

ملخص مسار الاندرويد

Android Track

ملخص مسار الاندرويد من ضمن مبادرة مليون مبرمج عربي

One Million Arab Coders

</10⁶>

مبادرة مليون مبرمج عربي
1 MILLION ARAB CODERS INITIATIVE

المحتويات

3	الدرس الاول
4	الدرس الثاني
10	الدرس الثالث
18	الدرس الرابع
21	الدرس الخامس
24	الدرس السادس
30	الدرس السابع
34	الدرس الثامن
49	الدرس التاسع
66	الدرس العاشر
69	الدرس الحادي عشر
73	الدرس الثاني عشر
	الدرس الثالث عشر
	الدرس الرابع عشر
87	الخاتمة

الدرس الاول : التفكير بطريقة المبرمجين

يتضمن عدة طرق للتفكير وهي:

التفكير الاجرائي : وهو تقسيم المشكلة الى عدة خطوات للوصول الى الحل.

التفكير النظري : وهو التفكير المجرد او ما يعرف ب Abstract thinking

الطريقة المنهجية لحل الالخطاء systematic debugging حيث تحل الالخطاء بطريقة منهجية وتتبع الاسباب

التقمص التكنولوجي Technological Empathy ويتضمن كيفية رؤية الكمبيوتر للامور

التفكير المنظومي Systems Thinking وهو تحليل المشكلات الكبيرة الى اجزاء صغيرة وعادة ماتكون على الورق وتنطوي بالتفكير في الصورة الشاملة للاشياء

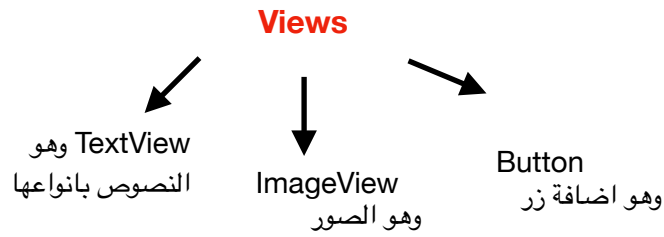
الدرس الثاني (انشاء التخطيطات الجزء الاول)

في البداية يجب عليك التعرف على اسماء وصور المدرسين في هذه الدورة

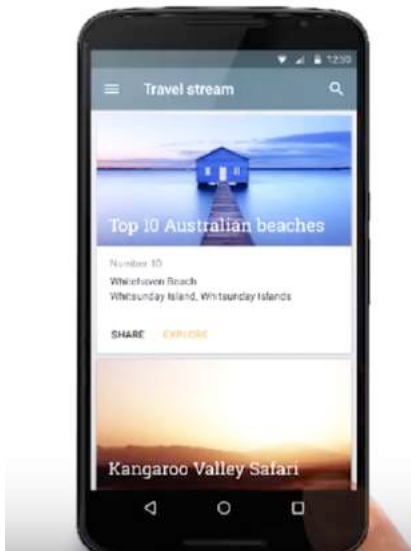


Lyla Fujiwara و Kunal Chawla و Katherine Kuan

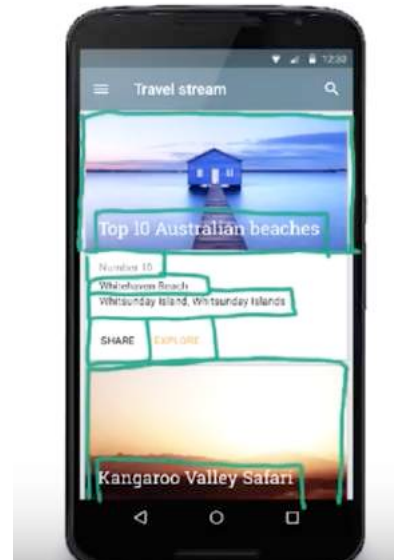
في الخطوة الاولى يجب عليك التعرف على ال Views العرض في الاندرويد مثل النص او الصورة او الزر وحيث تسمى هذ الواجهة UI Interface



ان كل عرض هو عبارة عن مستطيل في الشاشة كما هو موضح في الصورة ادناه:



كما تلاحظون في الصورتين كيف تم توضيح ان العرض مقسم في واجهة المستخدم على شكل مستطيلات كما هو موضح في الصورة على اليمين من صور ونصوص..الخ



ويجب الإشارة ان اسماء هذه العروض **views** يجب ان تستخدم فيها طريقة **Camel Case** وهذه تعني سنام الجمل والمقصود بها انه لايجب ان يكون هناك اي فراغات وان كل بداية كلمة يجب ان تكون بالاحرف الكبيرة مثل **TextView** كما لاحظنا ان عرض النص يتكون من كلمتين **Text** و **View** حيث لا يوجد فراغ بينهم وايضا كل كلمة تبدأ بحرف كبير **Capital Letter**

XML

Extensible Markup Language

ان **XML** هي طريقة لوصف كيف سيبدو تطبيق الاندرويد لكي نستخدمها في بناء المخططات **Layouts** (ببساطة هي وصف اين سنضع الاشياء وماالذي نحتاجه لوضع هذه الاشياء).

ان كتابة الاكواد هي اشبه بكتابة التعليمات (**تعليمات محددة للغاية**) والمنصة اللتي سنحتاجها لكتابة هذه الاكواد هي اندرويد ستوديو **Android Studio**

الان لنعاين اول كود برمجي لنا بلغة **XML** وسيكون لعرض النص **TextView**

```
<TextView
android:text="Happy Birthday!"
android:background=@android:color/dark_gray"
android:layout_width="150dp"
android:layout_height="75dp" />
```

وهذا ماسيظهر على شاشة الهاتف عند كتابة هذا الكود :



XML Syntax

ونعني بها القواعد التي تحدد التركيبة السليمة لل XML حيث اذا تمت كتابة قاعدة XML غير سليمة او بالاحرى غير صحيحة فلن يظهر على الهاتف.

ملاحظات حول XML Syntax لا تنسى ابدا اغلاق الكود </> بهذه العلامة او عند كتابة الكود يجب ان يكون افتتاح الكود ب
مثال:

```
<TextView
    android:text="Happy Birthday!"
    android:background="@android:color/darker_gray"
    android:layout_width="150dp"
    android:layout_height="75dp" />
```

كما لاحظتهم في المثال السابق انه قمنا بالاغلاق باستخدام />

ويطلق على العناصر داخل هذا الكود بالميزات أو Attributes

`<TextView`
`android:text="Happy Birthday!"`
`android:background="@android:color/darker_gray"`
`android:layout_width="150dp"`
`android:layout_height="75dp" />`

Annotations in the image:
 - Red arrow pointing to `android:text="Happy Birthday!"`: مميزات
 - Red arrow pointing to `android:background="@android:color/darker_gray"`: مميزات
 - Red arrow pointing to `android:layout_width="150dp"`: مميزات
 - Red arrow pointing to `android:layout_height="75dp"`: مميزات

وظيفة المميزات attributes هي خصائص تحدد تصرف او مظهر العرض على هاتف اندرويد.

كمثال بسيط انه الشخص مثلا لديه مميزات شعر اسود وبشرة سمراء ففي الكود لدينا مميزات مثل:

Text, Background, Width, Height

IDE

Integrated development environment

بيئة تطوير متكاملة

XML V

XML وهو بيئة عمل تطبيق كودات XML Visualizer

```
<TextView
    android:text="Happy Birthday, Kunal!"
    android:background="@android:color/darker_gray"
    android:layout_width="300dp"
    android:layout_height="500dp" />
```

كما تلاحظون في الكود اعلاه وجود حرفي **dp** وهو **Density Independent Pixels** وهو وحدة قياس المعايينات في العرض في الاندرويد والسبب من استخدامنا لل dp هو لتوحيد قياس عرض العرض والارتفاع على الاجهزة مختلفة الدقة في الشاشة.

كيفية تشخيص الاخطاء؟

- **اولا** قم بقراءة الخطا في الكود (النص البرمجي)
- **ثانيا** قم بالبحث عن الخطا عن طريق كتابته في محرك البحث كوكل
- **ثالثا** يمكنك ايضا نسخ الخطا ووضعها في محرك البحث والبحث عن اجابة
- **رابعا** هناك طريقة اخرى يمكنك ايضا استخدامها وهي تعرف ب **Android Cheat sheet** ويمكنك تنزيلها من هذا الرابط اذا لم تقم بتنزيلها سابقا.

<http://labs.udacity.com/images/Common-Android-Views-Cheat-Sheet.pdf>

- الفائدة من الطريقة الرابعة وهي لمقارنة الكود المكتوب وملاحظة اذا كانت هناك اي اخطاء
- **خامسا** وهي التراجع او **undo** عند الكتابة فمن الممكن تم كتابة حرف واحد قام بتغيير تركيب الكود باكماله او كتابة كود خاطئ فالفائدة من هذه الطريقة هو التراجع والتأكد من الكود وبعدها نقوم بكتابته مرة ثانية.
 - **سادسا** تصوير الشاشة ومشاركة الكود او رسالة خطأ الكود ولكن يفضل الكود مع رسالة الخطأ في المنتديات للحصول على الحل.

الفرق بين dp و wrap_content : هو انه عند تغيير النص او زيادته سيتم زيادة مساحة العرض مع زيادة او نقصان النص فاذا قمنا مثلا بتحديد 15dp سيتم عرض النص فقط على مساحة 15dp والباقي لن يتم عرضه اما عند اختيار wrap_content فسيتم عرض النص مهما كان طويلا وسيتم العرض بصورة كاملة.

ملاحظة يمكنك استخدام wrap_content مع العرض والطول width ، height حسب الرغبة او العرض.

SP

Scale Independent Pixels

وهي وحدة القياس المستخدمة في تكبير او تصغير حجم النص كما هو موجود في المثال التالي:

```
<TextView
    android:text="Excited for the gift you'll surprise me with."
    android:background="@android:color/darker_gray"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="45sp" />
```

ويمكنك الدخول الى موقع Material Design لتحديد الاحجام المناسبة للنصوص عند الكتابة وهذا شئ مهم عن تصميم التطبيق لكي يكون اكثر تناسقا.

<http://www.google.com/design/spec/style/typography.html#typography-styles>

ولتغيير اللون يمكنك ايضا الدخول موقع الماتيريال ديزاين للالوان لاضافة الالوان الي تحب لتغيير خلفية النص

<http://www.google.com/design/spec/style/color.html#color-color-palette>

ملاحظة عند النسخ الكود يجب مسح @android:color/ واستبدالها برمز hex للون كما في المثال التالي

```
<TextView
    android:text="I got you a free hug. Surprise!"
    android:background="#2196F3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="45sp" />
```

ونفس الشئ من الممكن عمله عند اضافة تغيير لون النص كما في المثال التالي:

```
<TextView
    android:text="I got you a free hug. Surprise!"
    android:background="#2196F3"
    android:textColor="#AED581"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="45sp" />
```

ImageView

عرض الصور

```
<ImageView
    android:src="@drawable/cake"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:scaleType="center"/>
```

كما تلاحظون في الكود اعلاه تم اضافة كود جديد لاضافة الصور وفي خلال هذه الدورة تم تحميل هذه الصور مسبقا في متصفح XML Visualizer حيث يرجى **الملاحظة** انه عند استعمال الاندرويد ستوديو يجب عليك تحميل الصور يدويا وهذا سيتم التطرق اليه لاحقا.

وتلاحظون ايضا ان تم اضافة كود جديد **scaleType** وهو كود لكيفية عرض الصورة وكما تشاهدون في الكود اعلاه تم تحديد العرض ب **center** ويرجى الانتباه بان **scaleType** لاتقوم بتصغير او تكبير الصورة وانما تقوم فقط بتوسيطها ولكي تختار الخيار الافضل لعرض الصورة يمكنك اختيار **centerCrop** كما هو في الكود التالي ويمكنك معرفة الفرق من خلال المقارنة بين الصورتين عند استخدام **center** و **centerCrop**

```
<ImageView
    android:src="@drawable/cake"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:scaleType="centerCrop"/>
```

ملاحظة مهمة يطلق على الصور اللتي تملأ الجوانب بصورة كاملة بدون نقصان ب **Full Bleed** **Image** وهي الصورة الموجودة على اليمين في الاسفل.

عند استعمال **center**



عند استعمال **centerCrop**



الدرس الثالث (انشاء التخطيطات الجزء الثاني)

ViewGroups

مجموعات العرض

LinearLayout التخطيط النسبي
حيث يمكنها ترتيب الابناء بشكل عمودي او افقي.
ويقصد بالابناء هنا مثلا TextView
حيث يعتبر LinearLayout بأحد الابوين Parent

RelativeLayout التصميم النسبي
وهو تحديد تموضع العروض سواء كانت في الاعلى ام في الاسفل او في الجهة اليمنى او اليسرى وحتى في المنتصف.

Nested Groups المجموعات المتداخلة
سيتم شرحها بشكل مفصل اكثر في الدرس السابع

وهذا ماسيتم عرضه على الهاتف عند كتابة كود LinearLayout

مثال على LinearLayout

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">

  <TextView
    android:text="Guest List"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp" />

  <TextView
    android:text="Kunal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp" />

</LinearLayout>
```



الفرق في العرض والارتفاع بالنسبة للابناء او العروض views
ستلاحظون في الصور ادناه الفرق عند تغيير العرض والارتفاع اذا كان رقم او wrap_content او match_parent

يرجى الملاحظة ان في هذه الصورة تم تغيير العرض

DIFFERENT WIDTHS ON CHILD VIEWS

200 dp



wrap-content



match-parent



يرجى الملاحظة ان في هذه الصورة تم تغيير الطول

DIFFERENT HEIGHTS ON CHILD VIEWS

200 dp



wrap-content



match-parent



كيفية توزيع الابعاد بصورة متساوية بين عرض النصوص على الشاشة

ستلاحظون في الصورة التالية اننا استخدمنا سطر برمجي جديد في عرض النصوص المتمثل ب **layout_weight**

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:text="Tom"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textSize="24sp"
        android:background="#31698a" />

    <TextView
        android:text="Tim"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textSize="24sp"
        android:background="#f6f1e9" />

    <TextView
        android:text="Todd"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textSize="24sp"
        android:background="#ff5900" />

</LinearLayout>
```



وسيظهر على الهاتف بهذه الطريقة:

يرجى **الملاحظة** انه عندما تريد التوزيع بصورة متساوية يرجى مراعاة تغيير الارتفاع height الى 0dp والثقل weight الى 1 وهنا لانحتاج الى اضافة dp

Relative Layout

الغاية من التصميم النسبي Relative Layout هو تحديد تموضع العروض views مثلا النصوص TextView او الصور ImageView او الازار Button.

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:textAppearance="?android:textAppearanceLarge"
        android:text="Happy" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:textAppearance="?android:textAppearanceLarge"
        android:text="Birthday" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:textAppearanceLarge"
        android:text="To You" />

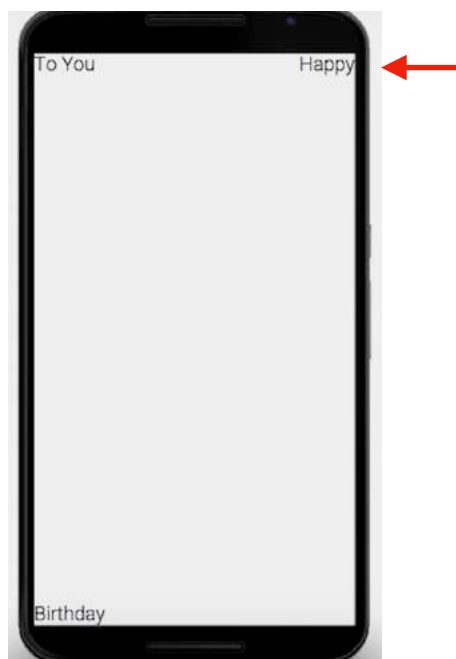
</RelativeLayout>
```

تلاحظون في هذا الكود تم اضافة اسطر برمجية جديدة وهي :

android:layout_alignParentRight="true" وتعني انه تم تحديد النص في الجهة اليمنى من الشاشة

android:layout_alignParentTop="true" وتعني انه تم وضع النص في اعلى الشاشة

وهو موضح في الصورة في الاسفل :



الآن لتحديد عدة نصوص وتوزيعها حسب الرغبة لنفترض اننا نود توزيع اسماء الاسماء على الطاولات ونود ان نرى ذلك على شاشة الهاتف ولكن قبل وضع الكود والصورة النهائية على شاشة الهاتف هناك سطر برمجي جديد وهو المعرف id ان الغرض من انشاء هذا المعرف هو لتحديد مثلا وضع Natalie على نفس طاولة Lyla

كما لاحظتكم انه تم انشاء معرف لنا التالي `android:id="@+id/natalie_text_view"`

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/lyla_text_view" ←
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:textSize="24sp"
        android:text="Lyla" />

    <TextView
        android:id="@+id/me_text_view" ←
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_toRightOf="@id/lyla_text_view"
        android:textSize="24sp"
        android:text="Me" />

    <TextView
        android:id="@+id/natalie_text_view" ←
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"

    <TextView
        android:id="@+id/jennie_text_view" ←
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:textSize="24sp"
        android:text="Jennie" />

    <TextView
        android:id="@+id/omoju_text_view" ←
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_above="@id/jennie_text_view"
        android:textSize="24sp"
        android:text="Omoju" />

    <TextView
        android:id="@+id/amy_text_view" ←
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_above="@id/omoju_text_view"
        android:textSize="24sp"
        android:text="Amy" />
```

```

<TextView
    android:id="@+id/ben_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:textSize="24sp"
    android:text="Ben" />

<TextView
    android:id="@+id/kunal_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_toLeftOf="@id/ben_text_view"
    android:textSize="24sp"
    android:text="Kunal" />

<TextView
    android:id="@+id/kagure_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@id/ben_text_view"
    android:textSize="24sp"
    android:text="Kagure" />

</RelativeLayout>

```

وتم اضافة سطور برمجية جديدة مثل :

android:layout_toLeftOf

android:layout_toRightOf

android:layout_above

حيث ان هذه السطور لتسهيل تموضع العنصر ويجب ان لاننسى استخدام المعرف معها لكي لا يحدث

خطأ في الكود مثال:

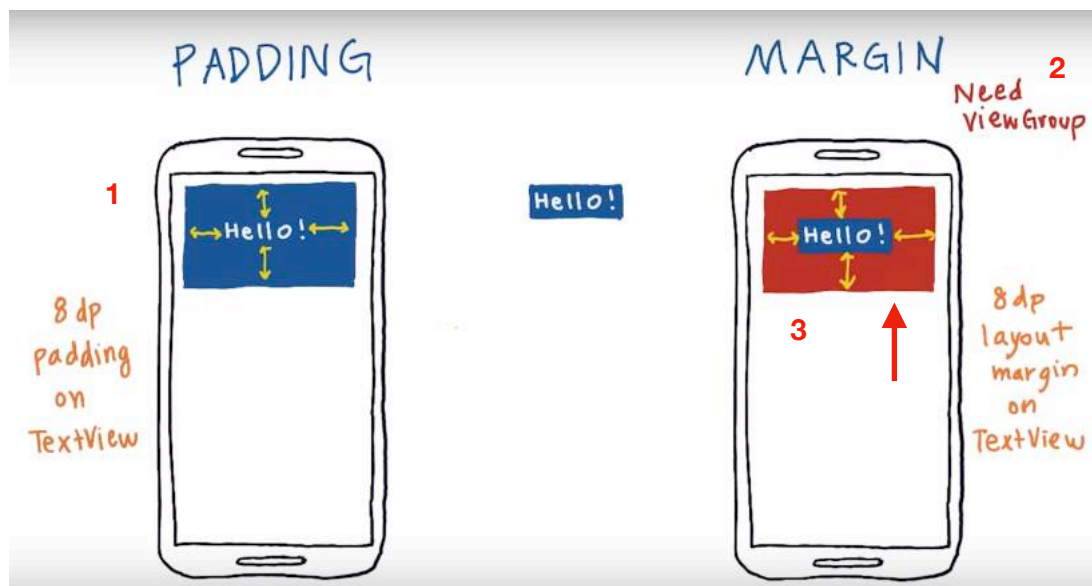
android:layout_toRightOf="@+id/ben_text_view"

وهذا ماسيظهر على شاشة الموبايل:



الهامش Margin & الحشو Padding

وهو لإضافة بعض المساحات او الفراغات كل لاتكون العروض مزدحمة



- من الصورة السابقة سنكون عدة ملاحظات سيتم ترقيمها لمعرفة الفرق بين ال Margin وال Padding :
- ١- الحشو Padding سيكون بأضافة رقم نحدده مثلا 8dp حيث سيقوم بأضافة 8dp لكل جوانب النص TextView يعني من اليمين واليسار وفوق وتحت.
 - ٢- الفرق بين حشو والهامش ان الهامش Margin يحتاج الى مجموعة العرض ViewGroup.
 - ٣- ان الفرق الكبير الذي ستراه ما بين الحشو والهامش هو اذا كانت خلفية النص ملونة بالاضافة الى ان مجموعة العرض ViewGroup ايضا ملونة كما تلاحظون في الصورة السابقة على اليمين. وملاحظة اخيرة اذا لم تكن الخلفية ملونة بالنسبة للنوعين فعلى شاشة الهاتف سيبدوان نفس الشكل.

