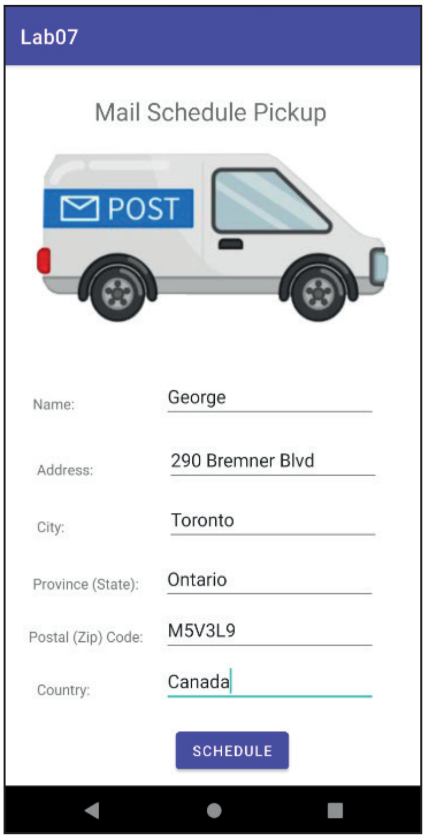
Lab 7:

Creating a Mail Schedule Pickup App

Lab Scenario:
In this lab you will practice how you may pass information from one app interface to another using the Intent class.

You will create an interface including a mail schedule pickup form. The app user should enter his/her address, then when this app user taps the Schedule button, the app user's address details will pass to another activity to check before send or edit the app user data.

.



Name:	George
Address:	290 Bremner Blvd
City:	Toronto
Province (State):	Ontario
Postal (Zip) Code:	M5V3L9
Country:	Canada

To create this app, follow the following steps:

1- Open Android Studio, and then click **File → New → New Project**

2- Select **Empty Activity**, and click **Next**

3- Type: **Lab07** for the application name, then click **Finish**

4- Before you start with typing the Kotlin code in your app, check the **build.gardle (Module:Lab07.app)** file content and be sure it has the Kotlin plugin. If not, add the following code: **id 'kotlin-android-extensions'**

And click the **Sync Now**. The configuration should be as follows:

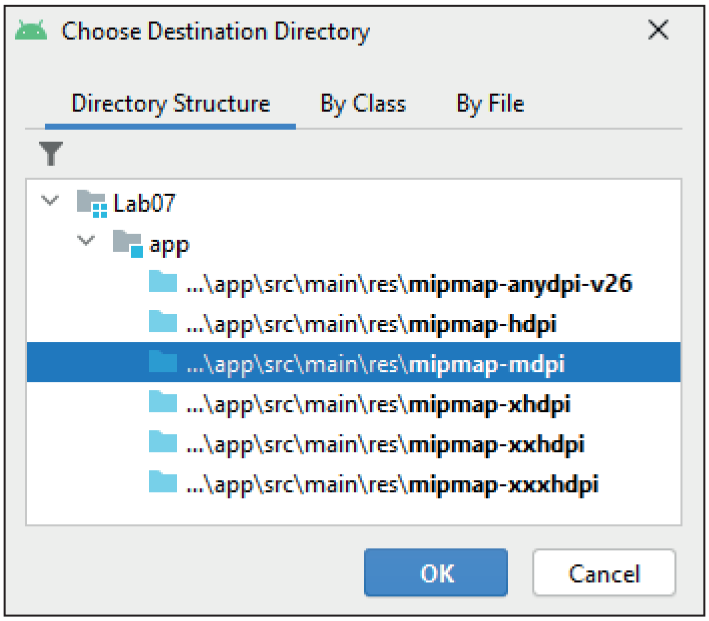
```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-android-extensions'
}
```

5- Open the **activity_main.xml** in Design mode and **Delete** the "Hello World!" **TextView**.

