, on écrit fait appel à la méthode de classe Activity, ***getString*** :

```
getString(R.string.message_resultat)
```

Présentation

- ✓ Le premier texte (message de bienvenue) est centré
- ✓ Une marge de 15 dip a été laissée au dessus de ce texte
- ✓ A gauche du texte qui affiche le nombre aléatoire, il y a une marge de 30 dip

Question

Coder l'application et la tester sur l'émulateur.

Exercice 3 - Application pour générer un nombre aléatoire - Amélioration

Avant de traiter l'exercice, lire le support de cours pages 34 et 46.

Le but de l'exercice est d'améliorer l'aspect visuel de l'application codée dans l'exercice 2. Pour ce faire, on codera proprement en définissant des couleurs et des dimensions dans les fichiers adéquats.

1) Modifier le fichier ***colors.xml*** pour lui ajouter la définition d'une ou deux couleurs de votre choix. (voir le cours page 34).

2) Ajouter en tant que ressources un fichier contenant la définition de toutes les dimensions utilisées dans l'interface, donc utilisées dans le fichier ***layout*** (voir le cours page 34). Prévoir une dimension pour afficher en plus grand l'un des textes de l'application.

3) Modifier en conséquence le fichier ***layout*** décrivant la vue de l'activité. Le but est d'utiliser les constantes définies dans les 2 fichiers précédents. De plus, on souhaite que l'un des textes s'affiche en gras et/ou en italique.

   - voir page 34 pour savoir comment faire référence à une couleur ou dimension définie en tant que constante

   - voir page 46, les noms des propriétés d'un ***TextView*** permettant notamment de définir une taille de caractères ou un style de caractères