الكودات المستخدمة في الحشو Padding :

هذا الكود سيقوم بتحديد الحشو لجميع الاطراف بنسبة ٨
او يمكننا تحديد مقدار الحشو بالنسبة لليمين واليسار والاعلى والاسفل

```
android:paddingRight="8dp"
android:paddingLeft="8dp"
android:paddingTop="8dp"
android:paddingBottom="8dp"
```

ولاننسى ان هذه الكودات تكون داخل مميزات عرض النص TextView Attribute

الكودات المستخدمة في الهامش Margin :

android:layout_margin="8dp"

او يمكننا تحديد الهامش Margin بصورة اكثر دقة من حيث تحديد الهامش بالنسبة لليمين واليسار والاعلى والاسفل كما فعلنا مع الحشو

android:layout_marginRight="8dp"

android:layout_marginLeft="8dp"

android:layout_marginTop="8dp"

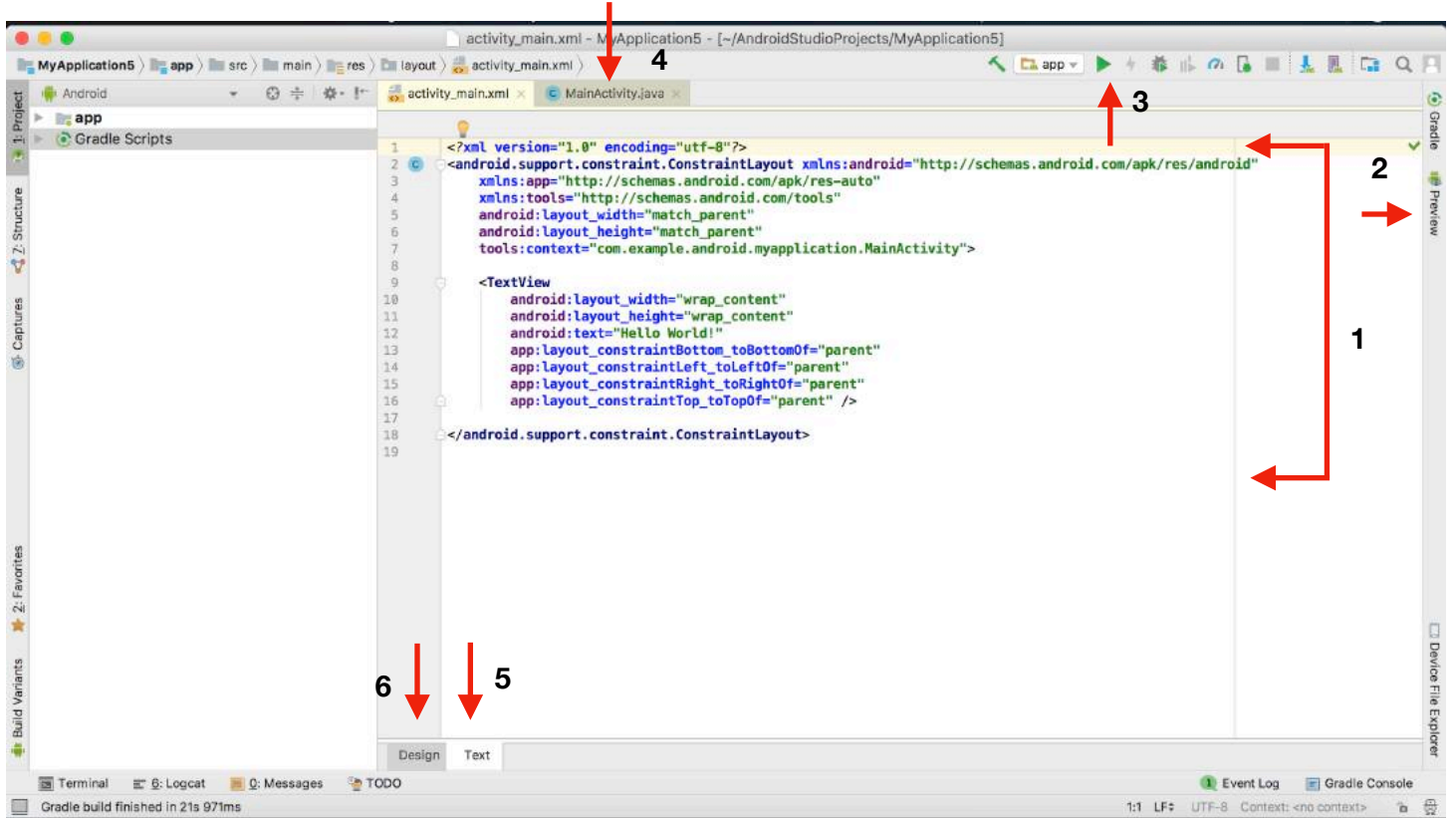
android:layout_marginBottom="8dp"

لاحظنا في هذا الكود تم اضافة layout وذلك كما سبق وذكرنا انه الهامش يستخدم لوضع فراغات في العرض داخل مجموعة العرض ViewGroup.

الدرس الرابع : مجموعة الممارسة (انشاء التخطيطات)

في هذا الدرس والدروس القادمة ان شاء الله سيكون تطبيق الكودات على Android Studio

لمحات سريعة على منصة اندرويد ستوديو



- 1 مكان كتابة الاكواد
- 2 العرض او Preview وهو لعرض ماكتبته ليظهر على شكل شاشة موبايل
- 3 لتشغيل النص البرمجي على موبايلك او محاكي الموبايل
- 4 سيتم التطرق اليه بشكل اوسع في الدروس القادمة
- 5 مثل ماوضحنا انه مكان كتابة الكود
- 6 وهو المكان الذي تستطيع منه اختيار ماتريد اضافته في الكود من نص او صور...الخ

تطبيق بطاقة التهئة بعيد الميلاد

هذه هي النصوص البرمجية لعمل تطبيق بطاقة تهئة عيد الميلاد

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/androidparty"
        android:scaleType="centerCrop"/>

    <TextView
        android:text="Happy Birthday, Android"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="36sp"
        android:fontFamily="sans-serif-light"
        android:textColor="@android:color/white"
        android:padding="20dp"/>

    <TextView
        android:text="From Omar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:textSize="36sp"
        android:fontFamily="sans-serif-light"
        android:textColor="@android:color/white"
        android:padding="20dp"/>

</RelativeLayout>
```



وهذا هو الشكل النهائي للتطبيق

ملاحظة مهمة جدا : عند كتابة الاكواد يجب مراعاة الترتيب فمثلا في الكود السابق اذا قمنا بكتابة عرض النصوص TextView وبعدها كود عرض الصور ImageView فإن الصورة ستغطي عرض النصوص لان في الاندرويد الاستوديو يتم عرض الاكواد حسب الترتيب وهذا مثال :

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:src="@drawable/androidparty"
    android:scaleType="centerCrop"/>
```

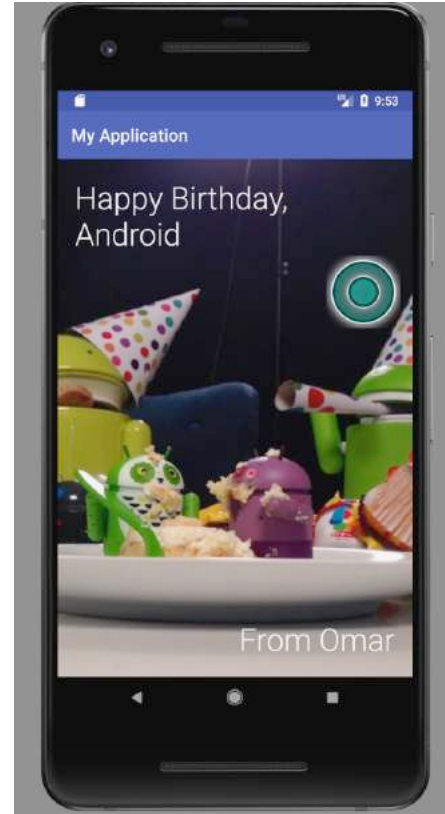
1

```
<TextView
    android:text="Happy Birthday, Android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="36sp"
    android:fontFamily="sans-serif-light"
    android:textColor="@android:color/white"
    android:padding="20dp"/>
```

2

```
<TextView
    android:text="From Omar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:textSize="36sp"
    android:fontFamily="sans-serif-light"
    android:textColor="@android:color/white"
    android:padding="20dp"/>
```

3



الدرس الخامس : تحدي القهوة

السؤال الاول: اذا اردت اظهار صورة على جهازي اي وسيلة عرض يجب ان استخدم؟

ج/ **ImageView**

تلميح 2-is

السؤال الثاني: android:layout_width, android:text, android:src هي امثلة على ماذا؟

ج/ **Attributes**

تلميح 5-lead

السؤال الثالث: اكتب الجملة التالية "i like practive sets the best" بصيغة Camel Case او مايعرف بسنام الجمل.

ج/ **ILikePracticeSetsTheBest**

تلميح 7-on

السؤال الرابع: Linear Layout & Relative Layout ماذا نعني بهم؟ او بماذا يعرفون؟ او ماهي

اسمائهم؟

ج/ **view group**

تلميح 1-who

السؤال الخامس: انا دائما احتاج الى — و الى — التذييلات الخاصة بي.

ج/ انا دائما احتاج الى (فتح) والى اغلاق) التذييلات الخاصة بي

تلميح 3-the

السؤال السادس: ماهي اسماء المعلمين الاثنين (يقصد المعلمين من Udacity خلال هذا الكورس)؟

ج/ **Katherine & Kunal**

تلميح 10-webpage

السؤال السابع: في اي سطر يوجد خطأ برمجي؟

ج/ 21

21 android:layout_weight=1

تلميح 9-materials

```
<RelativeLayout
    android:layout_width="350dp"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Cup of Coffee" />
</RelativeLayout>
```

السؤال الثامن: ماهو عرض width النص

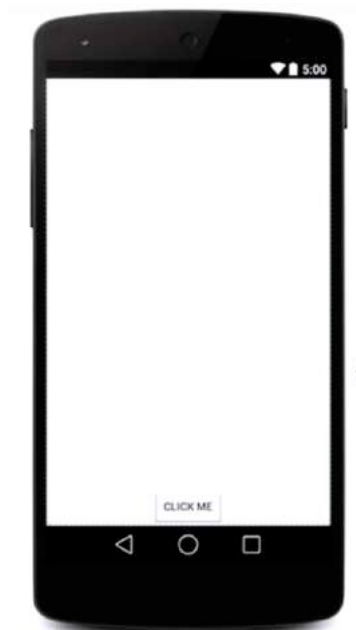
Textview

ج/ **350dp**

تلميح 8-the

السؤال التاسع: ماهو المفقود في هذا الكود؟ لكي يكتمل ويصبح مثل مافي الصورة؟

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click me"
        android:layout_weight="0"
```



ج/

`android:layout_alignParentBottom="true"`

`android:layout_centerHorizontal="true"`

وهنا يقصد ماهو النقص في كود الزر Button

تلميح look-6

السؤال العاشر: لنفترض انك تستخدم XML وتمتلك شاشة بطول 600dp، ماهو ارتفاع النص
Chai Tea ؟
ج/ 100dp
تلميح 4-Production

السؤال الاخير: هذا السؤال مرتبط بالتلميحات التي ظهرت عند الاجابة عن كل سؤال بصورة
صحيحة!

السؤال هو Who am i making this coffee to اي (لمن اقوم بإعداد هذه القهوة؟)
وبجمع التلميحات وترتيبها حسب التسلسل

ج/ Who is the production lead look on the materials webpage هذه مجموعة التلميحات
والجواب هو Kagure Kabue

الدرس السادس جعل التطبيق تفاعليا الجزء الاول

انشاء تطبيق Just Java

من خلال انشاء هذا التطبيق سنقوم بأضافة سطر برمجي جديد ووسيله عرض جديدة تدعى بالزر Button والكود البرمجي الخاص بها هو:

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="Order"
    android:onClick="submitOrder"/>

```

لدينا سطر برمجي جديد وهو لتنفيذ الكود عند الضغط على الزر order

```

<LinearLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/quantity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:text="Quantity"
        android:textAllCaps="true"
        android:textSize="16sp"
        android:padding="10dp"/>

    <TextView
        android:id="@+id/quantity_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textColor="@android:color/black"
        android:textSize="16sp"
        android:padding="10dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Order"
        android:onClick="submitOrder"/>

</LinearLayout>

```

النص البرمجي الكامل لتطبيق Just Java وفي الاسفل الصورة للتطبيق



صورة التطبيق

تبويب او صفحة
تبويب

الان سوف نقوم بالانتقال من
الى **activity_main.xml**
MainActivity.java



```
package com.example.android.justjava;

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

/**
 * This app displays an order form to order coffee.
 */
public class MainActivity extends AppCompatActivity {

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

    /**
     * This method is called when the order button is clicked.
     */
    public void submitOrder(View view) {
        display( number: 1);
    }

    /**
     * This method displays the given quantity value on the screen.
     */
    private void display(int number) {
        TextView quantityTextView = (TextView) findViewById(R.id.quantity_text_view);
        quantityTextView.setText("" + number);
    }
}
```

عند كتابة الكود السابق والضغط على زر **Order** سيتم تحويل رقم **Quantity** من 0 الى 1 ولكن هناك اشياء تحدث خلف الكواليس وكيف لزر ان يقوم بتغيير لكن قبل مناقشة كل هذه الامور وكيف تمت يجب علينا معرفة ماذا يوجد في تبويب او صفحة

MainActivity.java

والمقارنة بينهم ومعرفة متى نستخدم كل منهم

الان لناقش الكودات في **XML** وملف ال **Java** وماهو الربط بينهم ولنفترض ان كود الزر **Button** هو رقم واحد وكود **عرض النص** هو رقم اثنان وكود ال **public void** هو رقم ثلاثة وكود **private void** هو رقم اربعة هذا ماسيحصل عند الضغط على زر **Order** ان ماسيحصل هو ظهور رقم 2 على الشاشة.

لنكرر العملية بوضوح مرة ثانية :

- زر ال **Order** رقم 1
 - كود عرض النص رقم 2
 - كود **public void** رقم 3
 - كود **private void** رقم 4
- الان السؤال الذي سيتبادر الى ذهنكم كيف ستتم هذه العملية وهو بمشاهدة المخطط التالي في الصفحة التالية.

1

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="Order"
    android:onClick="submitOrder"/>
```

عند الضغط على الزر هنا سيتم تنفيذ الكود واطهار رقم ٣ كما تشاهد في الكود والذي بدوره سيرسل اشعار الى رقم ٤ لاطهار النص على الشاشة

```
<TextView
    android:id="@+id/quantity_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="0"
    android:textColor="@android:color/black"
    android:textSize="16sp"
    android:padding="10dp"/>
```

2

3

```
public void submitOrder(View view) {
    display( number: 2);
}
```

```
private void display(int number) {
    TextView quantityTextView = (TextView) findViewById(R.id.quantity_text_view);
    quantityTextView.setText("" + number);
}
```

4

الآن ستتسائلون ماهو دور رقم 2 الجواب ببساطة انه تم اضافة معرف في عرض النص التابع الى رقم 2 وبعدها تم اضافة هذا المعرف الى **private void** وذلك عند القيام بالعملية السابقة عند الضغط على الزر سيتحول الرقم الموجود على الشاشة من 0 الى 2

الآن سنبدأ باستخدام المتغيرات **variables** في تطبيقنا لتغيير عدد اكواب القهوة التي سيتم طلبها (السبب من استخدام المتغيرات في التطبيق لانه حالياً لايمكننا تعديل عدد اكواب القهوة المطلوبة حيث في الكود حالياً العدد ثابت 2 اي كوبان فقط! ، سنستخدم المتغيرات لزيادة ونقصان الاكواب المطلوبة وهذا سيتم عبر عدة خطوات).

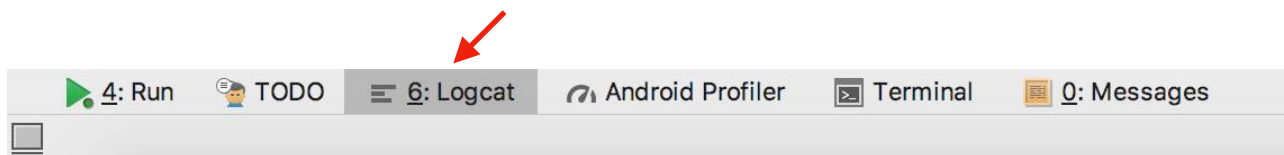
قبل ان نبدأ لناخذ لمحة سريعة على المتغيرات:
المتغيرات هي اشبه بصندوق يتم وضع عدد صحيح **int** فيه ويمكن تسميته باي اسم للرجوع اليه لاحقاً.