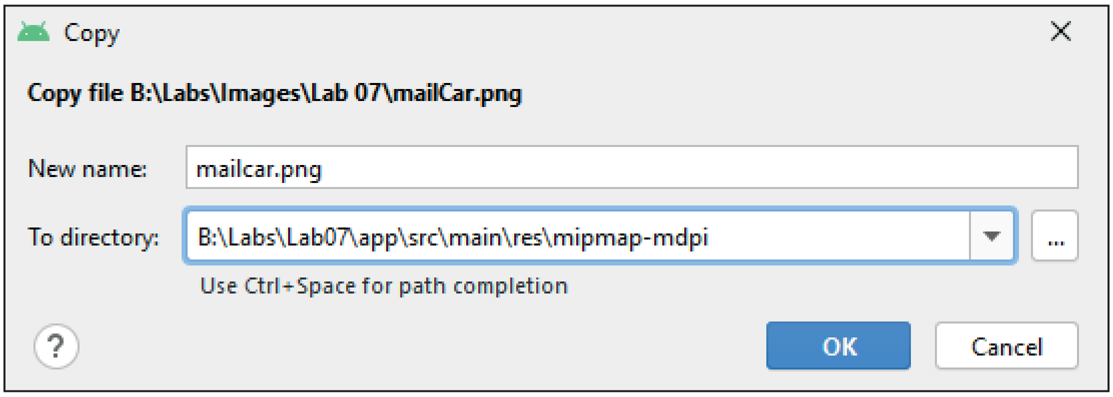
6- Open the **Labs → Images→ Lab 07** directory (this folder is a part of the Android ATC lab source files, if you don't have it, contact Android ATC support team at support@androidatc.com). Copy the image: **mailcar.png**

7- Open Android Studio, right click the directory: **mipmap (**app → res → mipmap), then select **Paste**

As illustrated in the figure below, select **mipmap-mdpi** , then click **OK**


Choose Destination Directory
Directory Structure
By Class
By File
Lab07
app
...\app\src\main\res\mipmap-anydpi-v26
...\app\src\main\res\mipmap-hdpi
...\app\src\main\res\mipmap-mdpi
...\app\src\main\res\mipmap-xhdpi
...\app\src\main\res\mipmap-xxhdpi
...\app\src\main\res\mipmap-xxxhdpi
OK
Cancel

8- The following dialog box displays where your image will be saved with your other app files. Click **OK**


Copy

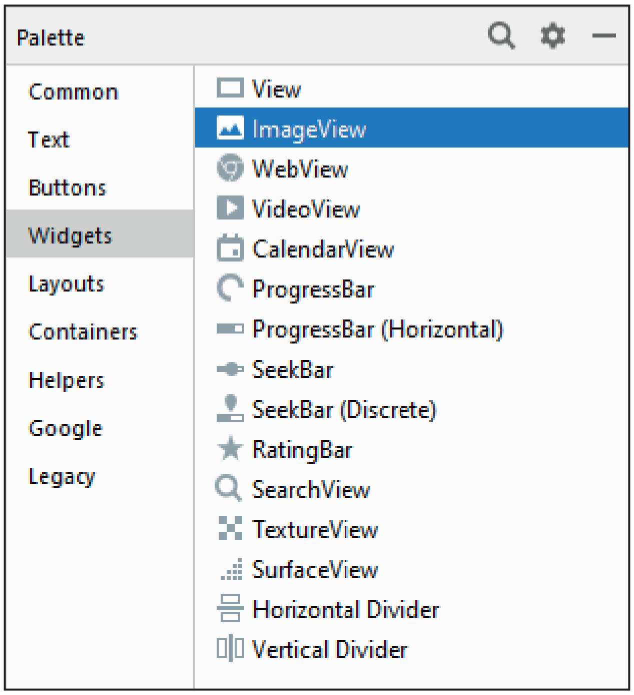
Copy file B:\Labs\Images\Lab 07\mailCar.png

New name: mailcar.png

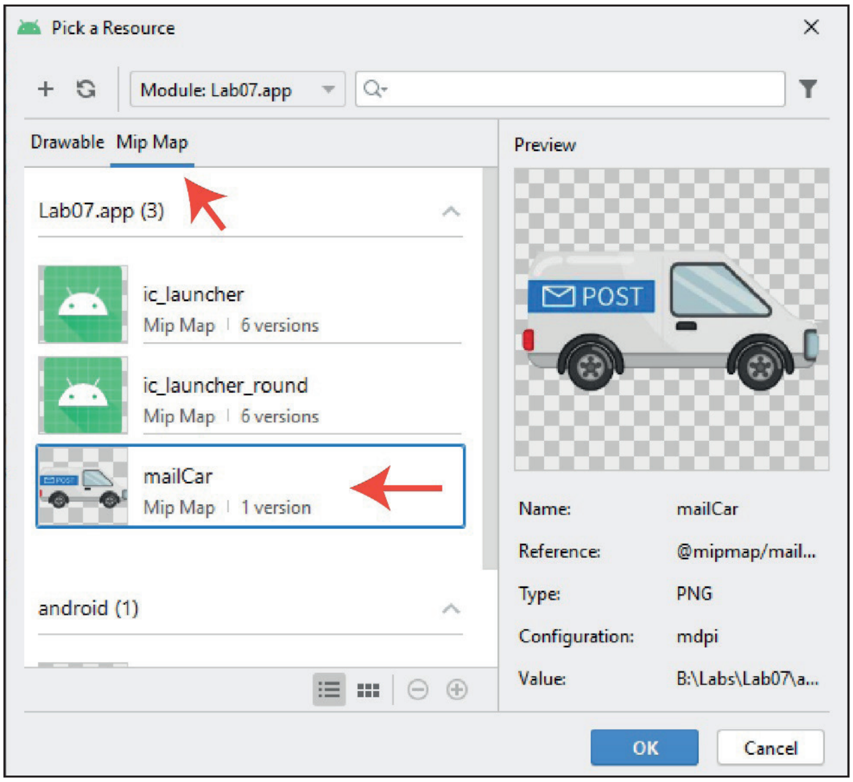
To directory: B:\Labs\Lab07\app\src\main\res\mipmap-mdpi