```
public void submitOrder(View view) {
    int numberOfCoffees = 2;
    display(numberOfCoffees);
    displayPrice( number: numberOfCoffees*5);
}
```

كما نلاحظ في الكود السابق:

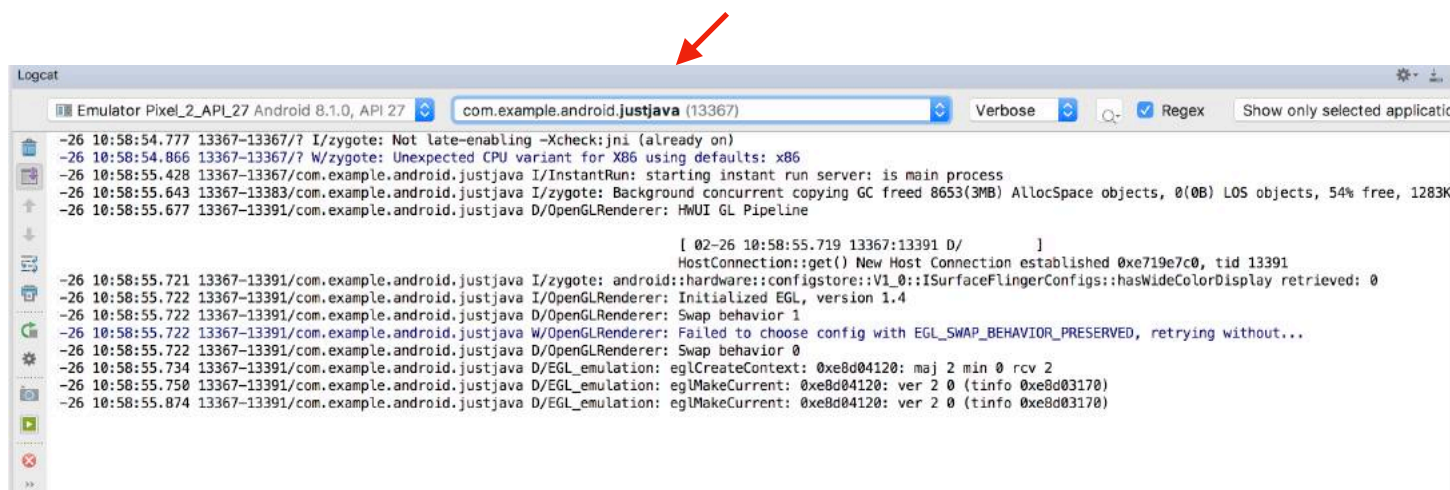
int هو المتغير

numberOfCoffees هو اسم المتغير يرجى الانتباه اننا استخدمنا طريقة **CamelCase**
2 هو القيمة المضافة للمتغير



لتشخيص او مشاهدة الاخطاء اللتي تظهر عن عمل التطبيق او عند تشغيل التطبيق سيقوم بالاغلاق الفوري مع ظهور رسالة خطأ اي بمعنى Crash للتطبيق يجب علينا معرفة الخطأ وتشخيصه يمكننا معرفة الخطأ بالضغط على logcat في منصة اندرويد ستوديو وبعد ذلك اختيار تطبيقنا ومن هناك ستنتم الفلتره ويمكننا قراءة الاسطر باللون الاحمر وتشخيص الخطأ وماكتبته سابقا

عند الضغط على logcat ستنبتق نافذة من خلالها يمكن اختيار تطبيقنا وبعدها مشاهدة الاخطاء ان وجدت



ومن هنا نختار تطبيقنا وكما تلاحظون انه ينتهي بأسم التطبيق Justjava حيث سيكون من الاسهل تشخيص الاخطاء لأنه قمنا بالفلتره واختيار تطبيقنا

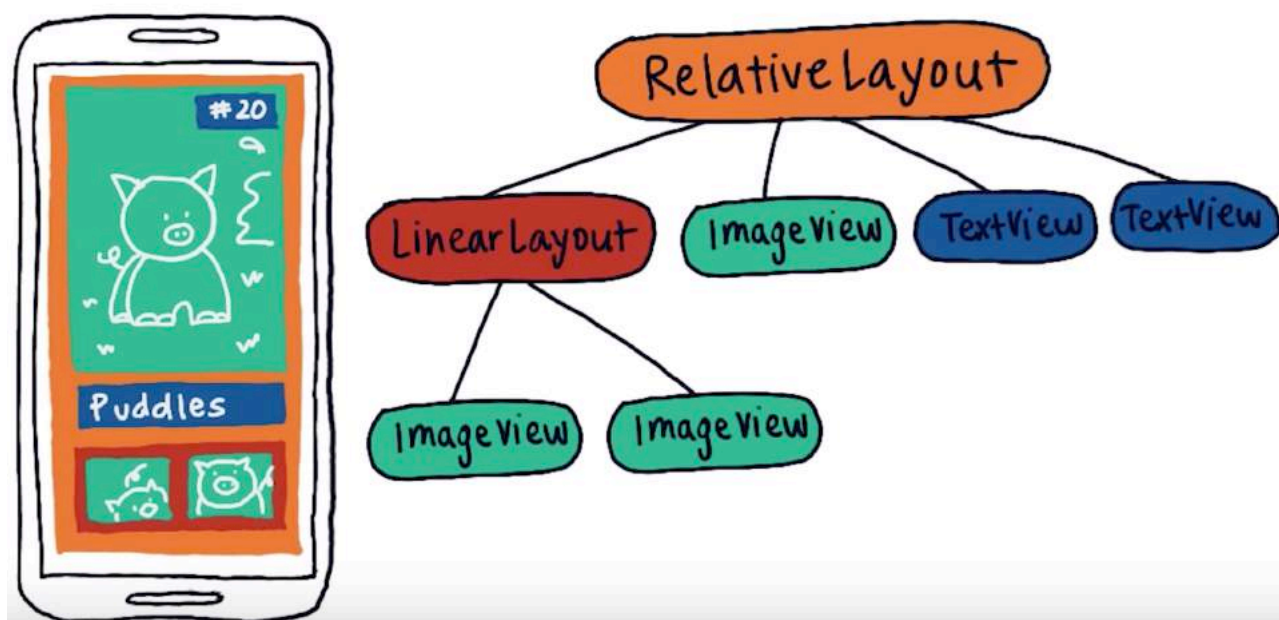
معلومات اكثر يمكنك الدخول للرابط ادناه لمعرفة امور مرتبطة ب **debugging**
<https://developer.android.com/studio/debug/index.html>

الدرس السابع : جعل التطبيق تفاعليا (الجزء الثاني)

في هذا الدرس تم ذكر مجموعة عرض جديدة وهي **Nested groups** وتعني المجموعات المتداخلة اي بمعنى تداخل **Linear layout** مع **Relative layout**.

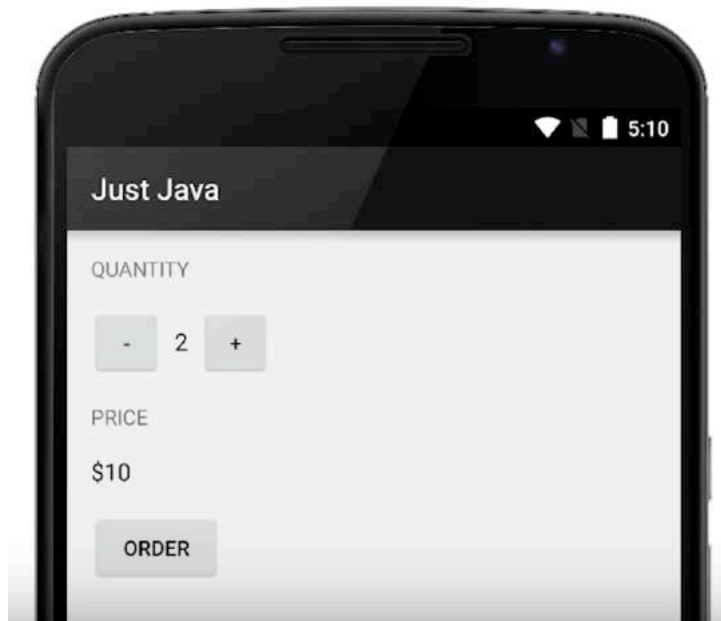
في هذا الدرس تم ذكر مصطلح **view hiarcy diagram** ويعني عرض المخطط الهرمي ويقصد به الاستعانة بورقة ورسم مايعنيه كل كود برمجي موجود **XML**

مثال على عرض المخطط الهرمي

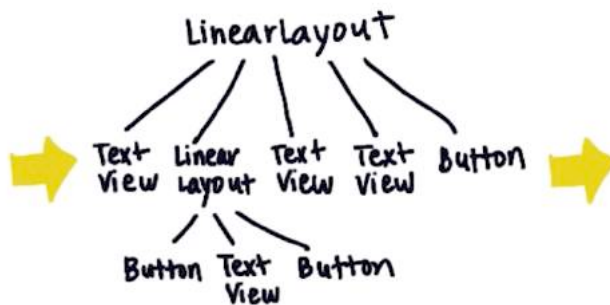


ومن المثال السابق يمكننا ان نرى كيف تداخل التصميم النسبي **Relative layout** مع التخطيط النسبي **Linear layout**.

في الدرس السابع سيطلب من عندنا تعديل التطبيق وجعل ازرار الكمية بهذا الشكل:



ولكن قبل البدء بهذا يجب علينا ان نعلم ماهي مجموعات العرض التي يجب ان نستخدمها لكي نصل إلى النتيجة اعلاه.



```

<LinearLayout ...>
  <TextView... />
  <LinearLayout ...>
    <Button... />
    <TextView... />
    <Button... />
  </LinearLayout>
  <TextView... />
  <TextView... />
  <Button... />
</LinearLayout>
  
```

String السلاسل النصية

كيفية انشاء متغير سلسلة نصية string variable

مثال:

```
String message = "free";
```

↑ ↑ ↑

قيمة المتغير اسم المتغير ونوع المتغير

ملاحظة مهمة عند كتابة المتغير ويوجد هناك علامات اقتباس داخل علامات الاقتباس الاصلية سيظهر خطأ ولكي اوضح ماذا اعني شاهد المثال

```
String message = "she said "one dollar"";
```

في هذه الحالة سيظهر عندك خطأ ولمعالجة هذا الخطأ يرجى اضافة هذه العلامات \\
مثال

```
String priceMessage = "she said \"one dollar\"";
```

ولتأكد ان كل شئ صحيح سيتحول اللون الى البنفسجي وهنا اقصد علامات \\
علامات الاقتباس كما في الصورة ادناه.

```
String priceMessage = "She said \"1 dollar\"";
```

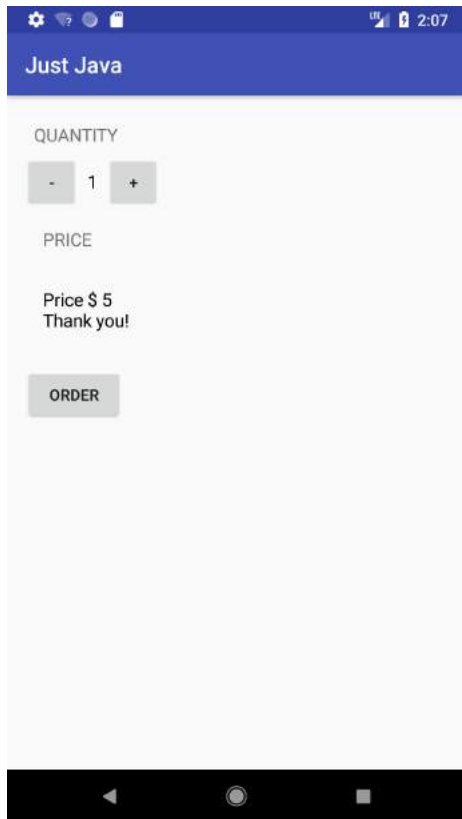
ولاتنسى الفاصلة المنقوطة لان الكود لن يعمل

لاننسى عندما نكتب **String** انه يجب ان تكون في الحروف الكبيرة **Capital letters**
حيث من الممكن انك تقول لماذا كتبنا **int** بأحرف صغيرة و **String** بأحرف كبيرة والسبب لكتابة **int** بأحرف صغيرة لان **int** هو نوع بدائي من البيانات بمعنى انه نوع من انواع بيانات جافا الاساسية.

مثال اخر لكيفية الدمج بين سعر طلب كوب القهوة مع كلمة **Price**

```
public void submitOrder(View view) {  
    String priceMessage = "Price $" + (quantity*5);  
    displayMessage(priceMessage);  
}
```

لمعرفة معلومات اكثر ابحت في الكوكل عن Java escape sequence



لاضافة سطر جديد مثل في الصورة في كلمة

Thank you!

يمكنك استخدام هذا الكود في الاسفل

```
public void submitOrder(View view) {
    String priceMessage = "Price $ " + (quantity*5) + "\nThank you!";
    displayMessage(priceMessage);
}
```

الدرس الثامن

جعل التطبيق تفاعليا

ادخل الرابط ادناه لمعرفة الكلمات الرئيسية في الجافا:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

تجربة تطبيق Sleep Debt كود الجافا

```
Override@
} protected void onCreate(Bundle savedInstanceState)
;super.onCreate(savedInstanceState)
;setContentView(R.layout.activity_main)
PASTE CODE YOU WANT TO TEST HERE //
;int weekday = 5
;int weekend = 9
;int optimalHours = 7 * 8

;int actualHours = weekday
;actualHours = actualHours + weekend * 2
;int solution = optimalHours - actualHours
;display(solution)
{
```

كود XML

```
TextView>
"android:id="@+id/display_text_view
"android:layout_width="wrap_content
"android:layout_height="wrap_content
""=android:text
</ "android:textSize="45sp
```

عند تشغيل البرنامج سيظهر الرقم 33 وهو يعني عدد الساعات

السبب من وضع المثال السابق هو لظهور عند اضافة العديد من المتغيرات حيث سيقوم الحاسوب بحساب كل شيء و اظهار الناتج الاجمالي.

اختبار اوقات السفر

وهو لحساب عدد الدقائق المقطوعة بالسيارة الى العمل او اماكن اخرى في الاختبار سيطلب من عندنا حساب الدقائق لثلاثة ايام ومن ثم تقسيمها على ثلاثة لكي يظهر لدينا معدل الدقائق!

كود Java

```
int day1 = 15;
int day2 = 22;
int day3 = 18;
display(day1 + day2 + day3 / 3);
```

اذا قمنا بعمل الحسابات ذهنيا او بالالة الحاسبة الناتج يجب ان يكون 18.3 لكن في الكود اعلاه هناك خطأ في ترجمة مانحاول فعله والكود الصحيح يجب ان يكون كالتالي:

```
int day1 = 15;
int day2 = 22;
int day3 = 18;
display((day1 + day2 + day3) / 3);
```

وضع اقواس لحساب عدد الدقائق في الايام الثلاثة اولا وبعدها تقسيمها على 3 لآظهار المعدل وهذه العملية تعتبر مهمة مستقبلا عند القيام بالعمليات الحسابية في تطبيقنا.

اختبار قائمة التسوق

كود الجافا

```
;int raspberryPrice = 5
;display1("1 box: $" + raspberryPrice)
;raspberryPrice = 10
;display2("2 boxes: $" + (raspberryPrice))
;display3("3 boxes: $" + (raspberryPrice * 3))
```

كما تلاحظون في الكود السابق انه يجب الانتباه على عدة امور انه تم تحديد ال سعر ال raspberry مرة ثانية بقيمة 10 بعد ان كانت 5 وان display2 لا يوجد فيها ضرب للسعر في 2 لانه اصلا تم تحديث السعر.

اختبار تحويل سعر العملات

كود الجافا

```
int dollars = 40;
    int dollarsToYen = 119;
    int yen = dollarsToYen * dollars;
    display(yen);
```

الاختبار السابق صحيح والنتيجة 4760

اختبار دليل البريد الاليكتروني

```
String firstName = "Lyla";
String lastName = "Fujiwara";
String contactInfo = firstName + " " + lastName;
contactInfo = "<" + lastName + "." + firstName +
"@justjava.com>";
display(contactInfo);
```

ما يجب ان يظهر في هذا الاختبار هو:

Lyla Fujiwara <fujiwara.Lyla@justjava.com>

لكن في الكود السابق لم يظهر هذا الامر بصورة كاملة لذا يمكننا تعديل الكود اعلاه الى:

```
String firstName = "Lyla";
String lastName = "Fujiwara";
String contactInfo = firstName + " " + lastName;
contactInfo = contactInfo + "<" + lastName + "." +
firstName + "@justjava.com>";
display(contactInfo);
```

اي اننا اضعنا **contactInfo** في هذا السطر

```
contactInfo = contactInfo + "<" + lastName + "." + firstName
+ "@justjava.com>";
```

لكي نحصل على النتيجة المطلوبة

اختبار تطبيق Cola Wars

```

public void voteUdacicola(View view) {
    int udacicolaVotes = 0;
    udacicolaVotes = udacicolaVotes + 1;
}

public void votePepcity(View view) {
    int pepcityVotes;
    pepcityVotes = pepcityVotes + 1;
}

public void showMeVotes(View view) {
    display(udacicolaVotes + " vs. " + pepcityVotes);
}
...
}

```

في اختبارنا لهذا التطبيق يوجد عدة اخطاء طبعا الاخطاء في الترجمة لما نريد ظهوره على الشاشة وليس اخطاء برمجية بمعنى ليست اخطاء برمجية تؤدي الى اغلاق التطبيق!

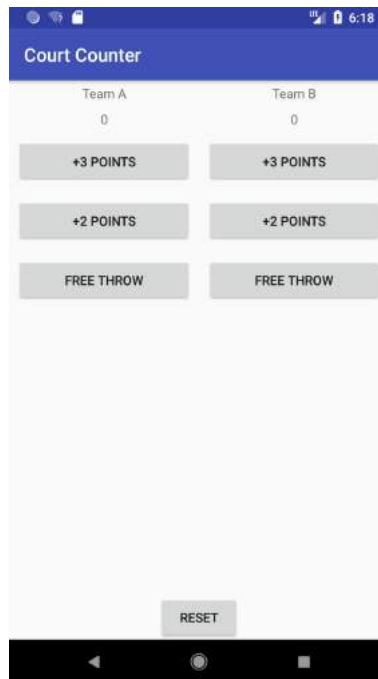
الخطأ الاول وهو تعرف المتغير `udacicolaVotes` حيث تم تعريفه `local variable` وهذا لايجوز حيث يجب ان يكون `global variable` وقد تتسائل لماذا `global`؟ وذلك عند الضغط كل مرة لتصويت على الكولا ستخزن النتيجة ولكن في النهاية عند محاولة معرفة الاصوات الكلية في النهاية لن تظهر!

اما بالنسبة الى الخطأ في `pepcityVotes` نفس الخطأ السابق ويجب ان يكون `global variable` بالاضافة الى ان هناك خطأ اخر وهو انه لم يتم تحديد قيمة اولية ل `pepcityVotes` مثل ما حددنا في `udacicolaVotes` حيث كانت القيمة الاولى = 0

تطبيق Court Counter

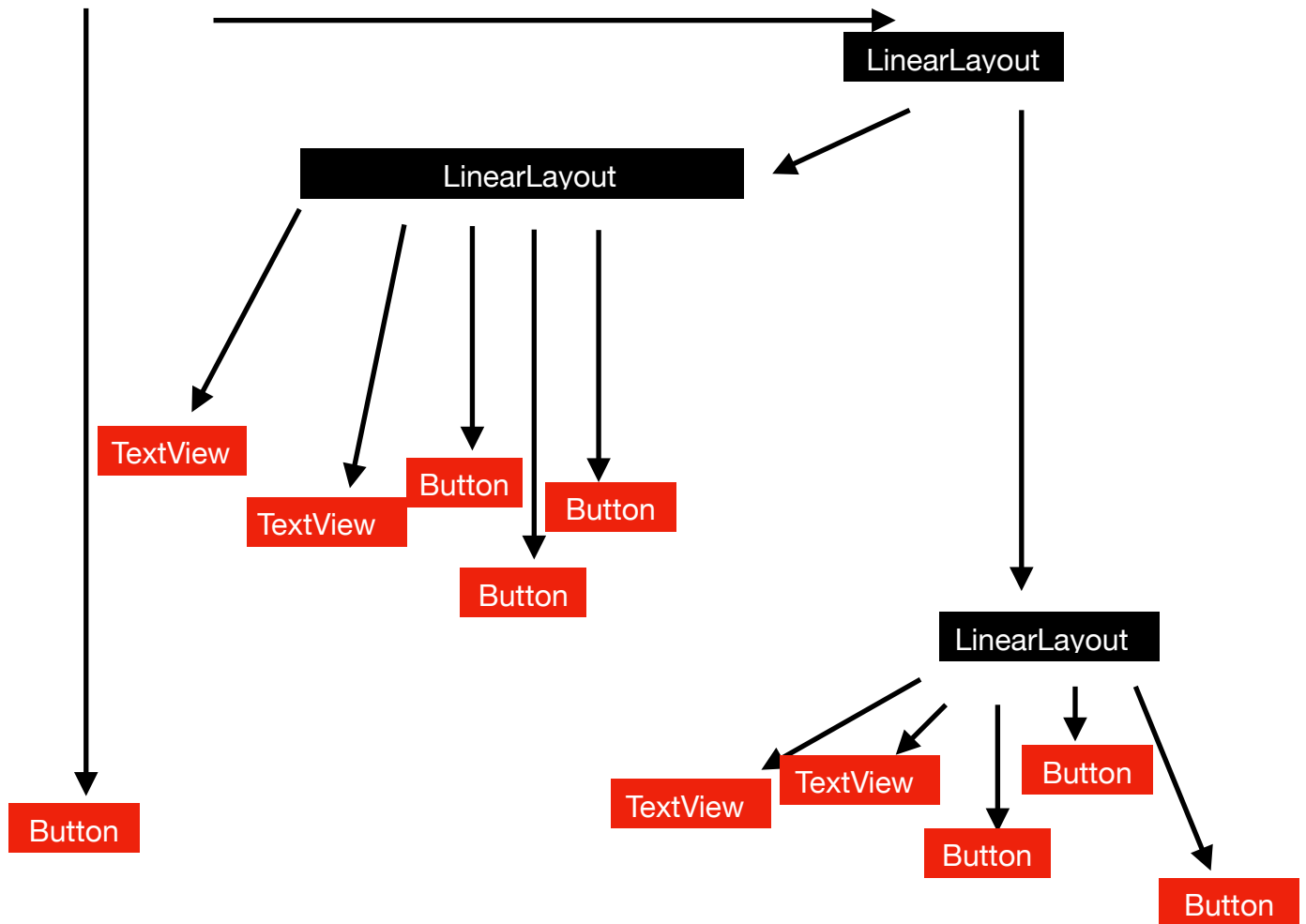
في هذا التطبيق سيتم وضع كود الجافا وكود XML وسيتم مناقشة الامور المهمة والاساسية التي تم التطرق اليها.

في البداية هذه هي صورة التطبيق قبل تعديل ال Styles



من الملاحظات المهمة اللتي تم استخدامها في هذا التطبيق هو اننا استخدمنا **LinearLayout** مرتين مع **RelativeLayout** مرة واحدة وكان هو **Root ViewGroup** السبب من استخدام هذا العدد من **مجموعات العرض ViewGroups** هو اننا قسمنا الشاشة الى قسمنا بالاضافة الى اننا فصلنا القسم الاسفل عن هذين القسمين ولتوضيح اكثر شاهد المخطط:

RelativeLayout



كود XML

```
<?xml version="1.0" encoding="utf-8?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    <android:layout_height="wrap_content
```

```
LinearLayout>
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    <android:orientation="horizontal
```

```
LinearLayout>
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="vertical"
    <"tools:context=".MainActivity
```

```
TextView>
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fontFamily="sans-serif-medium"
    android:gravity="center_horizontal"
    android:padding="16dp"
    android:text="Team A"
    android:textColor="#616161"
    </ android:textSize="14sp
```

```
TextView>
    android:id="@+id/team_a_score"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fontFamily="sans-serif-light"
    android:gravity="center_horizontal"
    android:paddingBottom="24dp"
    android:text="0"
    android:textColor="#000000"
    </ android:textSize="56sp
```

```
Button>
    android:layout_width="match_parent
```

```

"android:layout_height="wrap_content
"android:layout_marginBottom="8dp
"android:layout_marginLeft="24dp
"android:layout_marginRight="24dp
"android:onClick="Add3ForTeamA
</ "android:text="+3 Points

```

Button>

```

"android:layout_width="match_parent
"android:layout_height="wrap_content
"android:layout_marginBottom="8dp
"android:layout_marginLeft="24dp
"android:layout_marginRight="24dp
"android:onClick="Add2ForTeamA
</ "android:text="+2 Points

```

Button>

```

"android:layout_width="match_parent
"android:layout_height="wrap_content
"android:layout_marginBottom="8dp
"android:layout_marginLeft="24dp
"android:layout_marginRight="24dp
"android:onClick="Add1ForTeamA
</ "android:text="Free Throw
<LinearLayout/>

```

View>

```

"android:layout_width="1dp
"android:layout_height="match_parent
"android:layout_marginTop="16dp
</ "android:background="@android:color/darker_gray

```

LinearLayout>

```

"android:layout_width="0dp
"android:layout_height="wrap_content
"android:layout_weight="1
"android:orientation="vertical
<"tools:context=".MainActivity

```

TextView>

```

"android:layout_width="match_parent
"android:layout_height="wrap_content
"android:fontFamily="sans-serif-medium
"android:gravity="center_horizontal
"android:padding="16dp
"android:text="Team B
"android:textColor="#616161
</ "android:textSize="14sp

```

TextView>

```

"android:id="@+id/team_b_score

```

```

"android:layout_width="match_parent
"android:layout_height="wrap_content
"android:fontFamily="sans-serif-light
"android:gravity="center_horizontal
"android:paddingBottom="24dp
"android:text="0
"android:textColor="#000000
</ "android:textSize="56sp

```

Button>

```

"android:layout_width="match_parent
"android:layout_height="wrap_content
"android:layout_marginBottom="8dp
"android:layout_marginLeft="24dp
"android:layout_marginRight="24dp
"android:onClick="Add3ForTeamB
</ "android:text="+3 Points

```

Button>

```

"android:layout_width="match_parent
"android:layout_height="wrap_content
"android:layout_marginBottom="8dp
"android:layout_marginLeft="24dp
"android:layout_marginRight="24dp
"android:onClick="Add2ForTeamB
</ "android:text="+2 Points

```

Button>

```

"android:layout_width="match_parent
"android:layout_height="wrap_content
"android:layout_marginBottom="8dp
"android:layout_marginLeft="24dp
"android:layout_marginRight="24dp
"android:onClick="Add1ForTeamB
</ "android:text="Free Throw
<LinearLayout/>

```

<LinearLayout/>

Button>

```

"android:layout_width="wrap_content
"android:layout_height="wrap_content
"android:layout_alignParentBottom="true
"android:layout_centerHorizontal="true
"android:layout_marginBottom="8dp
"android:layout_marginLeft="22dp
"android:layout_marginRight="22dp
"android:onClick="resetScore
</ "android:text="Reset
<RelativeLayout/>

```

كود الجافا

```

package com.example.android.courtcounter;

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

public class MainActivity extends AppCompatActivity {

    int scoreTeamA = 0;
    int scoreTeamB = 0;

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

    public void Add3ForTeamA(View view) {
        scoreTeamA = scoreTeamA + 3;
        displayForTeamA(scoreTeamA);
    }

    public void Add2ForTeamA(View view) {
        scoreTeamA = scoreTeamA + 2;
        displayForTeamA(scoreTeamA);
    }

    public void Add1ForTeamA(View view) {
        scoreTeamA = scoreTeamA + 1;
        displayForTeamA(scoreTeamA);
    }

    public void Add3ForTeamB(View view) {
        scoreTeamB = scoreTeamB + 3;
        displayForTeamB(scoreTeamB);
    }

    public void Add2ForTeamB(View view) {
        scoreTeamB = scoreTeamB + 2;
    }
}

```

```

        displayForTeamB(scoreTeamB);
    }

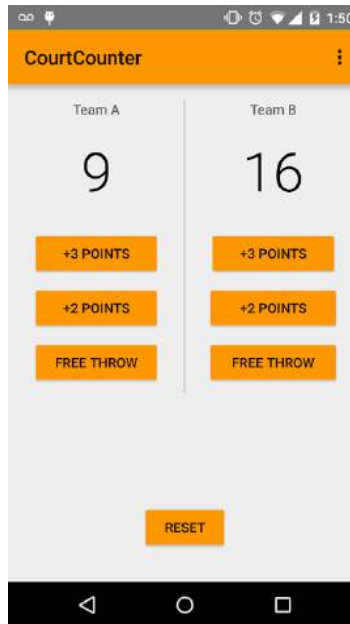
    public void Add1ForTeamB(View view) {
        scoreTeamB = scoreTeamB + 1;
        displayForTeamB(scoreTeamB);
    }

    /** Reset team A and team B score when pressed to 0
        * */

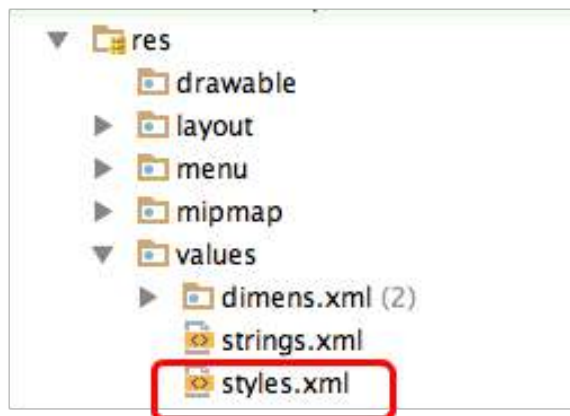
    public void resetScore(View v) {
        scoreTeamA = 0;
        scoreTeamB = 0;
        displayForTeamA(scoreTeamA);
        displayForTeamB(scoreTeamB);
    }
    /**
     * Displays the given score for Team A.
     */
    public void displayForTeamA(int score) {
        TextView scoreView = (TextView) findViewById(R.id.team_a_score);
        scoreView.setText(String.valueOf(score));
    }
    /**
     * Displays the given score for Team B.
     */
    public void displayForTeamB(int score) {
        TextView scoreView = (TextView) findViewById(R.id.team_b_score);
        scoreView.setText(String.valueOf(score));
    }
}

```

طبعا الكودات السابقة هي للبرنامج قبل تعديل **Styles** حيث سيطلب من عندنا عمل التطبيق بهذا الشكل :

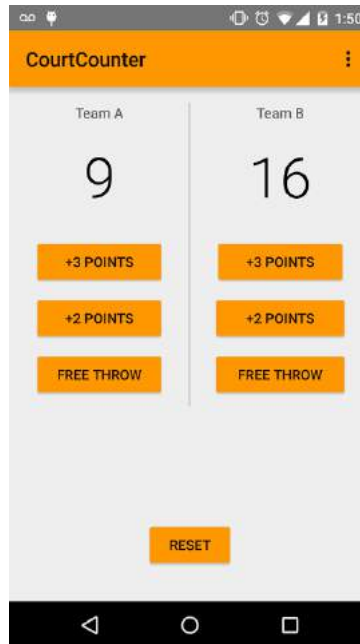


والان اذهب الى ملف **Styles.xml** كما هو موضح في الصورة ادناه:



وانسخ الكود التالي في هذا الملف وسيتحقق ما هو مطلوب في الصورة النهائية لشكل التطبيق:

```
<resources>
<!-- .Base application theme --!>
<"style name="AppTheme" parent="Theme.AppCompat.Light">
<!-- Primary theme color of the app (sets background color of app bar) --!>
<item name="colorPrimary">#FF9800</item>
<!-- Background color of buttons in the app --!>
<item name="colorButtonNormal">#FF9800</item>
</style>
</resources>
```



في النظر الى صورة التطبيق نلاحظ ان هناك **خط رمادي** يفصل ما بين الفريقين ولكي نضيف هذا الخط يتم اضافة هذا الكود:

```
<View
    android:layout_width="1dp"
    android:layout_height="match_parent"
    android:layout_marginTop="16dp"
    android:background="@android:color/darker_gray" />
```

ولكن يرجى الانتباه هناك **3 LinearLayout** في التطبيق ويجب اضافة الكود السابق ما بين **LinearLayout** التابع لفريق **A** و **LinearLayout** فريق **B** ويرجى الملاحظة ان يجب **تغيير** **android:layout_height** في جميع **LinearLayout** الى **wrap_content** والسبب من ذلك هو ان الخط كما تلاحظون لا يستمر الى الاسفل وانما يتوقف عند جملة **Free Throw**

الدرس التاسع (البرمجة الموجهة للكائنات الجزء الاول) OOP (Object Oriented Programming)

في الدرس التاسع تم توجيه نصائح الى المبرمجين حول التعامل مع الطرق **methods** والاكواد الجديدة والبحث عنها في كوكل عند الحاجة وتم وضع اختبار قصير حول حساب السعر وماهي الطريقة الصحيحة لحسابها مع انه لم يتم التطرق الى هذه الطرق والاكواد من قبل وهذا الاختبار يؤدي اولا الي تعزيز الثقة بالنفس من خلال معرفة الكود الصحيح حتى وان كان يمر لأول مرة بالاضافة وان اخطانا فسيكون الخطأ ثميناً حيث لن ينسى هذا الكود ابداً.

الخلاصة ياعزيزي المبرمج لا يوجد ماهو صعب في البرمجة وانما هو عالم مفتوح من التجارب والبحث والسؤال لحين الوصول الى المطلوب.

ولمزيد من المعلومات حول تعريف ال **Method** يمكنك الذهاب الى الموقع التالي:

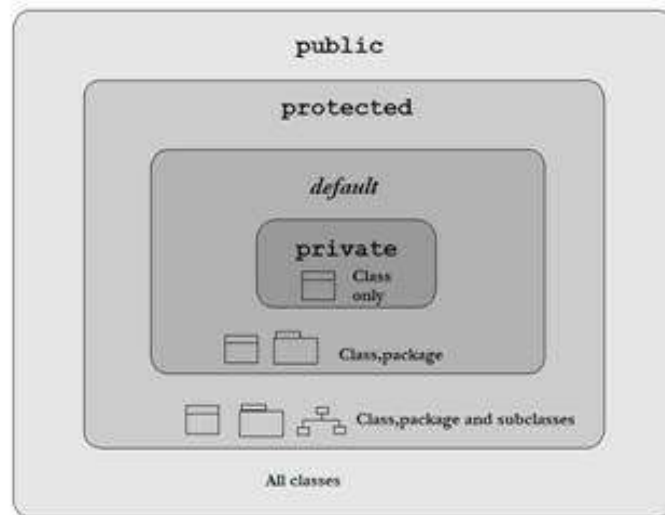
<https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html>

من خلال قراءة الموقع الذي تم ذكره في الاعلى يمكننا فهم ال **method** بصورة اكثر.
مثال:

```
private int calculatePrice(int quantity) {
    int price = quantity * 5;
    return price;
}
```

في هذا المثال نلاحظ عدة امور:

- وجود كلمة **int** قبل جملة **CalculatePrice** وذلك لتعريفها سواء كانت **integer** او **string**
- وجود اقواس () حول جملة **int quantity** وتسمى **parentheses**
- وجود اقواس فتح واغلاق {} وتسمى **Braces**
- عند تسمية المتغيرات او الاسماء يجب ان نتبع اسلوب **camelCase** الذي تم ذكره سابقا وهو ماموجود في جملة **calculatePrice** وتذكر دائما عند التسمية عدم ترك فراغات فهذا من شأنه ان يحدث خطأ يؤدي الى عدم عمل التطبيق
- وجود كلمة **private** وتعرف ب **java access modifier** اي يعني معدل الوصول لجافا والغرض منها هو تحديد من له الصلاحية للوصول الى هذه الوظيفة ونعني بالوظيفة **calculatePrice** (وللاطلاع هناك نوعين من معدلات الوصول **public** و **private**) حيث ان الفرق بينهم هو انه اذا حددت الوظيفة ب **private** ستكون محددة بنشاط محدد وانه سيتم استدعائها فقط من ضمن هذا النشاط اما **public** وهي ستكون غير محددة بنشاط محدد ومن الممكن استدعائها من خارج النشاط ويمكنك مشاهدة الصورة التالية لتوضيح اكثر حولهم من حيث الفرق ويجب ان انوه ان تم ذكر **protected** حيث سيتم التطرق اليها لاحقا ان شاء الله.



- اذا كان لدينا عدة مدخلات **inputs** يجب ان يتم الفصل بينهم بفاصلة , مثال :

calculatePrice(int quantity, int pieceOfOneCup)

- يجب التركيز على ترتيب الوظائف **methods** او حتى ترتيب المدخلات لان من الممكن ان يتم الحساب بصورة خاطئة

تعريف الطريقة او الوظيفة **Define a method**

في هذا الاختبار سيتم شرح كيفية تعريف الوظائف حيث سيتم وضع الاكواد ومن تحتها جدول يشرحها جميعها.

Method 1

```
private String createCalendarEventReminder(String eventName,
String location, int minutesAway) {
    String reminder = "You have an upcoming event in " +
minutesAway + " minutes.";
    reminder = reminder + " It is " + eventName + " held at "
+ location + ".";
    return reminder;
}
```

Method 2

```
private int deductPoints(int pointsUsed) {
    // Everyone starts with 100 points
    int numberOfPoints = 100 - pointsUsed;
    return numberOfPoints;
}
```

Method 3

```
private String findTotalTripLength(int distanceOfFirstTrip,
int distanceOfSecondTrip, int distanceOfThirdTrip) {
    // Assume we need 2 miles to go to our friend's home
    (where the trip will start).
    int totalLength = 2;

    // Then start adding in the trip length.
    totalLength = totalLength + distanceOfFirstTrip +
distanceOfSecondTrip + distanceOfThirdTrip;