Use Ctrl+Space for path completion

9- Open your **activity_main.xml** in **Design** mode. From the **Palette** panel and as illustrated in the figure below, select **Widgets**, then drag and drop the **ImageView** widget to your activity interface:

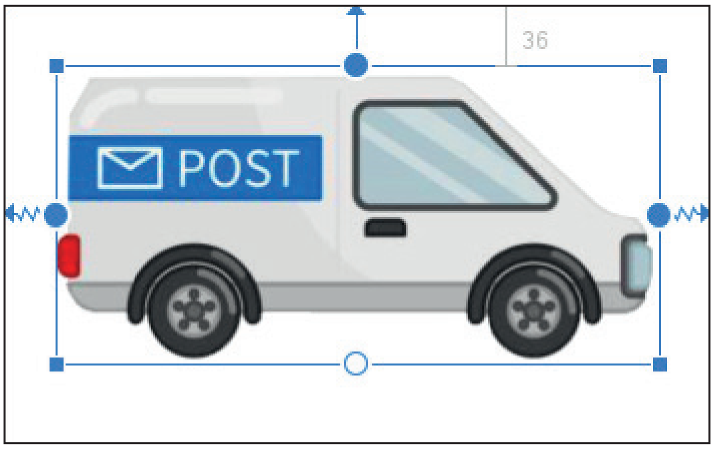


10- In the Pick a Resource dialog box and as illustrated in the following figure, click the **Mip Map** tab, select your image (mailcar.png) and click **OK**.



Name: mailCar
Reference: @mipmap/mail...
Type: PNG
Configuration: mdpi
Value: B:\Labs\Lab07\a...

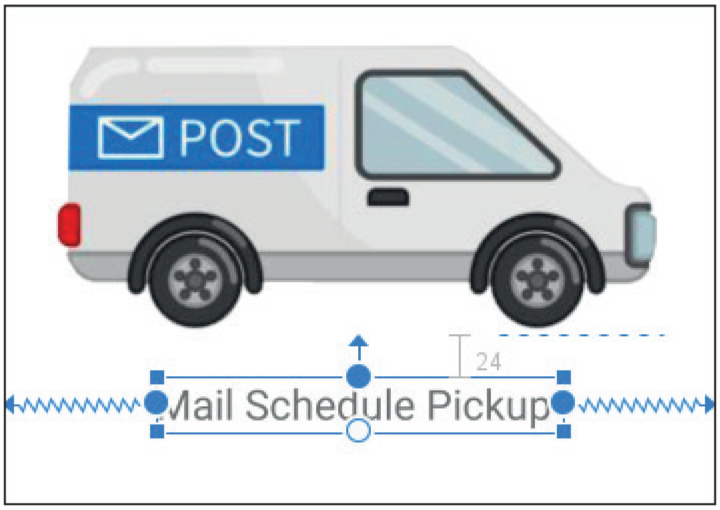
11- Set your image constraints as illustrated in the image below:



12- Add a **TextView** to your activity under your car image, and change its attributes' values as follows:

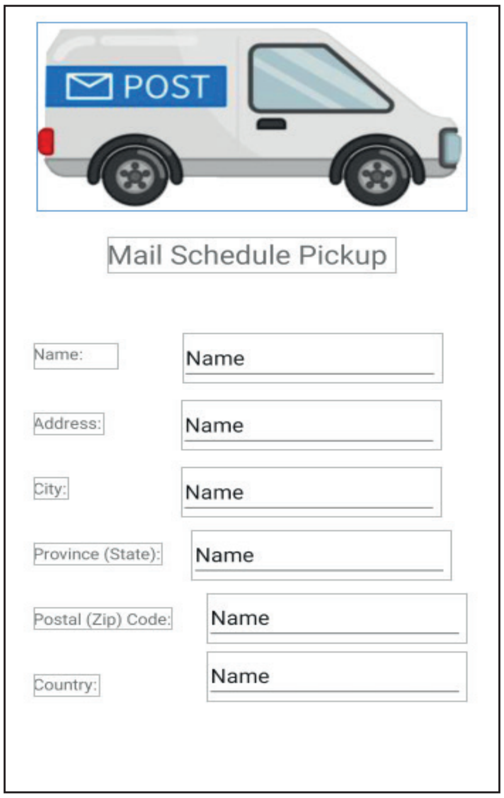
text: Mail Schedule Pickup	**textSize**: 24sp

and set its constrains as illustrated in the following figure:



13- Add the following **TextView**s and **Plain Texts** to your activity. Then, set their constraints
Name:
Address:
City:
Province (State):
Postal (Zip) Code:
Country:

As illustrated in the following figure:

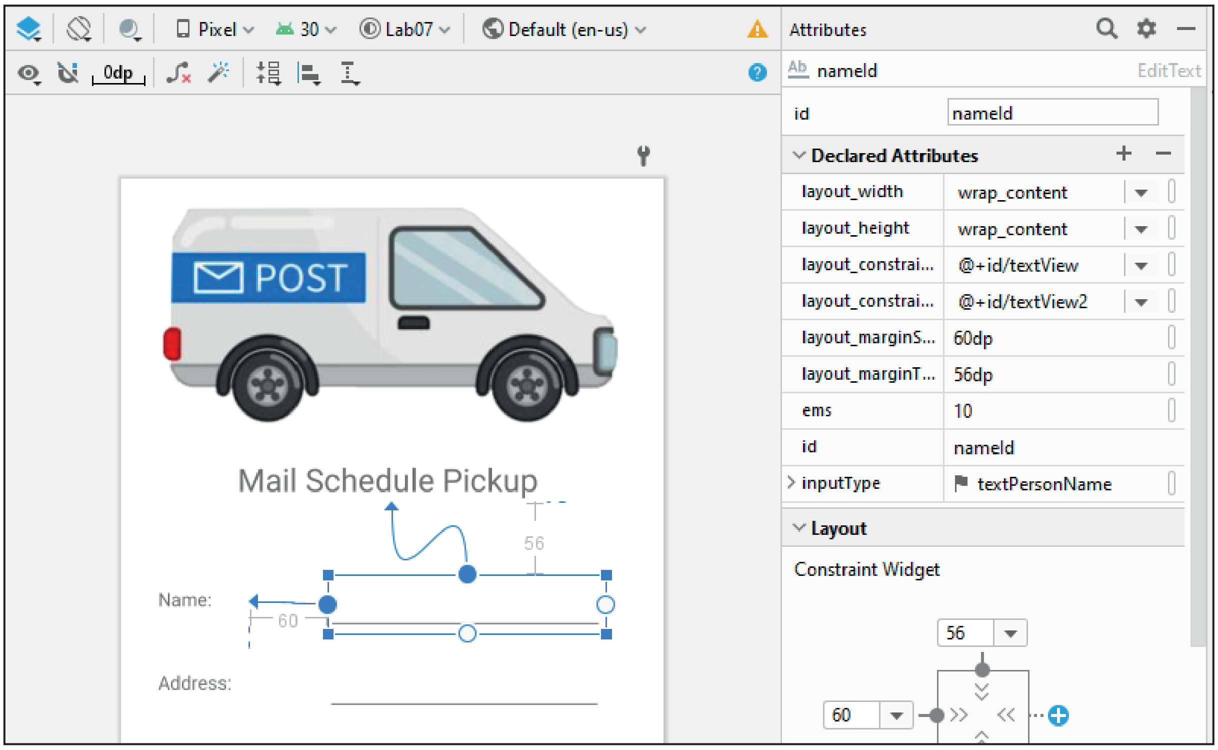


Name:	Name
Address:	Name
City:	Name
Province (State):	Name
Postal (Zip) Code:	Name
Country:	Name

14- For these six Plain Texts, delete the **text** attributes values (blank value), and set their **id** attributes values as follows:

Name:	nameId
Address :	addressId
City:	cityId
Province (State):	stateId
Postal (Zip) Code:	zipId
Country:	countryId