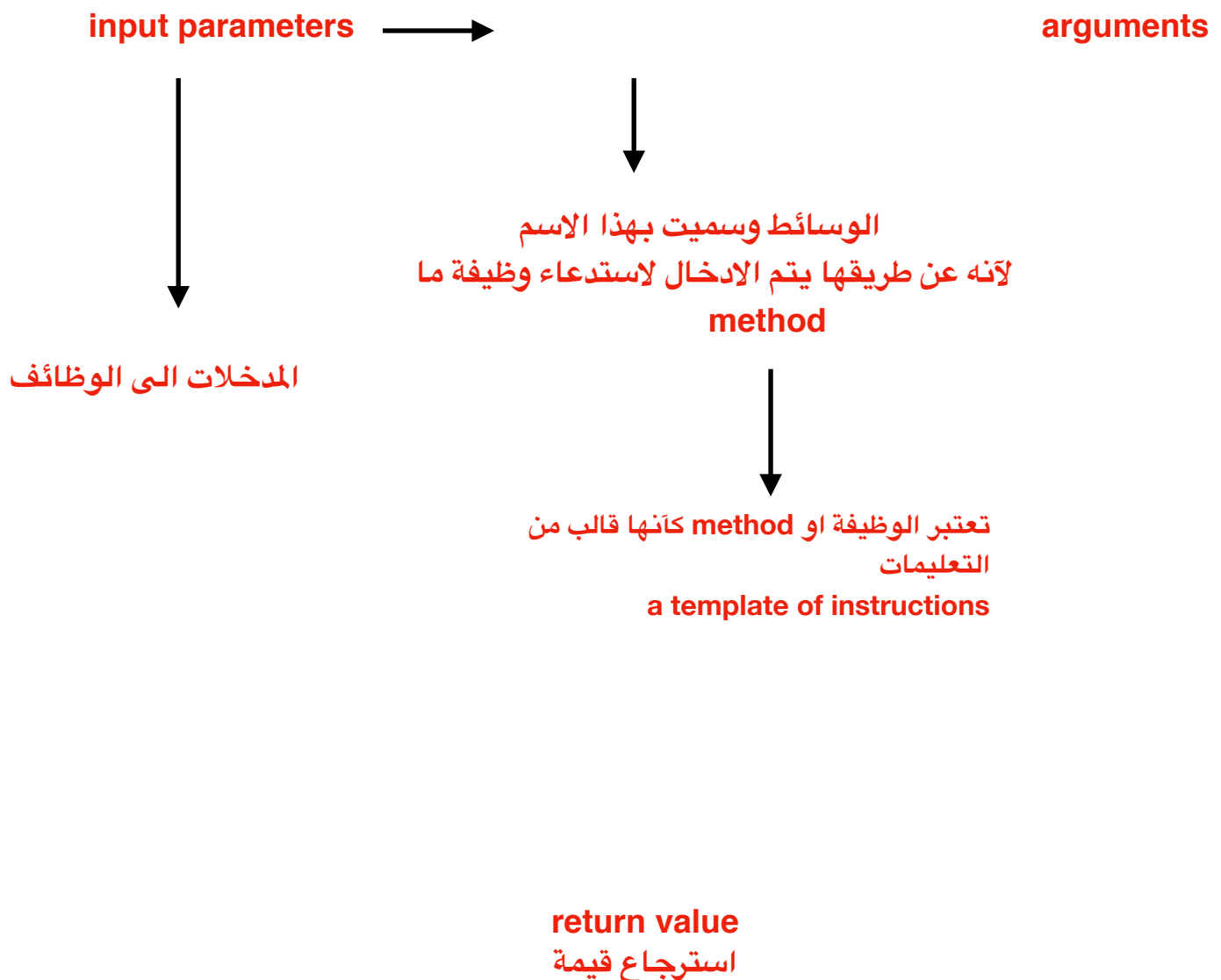
    // Form a string to display the total trip length.
    String message = "The total trip will be: " +
totalLength + " miles.";
    return message;
}
```

	Method Name	Input Parameters	Return Data Type
Method 1	createCalendarEventReminder	String eventName, String location, int minutesAway	String
Method 2	deductPoints	int pointsUsed	int
Method 3	findTotalTripLength	int distanceOfFirstTrip, int distanceOfSecondTrip, int distanceOfThirdTrip	String

تمرير المعلومة الى الوظيفة (الادخالات الى الاسلوب) Passing information to method

يمكنك الدخول الى الرابط التالي لمعلومات اكثر حول هذا الموضوع:

<https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html>



لمعلومات اكثر يمكنك الدخول الى الرابط التالي:

<https://docs.oracle.com/javase/tutorial/java/javaOO/returnvalue.html>

عند اضافة اسم متغير يجب مراعاة عدم تشابهها مع كلمات جافا الدلالية **Java Keywords** في الاسفل هناك مجموعة من كلمات جافا الدلالية :

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum***	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

هناك بعض الكلمات تحتوي على * بجانبها ففي الاسفل تفسير ماتعني:

*	وتعني لم تستعمل not used
**	تم اضافتها في الاصدار ١.٢ 1.2 added in
***	تم اضافتها في الاصدار ١.٤ 1.4 added in
****	تم اضافتها في الاصدار ٥.٠ 5.0 added in

يرجى الملاحظة ايضا انه في حال وجود كلمة **void** التي تأتي مع **public** او **private** فتعني انه لا يوجد قيمة استرجاع **return** فإذا اضيفت **return** في الاخير مع طريقة **private void** مثلاً فسيكون هناك خطأ

الان هناك اربع طرق **methods** تحتوي على اخطاء سيتم وضع الطرق وسيتم وضع جدول يبين الخطأ في اي سطر وذكر سبب الخطأ بصورة مختصرة:

Method 1

```
/**
 * Get the email account name.
 *
 * @return the name of the account.
 */
private String getAccountName() {
    return "android@gmail.com";
    return "droid@gmail.com";
}
```

Method 2

```
/**
 * Add $4 of tip onto the current bill.
 *
 * @return the total price of the bill (including tip).
 */
private addTip(int bill) {
    return bill + 4;
}
```

Method 3

```
/**
 * Sets up the app for the current city.
 */
private nothing setup() {
    cityName = "London";
}
```

Method 4

```
/**
 * Get the number of students in a class.
 *
 * @return the number of students.
 */
private int getStudentClassSize() {
    return "20";
}
```

الطريقة Method	Line # رقم السطر	الوصف Description
Method 1	8	الخطا الاول هو ان return لايمكن اي يأتي بعدها اي شي بالاضافة الى انه تكرر الى عنوان بريد اليكتروني خاطئ
Method 2	6	الوظيفة method تفتقر الى int وهي لتحديد نوع اسم الطريقة private int addTip(int bill)
Method 3	4	يجب اضافة void بدلا من nothing
Method 4	7	رقم 20 يجب ان يكون بدون علامات تنصيص

لكتابة اكواد اكثر اختصارا يمكنك الدخول الى الرابط التالي:

http://www.tutorialspoint.com/java/java_basic_operators.htm

اختبار display message

في تطبيق Just Java قمنا بعمل العديد من التعديلات على الطرق والان سنقوم بتعديل price الموجود في التطبيق السابق وجعله Order Summary اي يعني ملخص الطلب واستخدمنا كلمة ملخص لانه قمنا باضافة الاسم والكمية والسعر وهذا لايتناسب مع كلمة price لوحدها. (ملاحظة هذه الامور البسيطة تؤثر على مدى قوة تطبيقك ليس من ناحية الاكواد او الطرق وانما من ناحية التصميم وصياغة العبارات المناسبة التي تجعل من تطبيقك سلس ومنظم قدر الامكان)

من ضمن اختبار display message يتم الشرح عن الموارد resources وبالاخص في display message هناك موارد يمكن معرفة ماذا تعني وبماذا نستفيد منها عند قراءة الكود.

مثال:

```
private void displayQuantity(int numberOfCoffees){
    TextView quantityTextView=(TextView)findViewById(R.id.quantity_text_view);
    quantityTextView.setText(""+numberOfCoffees);
}
```

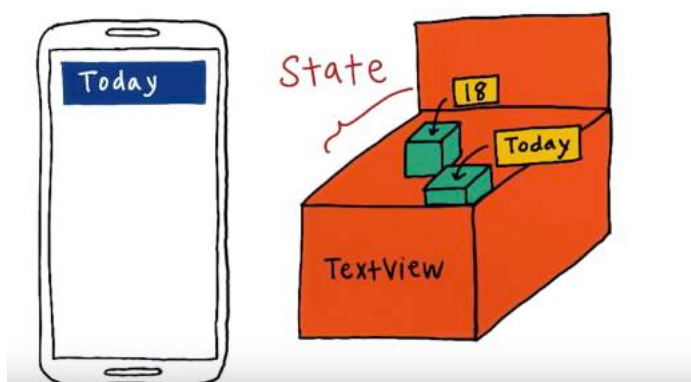
في هذا الكود يمكن معرفة المورد عن **R**. وكما نلاحظ ان هناك كلمة **id** وهي تشير الى انه مصدر لل **id** لل **quantity_text_view** بمعنى اخر اننا نشير الى المصدر بحرف **R**. وبعدها نشير ما هو موجود في ملف **XML** ليظهر عندنا الكود السابق وهي لربط ما هو موجود في **XML** و **Java**

Java Objects

كائنات جافا

Java object has a state & method كائن الجافا لديه حالة وطريقة حيث الحالة مكونة من عدة حقول والحقل هنا هو متغير **variable**

مثال:



كما ترون في هذه الصورة انه كلمة **Today** الموجودة على شاشة الهاتف هي **TextView** وتحتوي على عدة حقول منها **حجم النص ولونه ونوعه والكلمة نفسها Today** تعتبر ايضا حقل.

وللتفاعل مع كائن الجافا لدينا عدة طرق لاستدعائه مثل:

setText()
getText()

وهناك طرق للاستدعاء مثل:

getTextSize
setTextSize

كل وظيفة من هذه الوظائف هي لتنفيذ مجموعة من التعليمات

بشكل عام كائن `TextView` يحمل معلومات الحالة عما تحتويه واجهة العرض النص `textView` وبإستدعائنا هذه الوظائف يمكننا التفاعل مع واجهة النص هذه.

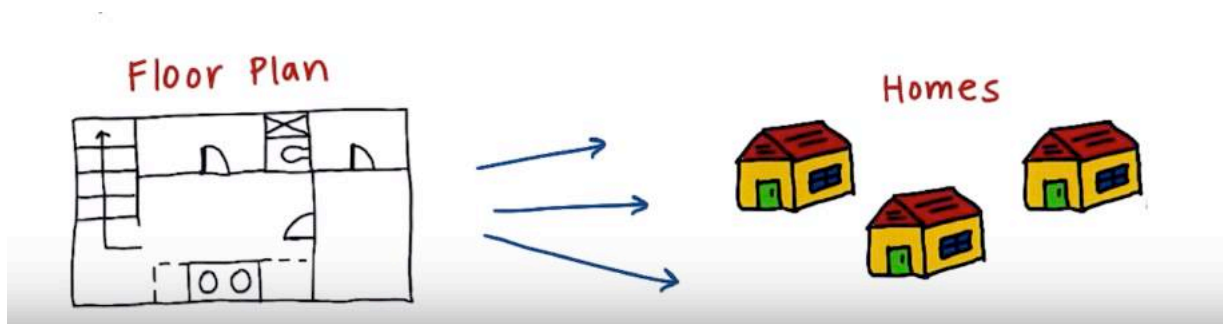
الآن السؤال هو كيف نعرف ماهي الحقول الموجودة داخل حالة `textView` او `ImageView`؟
ويمكننا الإجابة على هذا السؤال هو بالالقاء نظرة على تعريف هذا الكائن

class definition for that object

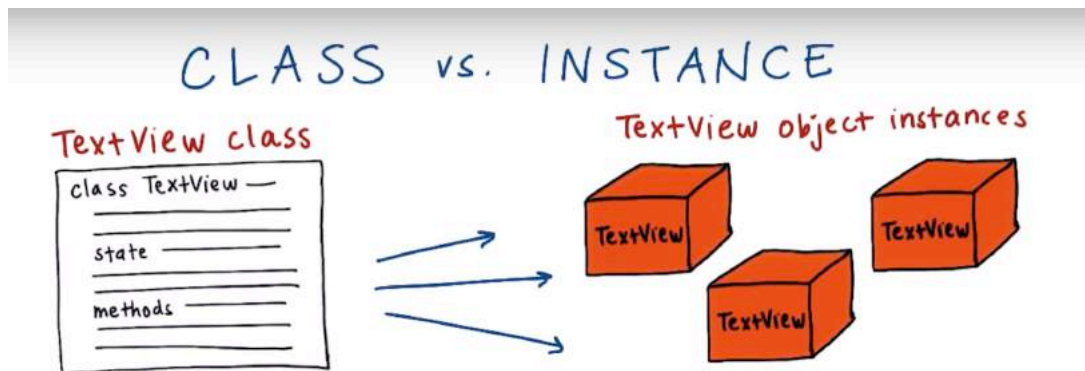
وهو ملف يدعى `TextView.java` وهو يشبه تعريفنا لملف `MainActivity.java`

ففي هذا الملف هناك معلومات عن الحالات والوظائف حيث يمكننا ان نحدد قواعد تصرف `TextView` مثلاً.

والآن لتبسيط الامر بصورة اكثر! يمكننا ان نعتبر هذا الملف اشبه بمخطط طوابق المنزل



كما تلاحظون في الصورة عند تخطيط خارطة المنزل يمكننا استخدام هذا التخطيط لبناء الطوابق داخل المنزل وتكون الطوابق مختلفة قليلاً ولكنها مبنية وفقاً للمخطط الذي قمنا بإنشاءه حيث ان هذا المخطط هو ليس منزلاً بحد ذاته وإنما هو مخطط لما نريد ان يكون عليه المنزل وهذا الشيء ينطبق أيضاً على ملف `TextView.java`



كما نلاحظ في هذه الصورة السابقة ان **TextView class** هي اشبه بمخطط المنزل وقد تفرعت منه **TextView** وهي ايضا اشبه بالطوابق فان فئة **TextView** او **TextViewClass** يتم انشائها لإنشاء نسخ الكائن هذه والنسخ هنا يقصد بها **instances** بالانكليزية وهي تكون بها القيم النصية الحقيقية مثل قيم الوان النصوص وحجم النصوص..الخ

فئة جافا Java Class

الان لمعرفة معلومات اكثر حول **Java Class** تم وضع كود مبسط لفئة جافا لفهمه بصورة اكثر: للحصول على الكود كاملا: ادخل الرابط التالي

<https://gist.github.com/udacityandroid/47592c621d32450d7dbc>

```
/**
 * Displays text to the user.
 */
public class TextView extends View {

    // String value
    private String mText;

    // Text color of the text
    private int mTextColor;

    // Context of the app
    private Context mContext;

    /**
     * Constructs a new TextView with initial values for text and text color.
     */
    public TextView(Context context) {
        mText = "";
        mTextColor = 0;
        mContext = context;
    }
}
```

كما تلاحظون هنا يوجد **public class** وهو **access modifier** واستخدمنا هنا **public** حتى تستطيع الاشياء الاخرى كالانشطة الوصول الى فئة **TextView** وكلمة **class** هنا لنبين لجافا انه تعريف فئة ثم لدينا ايضا اسم الفئة وهي **TextView** ونلاحظ هنا استخدمنا طريقة **camelCase** وهي عدم وجود فراغات واحرف كبيرة في بداية كل كلمة ويوجد ايضا حاصرة فتح وحاصرة اغلاق والتي تدعى **braces { }** وفي داخل هذه الفئة يوجد هناك العديد من المتغيرات كل منهم يمثل حقل في **TextView** حيث هناك حقل للنص نفسه وحقل للون..الخ وهناك ايضا متغير لحفظ الاشارة الى سياق التطبيق حيث انه موجود للوصول الى المعلومات حوق التطبيق مثلا تستطيع الوصول الى الموارد في التطبيق باستخدام السياق **context**

وظيفة **setText** تأخذ سلسلة حروف **String** كاحدى المدخلات **input** نلاحظ ايضا ان كل سياق يبدأ ب **m** بحرف صغير متبوعة ب **context** حيث ان تعني **member variable** اي تعني عضو متغير ويعني ان جميع الموجودة في فئة **TextView** هي اعضاء متغيرة

```

/**
 * Sets the string value in the TextView.
 *
 * @param text is the updated string to be displayed.
 */
public void setText(String text) {
    mText = text;
}

```

اختبار انشاء كائن

```

/**
 * Sets the text color of the TextView.
 *
 *
 *
 */
public

```

CREATE OBJECT WITH CONSTRUCTOR

Object Data Type Variable name = new Object Data Type (Input Args);

```

public
TextView priceTextView = new TextView(context);
ImageView coffeeImageView = new ImageView(context);
Button submitOrderButton = new Button(context);
CatView sleepyCatView = new CatView("Tiger");
}

```

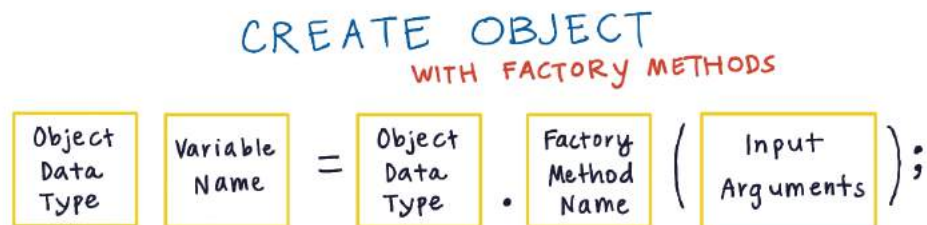
ان انشاء كائن جديد هو مشابه جدا لانشاء متغير كما تعلمنا سابقا وكما هو موجود في الصورة التالية:

Data type Variable name = Initial value ;

int quantity = 2;

نلاحظ في الصورة السابقة عند انشائنا لكائن جديد ان هناك كلمة **new** وهي ايضا من ضمن الكلمات الدلالية لجافا **Java keywords**

وفي بعض الاحيان عند انشاء كائن جديد نجد نصيحة لانشاء الكائن بحسب وظيفة مصنع لانشاء الكائن بدلا من الدالة الانشائية **constructor** والكلمة المستخدمة هنا هي **Object Data Type** اي يعني **MediaPlayer** متبوعة ب **create**. ولاننسى انه عند كتابة نوع الكائن كما هو موجد في **MediaPlayer** يجب ان تكون كلمة **Media** في الاحرف الكبيرة وليست الصغيرة اي بمعنى اخر يجب ان تكتب كما كتبت عندما كانت **Object Date Type** كما في الصورة التالية:



```

MediaPlayer player = MediaPlayer.create(context, R.raw.song);
Toast toastMessage = Toast.makeText(context, "Hi", duration);
  
```

استدعاء اساليب كائن

لأستدعاء كائن يجب علينا اتباع الطرق التالية:

- اولا كتابة اسم متغير الكائن **Object Variable Name**
- ثانيا نضيف نقطة ملحوقة بأسم الوظيفة **method name**.
- وضع الادخال اذا كان من نوع **String** او **int**

في المثال التالي سيكون الكلام اوضح بصورة اكبر:

CALL METHOD ON OBJECT



```
titleTextView.setText("News");
```

حيث نلاحظ من هذا المثال انه اسم متغير الكائن هو **titleTextview** وان اسم الطريقة سيكون **setText**.

ووضع الادخال هو سيكون من نوع **String** ويكون بين علامتي تنصيص ملحوقة بأقواس مع فاصلة منقوطة ("**News**");

وهناك مثال اخر مما سبق ايضا يمكننا الاستفادة منه ايضا:

```
titleTextView.setTextSize(18);
```

حيث ان اسم متغير الكائن هو **titleTextView** واسم الطريقة **set.TextSize**

والادخال هو رقم اي من نوع **int** وسيكون (**18**)

ومثال اخر

```
warningTextView.setTextColor(Color.RED);
```

نلاحظ في هذا المثال انه كلمة **RED** قد كتبت بأحرف كبيرة **Capital letters** لانه بحسب التنسيق الموجود في جافا يجب ان تكتب بهذا الشكل

واخير مثال

```
welcomeImageView.setImageResource(R.Drawable.cloud);
```

هنا نلاحظ الإدخال `R.Drawable.cloud` حيث أن `cloud` تعني اسم الملف أو اسم الصورة

اختبار ممارسة استدعاء اساليب كائن في الفيديو طلب من عندنا انشاء استدعاء كائن جديد بعد عمل عمل مشروع جديد لتطبيق جديد على الاندرويد ستوديو حيث قمنا بتغيير العرض وتعديل بعض الكودات في الجافا حيث تم عرض **Wow!** التي تم تعديلها بدلاً من **Hello world!** يمكنك التحقق من الكود في الاسفل.

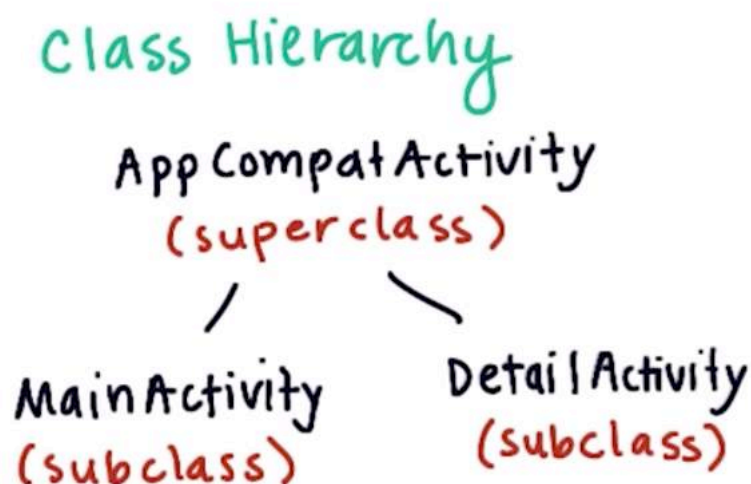
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView textView = new TextView(context: this);
    textView.setText("Wow!");

    setContentView(textView);
}
```

ويمكنك التحقق من الرابط التالي لمزيد من المعلومات

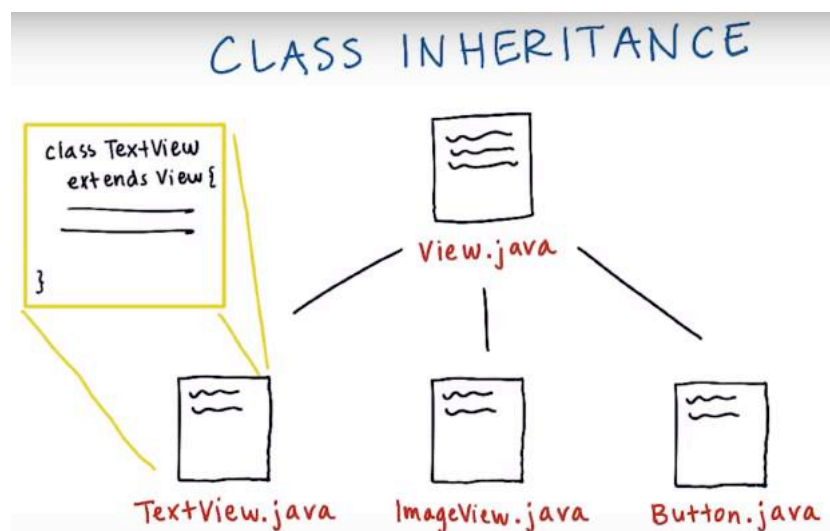
<https://developer.android.com/training/basics/firstapp/starting-activity.html>

سلوك التوريث



في هذا المخطط يظهر لدينا العلاقة بين فئات الجافا المختلفة حيث ان الفئة الاساسية او الرئيسية هي **Main Activity** وتعرف باسم **Superclass** اما **App Compat activity** و **Detail Activity** هي فئات فرعية وتعرف ب **Subclass**

ولشرح سلوك التوريث بصورة مفصلة اكثر عن طريق **الصورة التالية:**



نلاحظ في الصورة السابقة انه يمكننا استخدام سلوك التوريث او **Class Inheritance** عن طريق فئات العرض المختلفة مثل **TextView** و **ImageView** و **Button** فمثلا ان المشترك بين هذه الفئات هو العرض والطول بالنسبة الى النصوص والصور والازرار فبدلا من كتابة الطول والعرض في كل فئة او نسخها في كل فئة يمكننا استخدام **معامل التوريث بعمل ملف View.java** والاشارة الى **TextView** مثلا لتوريث الخواص الموجودة في **View.java** وذلك بكتابة الكود الموجود اعلى يسار الصورة.

مثل ما ذكرنا سابقا فان **View.java** هي **Superclass** لان هي الفئة الرئيسية اما **Textview.java** و **ImageView.java** و **Button.java** هي فئات فرعية **Subclass**

findViewById

العثور على العرض حسب رقم التعريف

في الصورة مثال على العثور على العرض حسب رقم التعريف ورقم التعريف هنا يقصد به ID حيث

```
private void displayMessage(String message) {
    TextView orderSummaryTextView = (TextView) findViewById(R.id.order_summary_text_view);
    orderSummaryTextView.setText(message);
}
```

نذكر عندما قمنا بعمل تطبيق **Just Java** لطلب القهوة قمنا بعمل id في بعض العروض النصية **TextView** وكان احد هذه ارقام التعريف **order_summary_text_view** اذا ففي الصورة السابقة كيف قمنا بالعثور على العرض حسب رقم التعريف id بكتابة **R.id.order_summary_text_view** ولكن تم وضعها بين اقواس وكانت ملحوظة ب **findViewById** والسبب من ذلك وذلك لتحديد الخاصية

او المطلوب عمله بالاضافة الى تم وضع متغير `orderSummaryTextView` وهو متغير لربط ما يتم عرضه على شكل بالمعرف `R.id.order_summary_text_view` ربما الموضوع صعب نوعا ما ولكن الفكرة هي عمل رقم تعريف في `id` في XML من خلال `TextView` محدد وبعدها يتم ربط مانود عرضه في هذا ال `TextView` وذلك عن طريقة كتابة الكود في الجافا كما تمت كتابته في الصورة أعلاه! ولتوضيح اكثر، نذكر عند عمل تطبيق `Just Java`

```
private void displayMessage(String message){
    TextView priceTextView=(TextView)findViewById(R.id.price_text_view);
    priceTextView.setText(message);
}
```

نلاحظ هنا كيف تم العرض عند طلب سعر القهوة بعد تحديد الكمية ولتوضيح ما اعنيه هو عند كتابة كود XML في الصورة التالية كيف تم انشاء معرف `id` والربط بينهم

```
<TextView
    android:id="@+id/price_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="8dp"
    android:text="$0"
    android:textColor="@android:color/black"
    android:textSize="16sp" />
```

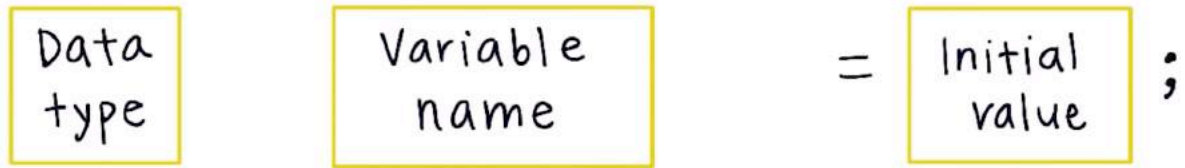
الدرس العاشر (البرمجة الموجهة للكائنات الجزء الثاني)

في هذا الدرس سنتعرف على كود جديد لمربع التحديد او صندوق التحديد `Check box`

☐ Notify me

```
<CheckBox
    android:id="@+id/whippedCream_check_box"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:text="Whipped Cream"
    android:paddingLeft="24dp"/>
```

طرق ادخال Boolean هي تشبه طرق ادخال int



```
boolean    hasWhippedCream    =    true;
```

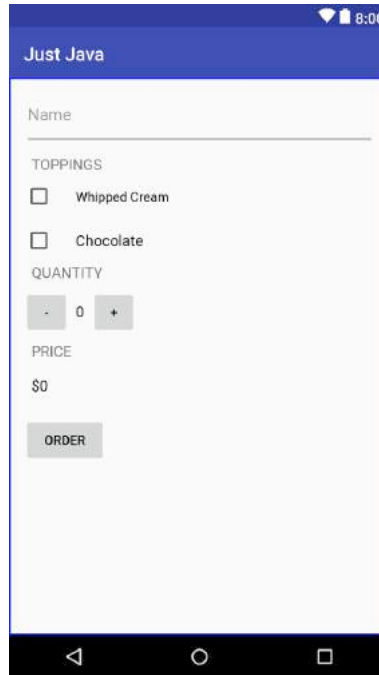
كيفية التمرير في تطبيق اندرويد؟ How to scroll in Android app
 والتمرير يقصد به هو ان تتصفح واجهة التطبيق فمثلا اذا كانت طويلة تستطيع التمرير
 بالاصبع والوصول الى الاسفل

نستخدم الكود التالي في XML ويجب ان يكون في البداية يعني قبل RelativeLayout
 ووسم الاغلاق يجب ان يكون في الاخير ايضا لايوجد بعده شيء

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
</ScrollView>
```

اختبار اضافة الاسم

في هذا الاختبار نستعرض كود جديد في XML وهو كود **EditText** لاضافة الاسم كما هو موجود في الصورة التالية:



```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/name_field"
    android:hint="Name"
    android:inputType="textCapWords"
    android:paddingBottom="24dp"/>
```

الاكواد الشرطية if/else

عوامل تشغيل التساوي والارتباط في Java

• == يساوي (لاحظوا أنها ليست علامة = مفردة لأن هذه هي عامل تشغيل

التعيين)

- != لا يساوي
- < أكبر من
- <= أكبر من أو يساوي
- > أقل من
- >= أقل من أو يساوي

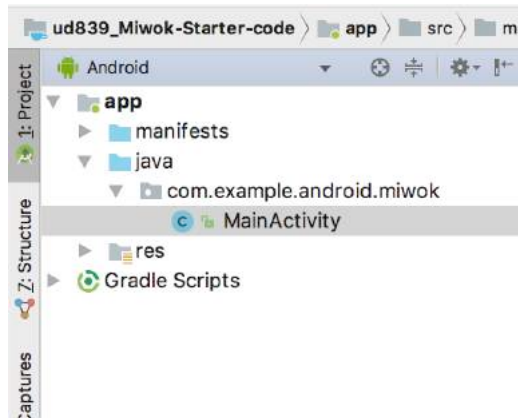
مثال:

$10 > 5$ هو تعبير يعني أن 5 أقل من 10. ويعطي هذا القيمة **true**، لأن 5 أقل من 10.
 $5 == 5$ يعطي القيمة **true**.
 $5 != 5$ يعطي القيمة **false**. وهذا لأنه من الخطأ أن نقول خمسة لا تساوي خمسة.

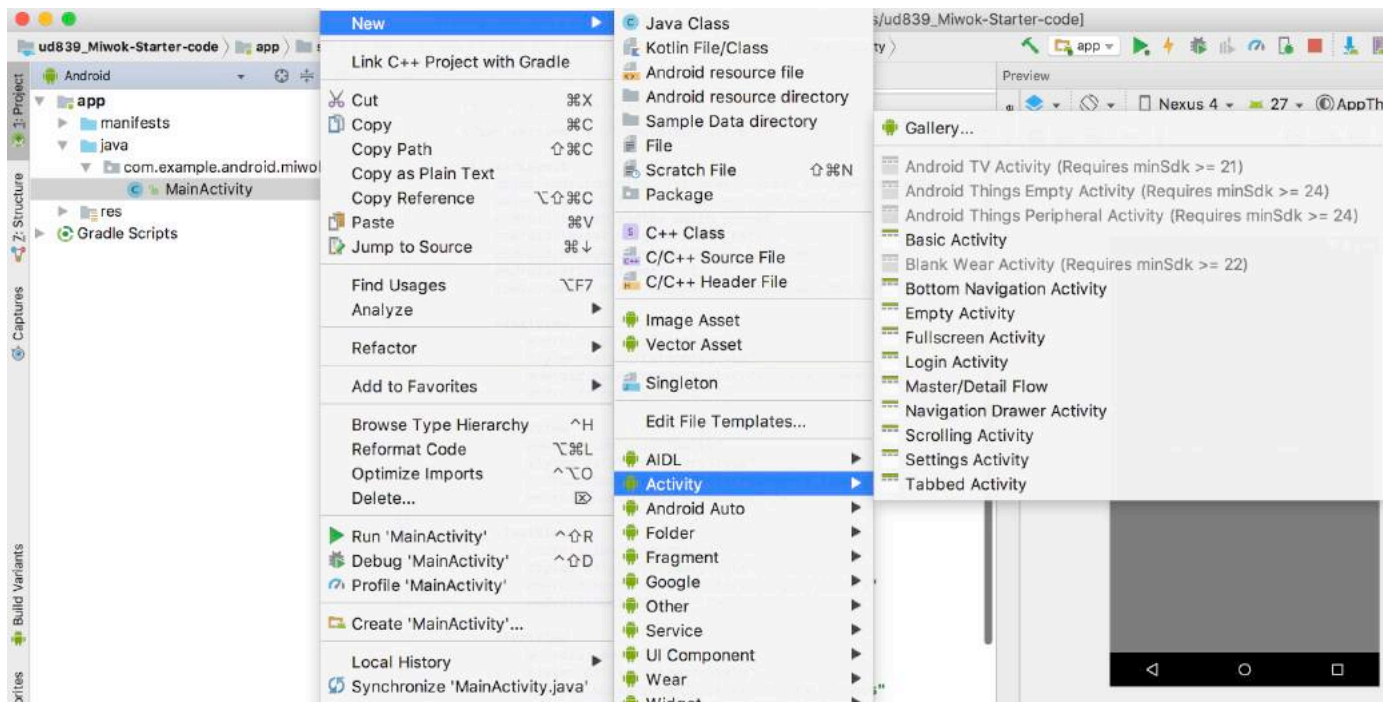
الدرس الحادي عشر

Multi screen app تطبيق متعدد الواجهات

في التطبيقات السابقة تطرقنا الى عدة تطبيقات ذات واجهة واحدة لكن سنتعلم كيفية اضافة واجهات اضافية الى التطبيق اذهب الى Project كما هو موضح في الصورة:



وبعدها الزر الايمن للماوس واختر :



وبعدها قم بتسمية النشاط وبعدها اضغط انهاء New>>Activity>>Empty Activity

ولربط هذه الواجهات معا يجب علينا اولا تعديل ملف XML بأضافة كود onClick وتسميته

```
<TextView
    android:id="@+id/numbers"
    style="@style/CategoryStyle"
    android:background="@color/category_numbers"
    android:text="Numbers"
    android:onClick="openNumbersList"/>
```



الخطوة التالية هي اضافة كود للجافا MainActivity.java

```
public void openNumbersList(View view){
    Intent i = new Intent( packageContext: this, NumbersActivity.class);
    startActivity(i);
}
```

قمنا بأضافة الكود السابق الى جافا لربط واجهة عرض الارقام NumbersActivity مع واجهة العرض الرئيسية.

OnClick Listener

وهي طريقة استدعاء فردية عند تفاعل المستخدم مع واجهة العرض والضغط عليها.

ولأضافة OnClick Listener يجب اتباع هذه الخطوات:

- **اولا تعريف مستمع الحدث event listener والسلوك المخصص** عندما يحدث الحدث هذا مثال للكود ويجب ان يتم وضعه في ملف جديد اسمه NumbersClickListener.java

```
public class NumbersClickListener implements OnClickListener {
    @Override
    public void onClick(View view) { Toast.makeText(view.getContext(),
        "Open the list of numbers", Toast.LENGTH_SHORT).show();}
}
```

- **ثانيا** قم بإنشاء نسخة كائن جديدة من مستمع الحدث بإستخدام **دالة انشائية Constructor**

2. Create a new object instance of the event listener (using the constructor)

In MainActivity.java:

```
NumbersClickListener clickListener = new NumbersClickListener();
```

- **ثالثا** ربط المستمع بالواجهة التي تريد استماع الاحداث عليها

3. Attach the listener to the view

In MainActivity.java:

```
buttonView.setOnClickListener(clickListener);
```

وهناك طريقة لاضافة المستمع وبهذه الطريقة لانحتاج الى نشاط **Activity** جديد وانما

يمكننا ادراج الكود في ملف **MainActivity.java**

مثال للكود في الاسفل (ملاحظة قمت بعرض الكود الخاص بالارقام **Numbers Activity**

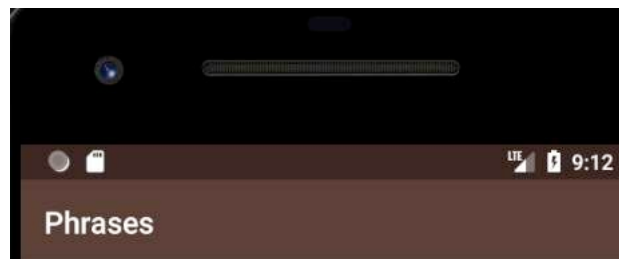
فقط ويمكنك استخدام نفس الطريقة المدرجة ادناه لتكرار بقية الانشطة)

```
// Find the View that shows the numbers category
TextView numbers = (TextView) findViewById(R.id.numbers);
// Set a click listener on that View
numbers.setOnClickListener(new View.OnClickListener() {
    // The code in this method will be executed when the numbers View is clicked on.
    @Override
    public void onClick(View view) {
        Intent numbersIntent = new Intent( packageContext: MainActivity.this, NumbersActivity.class);
        startActivity(numbersIntent);
    }
});
```

عند عرض اي نشاط من الأنشطة على التطبيق سنلاحظ ان كلمة **Miwok** تضل في الاعلى



حتى وان قمنا بالدخول مثلا على **Phrases** ستضل هذه الكلمة في الاعلى ولتغييرها بحسب كل نشطة تدخل وتكون مثل الصورة ادناه



اذهب الى ملف **AndroidManifest.xml**
قم باضافة كود **android:label="Phrases"**

```
<activity android:name=".NumbersActivity"
    android:label="Numbers"/>
<activity android:name=".FamilyMembers"
    android:label="Family Members"/>
<activity android:name=".Colors"
    android:label="Colors"/>
<activity android:name=".Phrases"
    android:label="Phrases"/>
```

الدرس الثاني عشر (الصفائف والقوائم والتكرارات الحلقية والفئات المخصصة)

انواع بيانات جافا

Primitive		
Type	Examples	Description
int	1 100 -234 2000127	This is a whole number. It is usually used as the default data type for integral values. Ranges from -2^{31} to 2^{31} .
double	. 12345678901234 5 66.66667 100000000. 0d	This can be a decimal that covers the same range as a float. It is usually used as the default data type for decimal values. No commas. If no decimal point, must end with d.
boolean	true false	A boolean can hold the result of a comparison or decision. There are only two possible values or states.
char	'a' 'b' 'C' ' ; '	This is a single character. Must surround with single quotes.
long	0L -3333335L 12345678900000 L	Is an integer but can cover a bigger range of values than int. Ranges from -2^{63} to $2^{63} - 1$. No commas or decimal point. When declaring a long, end with L.
float	1.223f -12.319f -37208.204567f	This can be a decimal. No commas. When declaring a float, end with f.

short	-31768 54321 9830	Is a whole number. Can be used to save memory as byte data type. A short is 2 times smaller than an int. Ranges from -32,768 and a maximum value of 32,767.
byte	-50 0 128	Is a whole number. Can be used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an int. Ranges from -128 to 127.
Object		
Type	Examples	Description
String	"hello "this is a note" "1:20pm" " " "Mountain View, CA"	Holds a text string. Must surround with double quotes.

وبحسب تطبيق **Miwok** سنقوم بأضافة الارقام باللغة الانكليزية من **1 الى 10** ولكن يرجى الانتباه ان عند اضافة **Array** سوف تبدأ من رقم **0 وليس رقم 1** وبما انه لدينا كلمات سنقوم بأضافة **String** **Array** والسبب من اختيارنا **String** هو من نوع **Object** وليس من نوع بيانات بدائية **Primitive** كما هو موضح في الجدول السابق والان لاضافة الكلمات نذهب **NumbersActivity.java** وهذا هو الكود الذي سنقوم بأضافته في ملف الجافا:

```
//Create an array of words

String [] words = new String[10];
words [0] = "One";
words [1] = "Two";
words [2] = "Three";
words [3] = "Four";
words [4] = "Five";
words [5] = "Six";
words [6] = "Seven";
words [7] = "Eight";
words [8] = "Nine";
words [8] = "Ten";
```

سنقوم باضافة **Log** وهو المسجل الذي يقوم بتسجيل رسائل التطبيق فعند تشغيل التطبيق يمكن التحقق من ان التطبيق يعمل بصورة جيدة! وبدون مشاكل وهو اشبه بطريقة **Debug** التي تم التطرق اليها سابقا في هذا الكورس.

```
//Create an array of words

String [] words = new String[10];
words [0] = "One";
words [1] = "Two";
words [2] = "Three";
words [3] = "Four";
words [4] = "Five";
words [5] = "Six";
words [6] = "Seven";
words [7] = "Eight";
words [8] = "Nine";
words [9] = "Ten";

Log.v( tag: "NumbersActivity", msg: "Word at index 0:" + words[0] );
Log.v( tag: "NumbersActivity", msg: "Word at index 1:" + words[1] );
Log.v( tag: "NumbersActivity", msg: "Word at index 2:" + words[2] );
Log.v( tag: "NumbersActivity", msg: "Word at index 3:" + words[3] );
Log.v( tag: "NumbersActivity", msg: "Word at index 4:" + words[4] );
Log.v( tag: "NumbersActivity", msg: "Word at index 5:" + words[5] );
Log.v( tag: "NumbersActivity", msg: "Word at index 6:" + words[6] );
Log.v( tag: "NumbersActivity", msg: "Word at index 7:" + words[7] );
Log.v( tag: "NumbersActivity", msg: "Word at index 8:" + words[8] );
Log.v( tag: "NumbersActivity", msg: "Word at index 9:" + words[9] );
```

كما لاحظتم في الاعلى اضعنا رسائل تسجيل والفائدة منها انه اننا نتأكد ان السلسلة تعمل بصورة صحيحة ويمكننا التأكد بالذهاب الى **Logcat** في منصة اندرويد وقراءة رسائل التسجيل حيث يجب ان تظهر بهذا الشكل:

```
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 0:One
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 1:Two
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 2:Three
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 3:Four
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 4:Five
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 5:Six
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 6:Seven
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 7:Eight
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 8:Nine
03-18 09:57:02.834 16548-16548/com.example.android.miwok V/NumbersActivity: Word at index 9:Ten
```

وان رسائل التسجيل هذه هي رسائل مؤقتة يمكن حذفها لاحقا حيث ان الغرض منها هي للتأكد ان التطبيق يعمل بطريقة صحيحة وان السلسلة النصية تم كتابتها بصورة صحيحة.

ArrayList

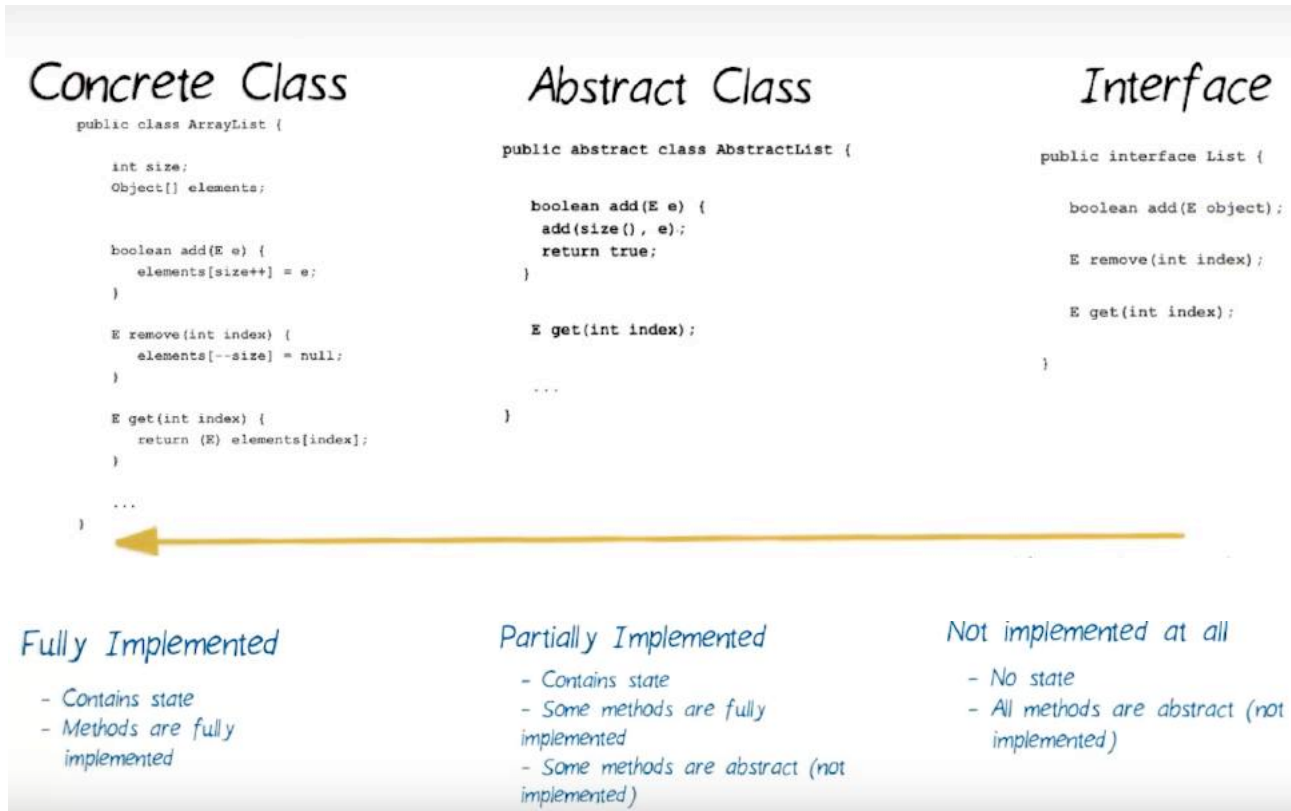
وبعدما تعرفنا على **String Array** الان سنتطرق الى **ArrayList** وسيتم استخدامه بدلا من **String Array** والسبب وراء ذلك لان **StringArray** هو محدود نوعا ما بينما **ArrayList** هو متميز بمرورته اكثر من حيث اضافة البيانات ويمكن اضافة اليه بيانات قدر ما نشاء! ولكن يرجى الانتباه الى ان **ArrayList** يتقبل بيانات من نوع **Object** فقط وليس بيانات بدائية **Primitive** ولكن لنفرض اننا نريد اضافة رقم حيث بدلا من كتابة **int** التي هي من نوع بدائي **Primitive** يمكن كتابة **Integer**

والان سنقوم بعمل مقارنة سريعة بين **Array** و **ArrayList** لتتوضح الفكرة بصورة اكثر:

	Array	Array List
Can Change size one created هل يمكن تغيير الحجم عند انشائه	No لا	Yes نعم
Is a class هو يعتبر فئة	No لا	Yes نعم
Uses Methods to access and modify elements يتم استخدام الطرق للوصول وتعديل العناصر	No لا	Yes نعم
What can it store ما نوع البيانات الذي يستطيع خزنها	Primitive and Object data type Object بيانات بدائية و	Object Only بيانات من نوع Object فقط

الآن سنقوم باستبدال String Array ب ArrayList

ولكن قبل ان نبدأ لتتعرف بشكل اكبر على **Araay** و **ArrayList** في مخطط جديد:



- **ArrayList Concrete Class** (الفئة المحددة وتحتوي على حالة والطرق يتم تنفيذها بصورة كاملة)
- **Abstract Class** الفئة المجردة (لديها بعض الطرق او الاساليب اللتي يتم تنفيذها جزئياً وتحتوي على حالة)
- **Interface** القائمة (لايوجد اي تنفيذ من هذه الطرق ولاتوجد حالة وجميع الطرق هي مجردة لا يتم تنفيذها)

والان قمنا بحذف Array String واستبدالها ب ArrayList

```
//Create an array of words
```

```
ArrayList<String> words = new ArrayList<String>();
words.add("One");
words.add("Two");
words.add("Three");
words.add("Four");
words.add("Five");
words.add("Six");
words.add("Seven");
words.add("Eight");
words.add("Nine");
words.add("Ten");
```

ويجب علينا ايضا ان نقوم بتعديل Log ايضا ليتوافق مع النوع الجديد (ArrayList)

```
Log.v( tag: "NumbersActivity", msg: "Word at index 0:" + words.get(0));
Log.v( tag: "NumbersActivity", msg: "Word at index 1:" + words.get(1));
Log.v( tag: "NumbersActivity", msg: "Word at index 2:" + words.get(2));
Log.v( tag: "NumbersActivity", msg: "Word at index 3:" + words.get(3));
Log.v( tag: "NumbersActivity", msg: "Word at index 4:" + words.get(4));
Log.v( tag: "NumbersActivity", msg: "Word at index 5:" + words.get(5));
Log.v( tag: "NumbersActivity", msg: "Word at index 6:" + words.get(6));
Log.v( tag: "NumbersActivity", msg: "Word at index 7:" + words.get(7));
Log.v( tag: "NumbersActivity", msg: "Word at index 8:" + words.get(8));
Log.v( tag: "NumbersActivity", msg: "Word at index 9:" + words.get(9));
```

سنقوم بتعديل ملف **activity_numbers.xml** ليقوم بعرض ما قمنا بخرزته وذلك :

- تحويل واجهة العرض الى **LinearLayout**
- اضافة **Orientaion** من نوع عامودي **Vertical**
- اضافة معرف جديد **id** وهو سيكون بأسم **rootView**
- لاحظ **الحشو Padding** الموجود من ضمن واجهة العرض

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingTop="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical"
    android:id="@+id/rootView"
    tools:context="com.example.android.miwok.NumbersActivity">

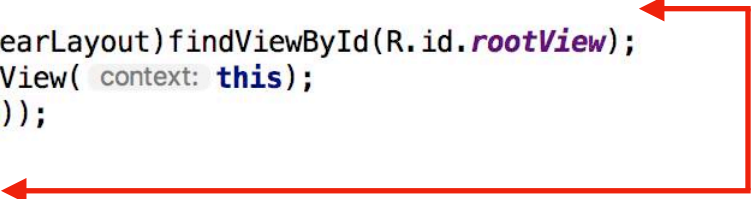
</LinearLayout>
```

بعدها نقوم نذهب الى ملف **الجافا الخاص ب Numbers** ونقوم بأضافة هذا الكود

```
//Create an array of words

ArrayList<String> words = new ArrayList<String>();
words.add("One");
words.add("Two");
words.add("Three");
words.add("Four");
words.add("Five");
words.add("Six");
words.add("Seven");
words.add("Eight");
words.add("Nine");
words.add("Ten");

LinearLayout rootView = (LinearLayout)findViewById(R.id.rootView);
TextView wordView = new TextView(context: this);
wordView.setText(words.get(0));
rootView.addView(wordView);
}
```



ويمكننا أيضا حذف كودات **Log** بعد التعديلات السابقة لأن ملفات **Log** هي فقط للتأكد من أن التطبيق يعمل حسب ما هو مخطط له.

وبعد كتابة هذه الكوادر سيظهر **One** عند الضغط على نشاط **Numbers** في التطبيق! ولكن الآن سنتطرق إلى موضوع جديد يدعى **Loops التكرارات الحلقية** حيث أن الحلقات التكرارية يوجد فيها عدة أنواع لكن الآن سنتعرف على **While Loop** وبعد إضافة **While Loop** إلى الكود الخاص بنا وفر علينا تكرار كل الكلمات وعرضها من جديد مرة ثانية.

```
ArrayList<String> words = new ArrayList<String>();
words.add("One");
words.add("Two");
words.add("Three");
words.add("Four");
words.add("Five");
words.add("Six");
words.add("Seven");
words.add("Eight");
words.add("Nine");
words.add("Ten");

LinearLayout rootView = (LinearLayout) findViewById(R.id.rootView);

int index = 0;
while (index < words.size()) {

    TextView wordView = new TextView(context: this);
    wordView.setText(words.get(index));
    rootView.addView(wordView);

    Log.v(tag: "NumbersActivity", msg: index + "Value: " + words.get(index));
    index++; //index = index + 1
}
```

ملاحظة: تم ذكر **index++** في الكود أعلاه وهي تعني **index = index + 1**

For Loop

وننتقل الان الى النوع الاخر من الحلقات التكرارية **For Loop**

ان الفرق بين **While Loop** و **For Loop** هو ان **While Loop** لفتح عدد غير محدود من الملفات مثلا هناك 20 ملف فعند استخدام **While Loop** سيتم فتحها كلها بينما **For Loop** هو لتحديد عدد الملفات التي نريد فتحها مثلا نريد فتح اول 5 ملفات فقط.

وهذا هو الشكل النهائي للكود بعد اضافة **for loop**

```
//Create an array of words

ArrayList<String> words = new ArrayList<String>();
words.add("One");
words.add("Two");
words.add("Three");
words.add("Four");
words.add("Five");
words.add("Six");
words.add("Seven");
words.add("Eight");
words.add("Nine");
words.add("Ten");

LinearLayout rootView = (LinearLayout) findViewById(R.id.rootView);

for (int index = 0; index < words.size(); index++)
{

    TextView wordView = new TextView( context: this);

    wordView.setText(words.get(index));

    rootView.addView(wordView);
}
```

View recycling

إعادة تدوير الواجهة

وتعني إعادة استخدام الواجهات اللتي لم تعد مرئية على الشاشة ولكن لهذه الطريقة مساوئها ومن الممكن ان تؤدي الى انهيار التطبيق وذلك لاستهلاك الكثير من ذاكرة الجهاز وهذا شيء لانود حصوله عند بناء تطبيقاتنا! لذلك يمكننا استخدام طريقة اخرى وهي **ListView + Array Adapter** حيث يمكننا انشاء مايكفي من الواجهات استنادا الى ما نحتاجه لما لا الشاشة.

قمنا بتعديل تطبيق miwok عن طريق اضافة **ListView** في ملف XML التابع ل **Numbers** و **ArrayList** و **Array Adapter** في ملف الجافا وفي الصور التالية التعديلات الي قمنا بها :

```
<ListView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingTop="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:id="@+id/list"
    android:orientation="vertical"
    tools:context="com.example.android.miwok.NumbersActivity">
```

```
</ListView>
```

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

    //Create an array of words

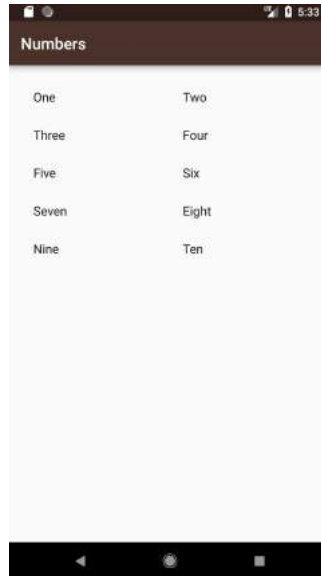
    ArrayList<String> words = new ArrayList<String>();
    words.add("One");
    words.add("Two");
    words.add("Three");
    words.add("Four");
    words.add("Five");
    words.add("Six");
    words.add("Seven");
    words.add("Eight");
    words.add("Nine");
    words.add("Ten");

    ArrayAdapter<String> itemsAdapter =
        new ArrayAdapter<String>( context: this, android.R.layout.simple_list_item_1, words);

    ListView listView = (ListView) findViewById(R.id.list);
}
```

استخدام GridView

GridView هي عبارة عن تخطيط لعرض العروض المراد عرضها على شكل اعمدة مثل الصورة التالية:



حيث قمنا بتعديل ملف XML و Java ل **NumbersActivity**

```

ArrayAdapter<String> itemsAdapter = new ArrayAdapter<String>
    ( context: this, android.R.layout.simple_list_item_1, words);

GridView listView = (GridView) findViewById(R.id.list);

listView.setAdapter(itemsAdapter);

```

```

<GridView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingTop="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:id="@+id/list"
    android:numColumns="2"
    android:orientation="vertical"
    tools:context="com.example.android.miwok.NumbersActivity">

</GridView>

```

الآن سنقوم بإنشاء ملف تخطيط جديد **Layout file** باسم **List_view.xml** واستخدام **LinearLayout** لمجموعة العرض ويجب اضافة عنصري **TextView** الى هذا الملف لتكون كالتالي:

هنا تم استخدام سطر برمجي جديد NS تعني Name Space وتم استخدام Tools لضمان عدم تعارض الاسماء اي بمعنى وضع اسم مميز لن يشاركك احد به

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/miwok_text_view"
        tools:text="Lutti"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/default_text_view"
        tools:text="One"/>
</LinearLayout>
```

انشاء فئة word مخصصة

الغرض من هذه الفئة المخصصة هو اضافة ترجمة بلغة miwok بالاضافة الى default translation
لانشاء هذه الفئة يجب انشاء ملف فئة جديد باسم words.java
بعد ذلك نقوم باضافة الكودات اليه كما هو موجود في الصورة:

```
package com.example.android.miwok;

public class Word {

    // Default language translation
    private String mDefaultTranslation;

    // Miwok Translation
    private String mMiwokTranslation;

    public Word(String defaultTranslation, String miwokTranslation) {
        mDefaultTranslation = defaultTranslation;
        mMiwokTranslation = miwokTranslation;
    }

    // get default translation
    public String getmDefaultTranslation(){
        return mDefaultTranslation;
    }

    // get miwok translation
    public String getmMiwokTranslation(){
        return mMiwokTranslation;
    }
}
```

وبعد انشاء هذا الملف واطافة الكودات اليه يجب علينا انشاء ArrayAdapter مخصص والسبب من ذلك هو لعرض الكلمة باللغة الانكليزية وتحتها بلغة miwok وبما انه Array Adapter اصلا موجود سنذهب الى NumbersActivity.java ونقوم بتعديل ArrayList وذلك بتغيير String الى word الذي هو نفس اسم فئة وورد Word.java بعد ذلك سنقوم بعدة تعديلات على الكلمات لاضافة الكلمة ومايقابلها من لغة Miwok حيث سنستخدم الكود التالي :

```
ArrayList<Word> words = new ArrayList<Word>();
words.add(new Word( defaultTranslation: "One" , miwokTranslation: "Lutti"));
```

الآن يجب علينا اضافة كلاس جديد وتسميته WordAdapter واطافة الكود التالي:

```
package com.example.android.miwok;

import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.TextView;

import java.util.ArrayList;

public class WordAdapter extends ArrayAdapter<Word> {

    public WordAdapter(Activity context, ArrayList<Word> words) {
        // Here, we initialize the ArrayAdapter's internal storage for the context and the list.
        // the second argument is used when the ArrayAdapter is populating a single TextView.
        // Because this is a custom adapter for two TextViews and an ImageView, the adapter is not
        // going to use this second argument, so it can be any value. Here, we used 0.
        super(context, 0, words);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Check if the existing view is being reused, otherwise inflate the view
        View listItemView = convertView;
        if(listItemView == null) {
            listItemView = LayoutInflater.from(getContext()).inflate(
                R.layout.list_item, parent, attachToRoot: false);
        }

        // Get the {@link AndroidFlavor} object located at this position in the list
        Word currentWord = getItem(position);

        // Find the TextView in the list_item.xml layout with the ID version_name
        TextView miwokTextView = (TextView) listItemView.findViewById(R.id.miwok_text_view);
        // Get the version name from the current AndroidFlavor object and
        // set this text on the name TextView
        miwokTextView.setText(currentWord.getMiwokTranslation());

        // Find the TextView in the list_item.xml layout with the ID version_number
        TextView defaultTextView = (TextView) listItemView.findViewById(R.id.default_text_view);
        // Get the version number from the current AndroidFlavor object and
        // set this text on the number TextView
        defaultTextView.setText(currentWord.getDefaultTranslation());

        // Return the whole list item layout (containing 2 TextViews and an ImageView)
        // so that it can be shown in the ListView
        return listItemView;
    }
}
```

وتعديل ملف NumbersActivity.java بتعديل الاكواد التالية:

```

ArrayList<Word> words = new ArrayList<Word>();

words.add(new Word( defaultTranslation: "One", miwokTranslation: "Lutti"));
words.add(new Word( defaultTranslation: "Two", miwokTranslation: "otiiko"));
words.add(new Word( defaultTranslation: "Three", miwokTranslation: "tolookosu"));
words.add(new Word( defaultTranslation: "Four", miwokTranslation: "oyyisa"));
words.add(new Word( defaultTranslation: "Five", miwokTranslation: "massokka"));
words.add(new Word( defaultTranslation: "Six", miwokTranslation: "temmokka"));
words.add(new Word( defaultTranslation: "Seven", miwokTranslation: "kenekaku"));
words.add(new Word( defaultTranslation: "Eight", miwokTranslation: "kawinta"));
words.add(new Word( defaultTranslation: "Nine", miwokTranslation: "wo'e"));
words.add(new Word( defaultTranslation: "Ten", miwokTranslation: "na'aacha"));

WordAdapter adapter = new WordAdapter( context: this, words);

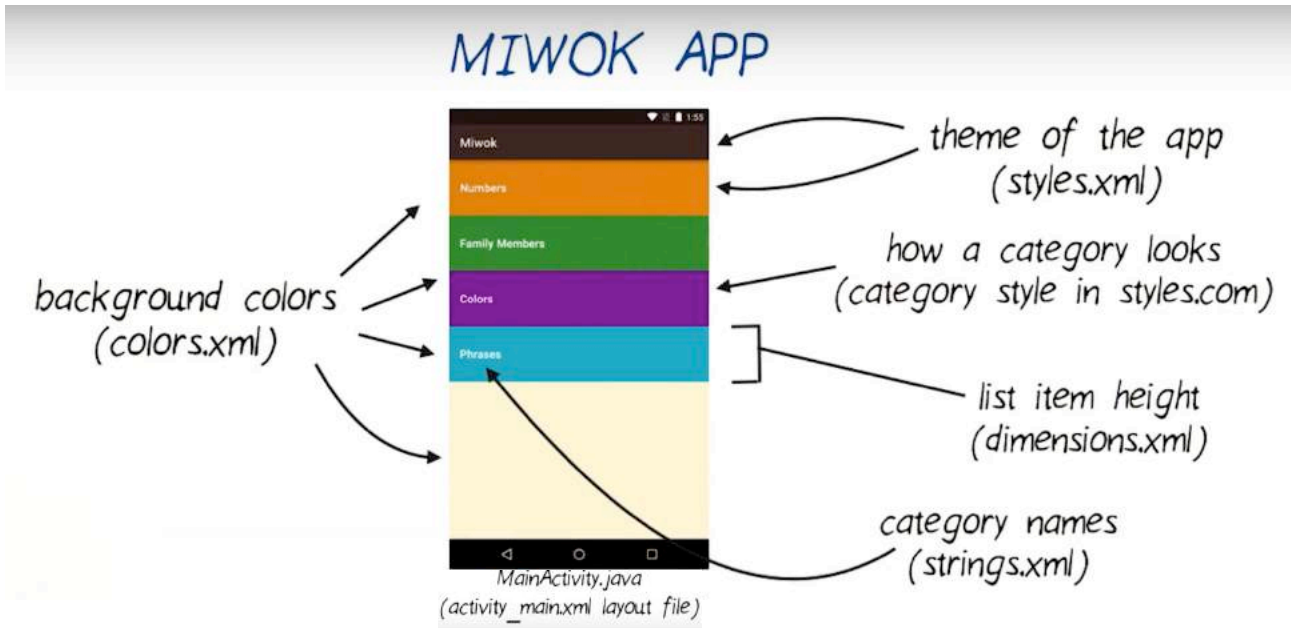
ListView listView = (ListView) findViewById(R.id.list);

listView.setAdapter(adapter);
}

```

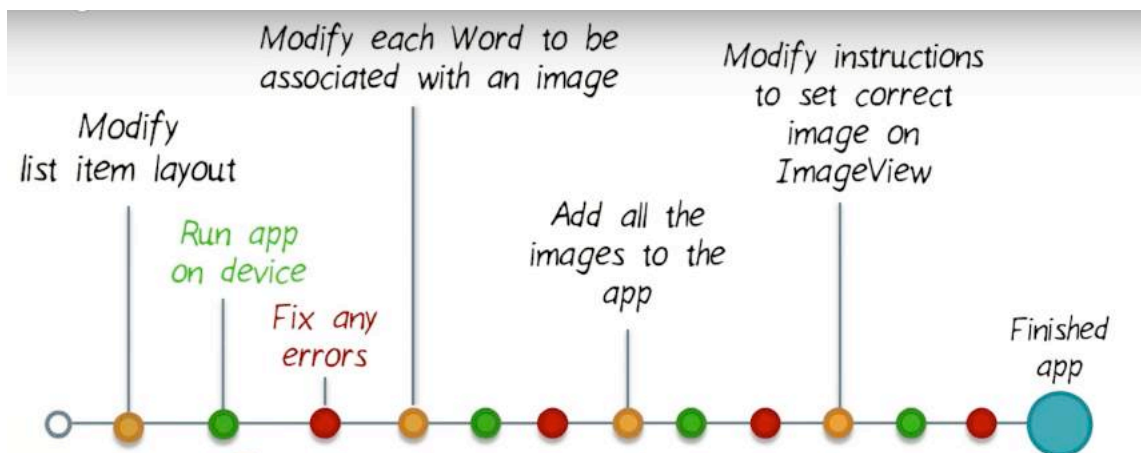
وقمنا باضافة جميع الكلمات لجميع الاقسام وتعديل ملف NumbersActivity.xml ليصبح word_list.xml ليكون ملف شامل لجميع الاقسام وليس قسم الارقام فقط وقمنا ايضا بحذف ملفات تخطيط غير مستخدمة مثل activity_colors.xml، activity_family.xml و activity_phrases.xml.

الدرس الثالث عشر (الصور واللمسات البصرية)

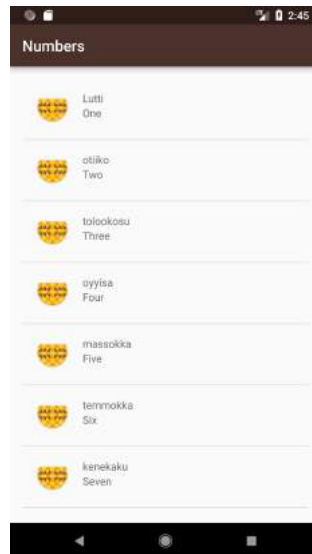


خريطة عقلية للتطبيق

نصائح عند البدء بعمل اي تطبيق يجب علينا تجربة التطبيق على المحاكى او على جهاز الموبايل وذلك لمعرفة الاخطاء ويقصد هنا بتجربة التطبيق اي عند اضافة نشاط او ملفات جديدة يجب تجربة التطبيق لاكتشاف هل توجد اخطاء؟ هذا سيسهل علينا الامر في حال قمنا باضافة ملفات وكودات ونشاطات عديدة.



إضافة صور الى التطبيق



- قمنا بإضافة صورة الى التطبيق بجانب الكلمات لتبدو بهذا الشكل حيث طبقنا عدة خطوات:
- قمنا بإضافة `ImageView` الى `list_item.xml`
- قمنا بإضافة `LinearLayout` جديد لكي يشكل مجموعة عرض متداخلة وذلك لكي يتم عرض النصوص بصورة عمودية وعرض الصورة بصورة افقية اي بجانب النص.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="16dp">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"
        android:id="@+id/image"/>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:id="@+id/text_container"
        android:paddingLeft="16dp">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/miwok_text_view"
            tools:text="Lutti"/>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/default_text_view"
            tools:text="One"/>

    </LinearLayout>
</LinearLayout>
```

ملفات الصور

يوجد هناك عدة انواع من ملفات الصور اللتي سيتم تحميلها الى التطبيق وكل مجلد من هذه المجلدات يحمل اسماً ذو دلالة على نوعية الصور الموجودة داخل المجلد.

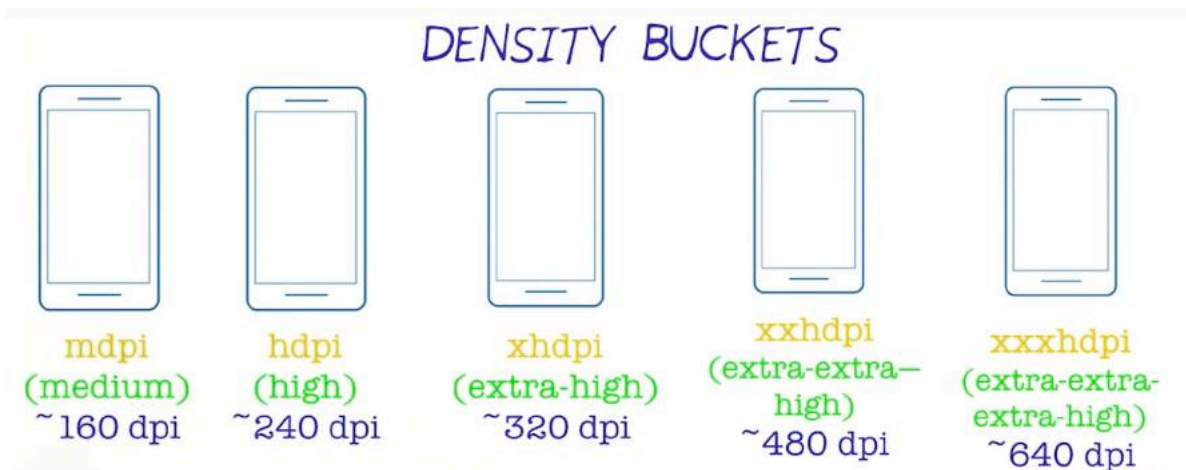
drawable-mdpi للجهاز متوسط الكثافة

drawable-hdpi للجهاز مرتفعة الكثافة

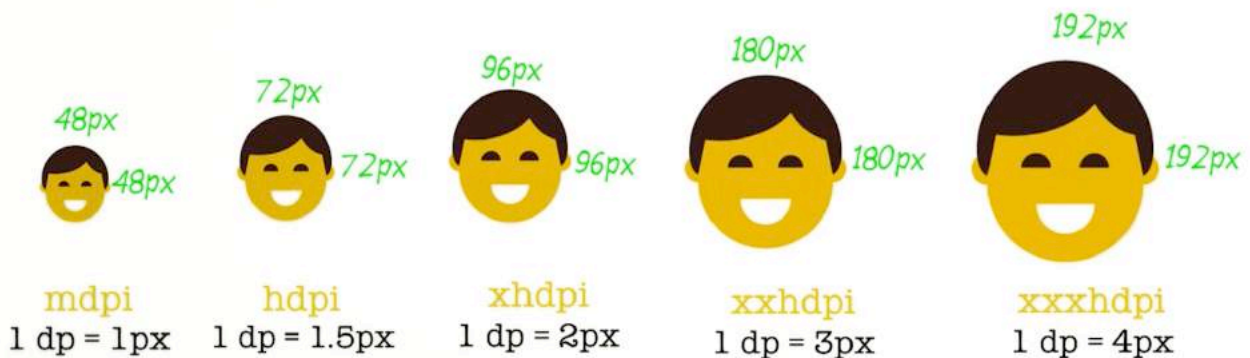
drawable-xhdpi للجهاز الاكثر ارتفاعا للكثافة

drawable-xxhdpi للجهاز الاكثر الاكثر ارتفاعا للكثافة

drawable-xxxhdpi للجهاز الاكثر الاكثر ارتفاعا للكثافة



SHOWING AN IMAGE AS 48dp x 48dp ON ANDROID



قمنا باضافة مجلدات الصور الى ملف Res من ضمن منصة Android وتعديل ملفات Numbers, Colors, Family :

```

ArrayList<Word> words = new ArrayList<Word>();

words.add(new Word( defaultTranslation: "One", miwokTranslation: "Lutti", R.drawable.number_one));
words.add(new Word( defaultTranslation: "Two", miwokTranslation: "otiiko", R.drawable.number_two));
words.add(new Word( defaultTranslation: "Three", miwokTranslation: "tolookosu", R.drawable.number_three));
words.add(new Word( defaultTranslation: "Four", miwokTranslation: "oyyisa", R.drawable.number_four));
words.add(new Word( defaultTranslation: "Five", miwokTranslation: "massokka", R.drawable.number_five));
words.add(new Word( defaultTranslation: "Six", miwokTranslation: "temmokka", R.drawable.number_six));
words.add(new Word( defaultTranslation: "Seven", miwokTranslation: "kenekaku", R.drawable.number_seven));
words.add(new Word( defaultTranslation: "Eight", miwokTranslation: "kawinta", R.drawable.number_eight));
words.add(new Word( defaultTranslation: "Nine", miwokTranslation: "wo'e", R.drawable.number_nine));
words.add(new Word( defaultTranslation: "Ten", miwokTranslation: "na'aacha", R.drawable.number_ten));

WordAdapter adapter = new WordAdapter( context: this, words);

ListView listView = (ListView) findViewById(R.id.list);

listView.setAdapter(adapter);
}

```

وتعديل ملف Word.java

```

public class Word {

    //Default language translation

    private String mDefaultTranslation;

    //Miwok language translation

    private String mMiwokTranslation;

    // Image resource ID for the word

    private int mImageResourceId;

    public Word(String defaultTranslation, String miwokTranslation, int imageResourceId) {

        mDefaultTranslation = defaultTranslation;
        mMiwokTranslation = miwokTranslation;
        mImageResourceId = imageResourceId;
    }

    //get default translation
    public String getmDefaultTranslation() { return mDefaultTranslation; }

    //get Miwok translation
    public String getmMiwokTranslation(){
        return mMiwokTranslation;
    }

    //Return the image resource ID of the word.

    public int getImageResourceId() {
        return mImageResourceId;
    }
}

```

وتعديل ملف WordAdapter.java

```

public class WordAdapter extends ArrayAdapter<Word> {

    public WordAdapter(Context context, ArrayList<Word> words) {
        super(context, resource: 0, words);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Check if an existing view is being reused, otherwise inflate the view
        View listItemView = convertView;
        if (listItemView == null) {
            listItemView = LayoutInflater.from(getContext()).inflate(
                R.layout.list_item, parent, attachToRoot: false);
        }

        Word currentWord = getItem(position);

        TextView miwokTextView = (TextView) listItemView.findViewById(R.id.miwok_text_view);
        miwokTextView.setText(currentWord.getMiwokTranslation());

        TextView defaultTextView = (TextView) listItemView.findViewById(R.id.default_text_view);
        // Get the default translation from the currentWord object and set this text on
        defaultTextView.setText(currentWord.getDefaultTranslation());

        ImageView imageView = (ImageView) listItemView.findViewById(R.id.image);
        imageView.setImageResource(currentWord.getImageResourceId());

        return listItemView;
    }
}

```

كما نلاحظ ان التغييرات التي قمنا بعملها هي فقط على Numbers, Color, Family ولم يطرأ اي تغيير على Phrases من ناحية الصور وعند تشغيل التطبيق سيظهر لدينا خطأ ويجب ان نقوم ببعض التغييرات لكي يعمل التطبيق بصورة جيدة خالية من اي اخطاء.

اولا نقوم بتعديل ملف Word.java باضافة كودين الكود الاول لتعريف قيمة ثابتة وهي:

```

private int mImageResourceId = NO_IMAGE_PROVIDED;
private static final int NO_IMAGE_PROVIDED = -1;

```

اما الاضافة الثانية هي **boolean** ليكون ناتج الكود (إما) (أو)

```
//Returns whether or not there is an image for this word

public boolean hasImage(){
return mImageResourceId != NO_IMAGE_PROVIDED;

}
```

بعدها ننتقل الى ملف **wordAdapter.java** ونقوم باضافة الكود التالي:

```
if (currentWord.hasImage()) {
    // If an image is available, display the provided image based on the resource ID
    imageView.setImageResource(currentWord.getImageResourceId());
    // Make sure the view is visible
    imageView.setVisibility(View.VISIBLE);
} else {
    // Otherwise hide the ImageView (set visibility to GONE)
    imageView.setVisibility(View.GONE);
}
```

حيث استخدمنا **if/else** للتعبير انه عند الانتقال الى **Phrases** لاحتاج الى صور وهذا كان سبب تعطل التطبيق انه في كوداتنا كنا نضع واجهتين نصيتين وصورة والان باستخدام **else/if** يمكننا التفريق انه حيث الانتقال الى **Phrases** ستظهر الواجهتين النصيتين بدون مشاكل اما عند الانتقال الى بقية الانشطة سيكون لدينا صور.

ان السبب من اختيار **GONE** بدلا من **Invisible** هو انه في حالة اختيار **invisible** سيتم ترك فراغ وهذا مالا نريده اما **GONE** سيتم ازالة الصورة وعدم ترك فراغ.

الدرس الرابع عشر (دورة حياة النشاط وتشغيل الصوت)

إضافة الملفات الصوتية الى تطبيق **miwok**
سنضيف الآن الملف الصوتي لرقم واحد **number one**
نقوم بتعديل ملف **Numbers.java** بإضافة

```
package com.example.android.miwok;

import android.media.MediaPlayer;
import android.view.View;
import android.os.Bundle;
import android.widget.AdapterView;
import android.support.v7.app.AppCompatActivity;
import android.widget.ListView;
```

إضافة

```
private MediaPlayer mMediaPlayer;
```

إضافة

```
@Override
public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
    mMediaPlayer = MediaPlayer.create(context: Numbers.this, R.raw.number_one);
    mMediaPlayer.start();
}
```

يجب اضافة جميع الملفات الصوتية التي تحميلها الى مجلد **raw** من ضمن مجلد **res** الموجود داخل التطبيق وعند اضافة هذه الملفات عند الضغط على بقية الارقام او الاحرف لا يتم نطق نفس الصوت المراد نطقه لهذا يجب علينا القيام بعدة خطوات لتحقيق هذا الامر.

التغييرات السابقة التي عملها في ملف **Numbers.java** يجب تكرارها في بقية الملفات لكن قبل التكرار هناك بعض الكودات التي يجب اضافتها الى ملف **Numbers.java**

```
final ArrayList<Word> words = new ArrayList<Word>();
words.add(new Word("one", "lutti", R.drawable.number_one, R.raw.number_one));
words.add(new Word("two", "otiiko", R.drawable.number_two, R.raw.number_two));
words.add(new Word("three", "tolookosu", R.drawable.number_three, R.raw.number_three));
words.add(new Word("four", "oyyisa", R.drawable.number_four, R.raw.number_four));
words.add(new Word("five", "massokka", R.drawable.number_five, R.raw.number_five));
words.add(new Word("six", "temmokka", R.drawable.number_six, R.raw.number_six));
words.add(new Word("seven", "kenekaku", R.drawable.number_seven, R.raw.number_seven));
words.add(new Word("eight", "kawinta", R.drawable.number_eight, R.raw.number_eight));
words.add(new Word("nine", "wo'e", R.drawable.number_nine, R.raw.number_nine));
words.add(new Word("ten", "na'aacha", R.drawable.number_ten, R.raw.number_ten));
```

واضافة

```
@Override
public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {
    // Get the {@link Word} object at the given position the user clicked on
    Word word = words.get(position);

    // Create and setup the {@link MediaPlayer} for the audio resource associated
    // with the current word
    mMediaPlayer = MediaPlayer.create(NumbersActivity.this, word.getAudioResourceId());

    // Start the audio file
    mMediaPlayer.start();
}
```

بعدها نقوم بتعديل ملف **word.java**
وتعديل ملف **WordAdapter.java**
للحصول على الكودات كاملة

https://github.com/udacity/ud839_Miwok/commit/aa0a0d83fc4a036df53e297fae12b96fcde61a8a

عذرا بسبب الانشغال الدرس الرابع عشر غير مكتمل

الخاتمة

اخواني اخواتي تم الانتهاء من هذا الملخص مع تمنياتي بالتوفيق للجميع والتأهل الى المرحلة القادمة ان شاء الله. احب ان انوه ان هذا الملخص في بعض محتوياته قد يحتمل الخطأ والصواب حيث لاتعتمد عليه 100% بصورة كاملة ومن هنا اخلي مسؤوليتي عن اي خطأ يحدث وانني شخص مبتدأ في عالم البرمجة ومنحتني هذه المبادرة فرصة وحافز للدخول في هذا المجال الرائع لذلك اذا كان هناك خطأ في ايصال معلومة او خطأ مطبعي ارجو منكم المعذرة وان هذا الملخص للمراجعة قبل التقييم النهائي (الامتحان) وكل درس كنت اقرأه كنت اقوم بتلخيص المعلومات لترسيخها وبعد اكمال هذا الملخص احببت ان شارك الملخص معكم وبالتوفيق في التقييم النهائي وتحياتي للجميع.

اخوكم **عمر**