Note: Here, don't be confused. The table above for the **Plain Text** tags (which are on the right side of the form), not for the **TextView** tags. The following figure displays how you may change the attribute value for the **Name:** Plain Text:



Attributes

nameId EditText

id: nameId

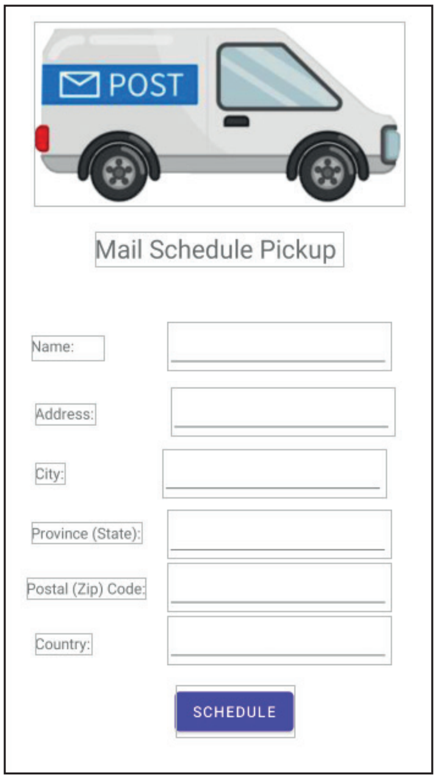
Declared Attributes

layout_width	wrap_content
layout_height	wrap_content
layout_constrai...	@+id/textView
layout_constrai...	@+id/textView2
layout_marginS...	60dp
layout_marginT...	56dp
ems	10
id	nameId
inputType	textPersonName

15- Add a button widget to your activity interface and set its constraints. Set this button attributes values as follows:

text: Schedule	**id**: scheduleBtn

As illustrated in the following figure:



The scenario of this lab is when the app user enters his/her address information, tap the **Schedule** button, all the app user's address information will be passed to another activity interface (using `putExtra()` Intent method). Now, as illustrated in the next step, you should create this new activity which will retrieve this app user's address info (Activity name: **Confirm**).

16- Right click **com.example.lab07** → **New** → **Activity** → **Empty Activity**.

17- Enter the new activity name as **Confirm**, and click **Finish**

18- Open the **MainActivity** file and write the following code which configures the "**Schedule**" button.

Create a **mySchedule** function which will include the Intent (link) to the **Confirm** activity.

The code is as follows:

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun mySchedule(view: View){}
}
```

19- Double click the **View** class which has a red color, click the pop-up red lamp, and then select **Import**.

20- Now, you can configure the **mySchedule** function to perform navigation from **MainActivity** to **Confirm** activity. Add the following code to this function:

```
var intent  =Intent(this,Confirm::class.java)
```

21-Double click the **Intent** class which has a red color, click the pop-up red lamp, and then select **Import**.

The code is as follows:

```
fun mySchedule(view: View){
    var intent  = Intent(this,Confirm::class.java)
}
```

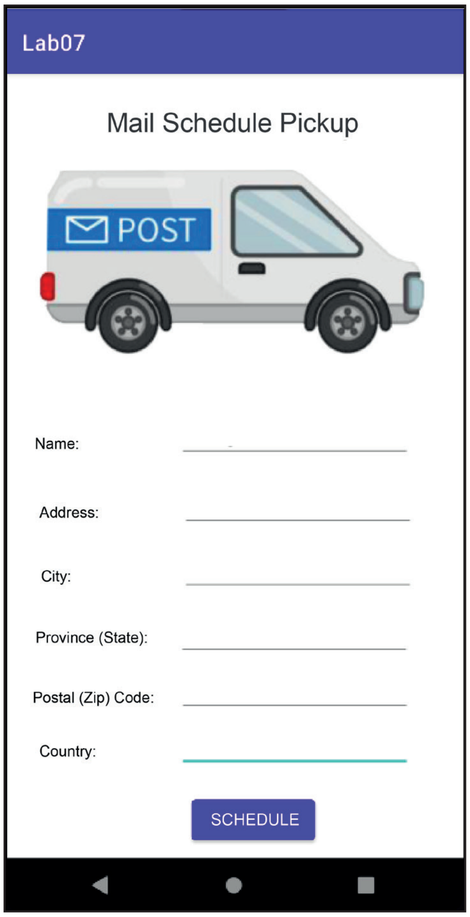
22- Now, add the **`startActivity()`** method which is responsible to launch or move to the Confirm activity using the intent information. The full code will be as follows:

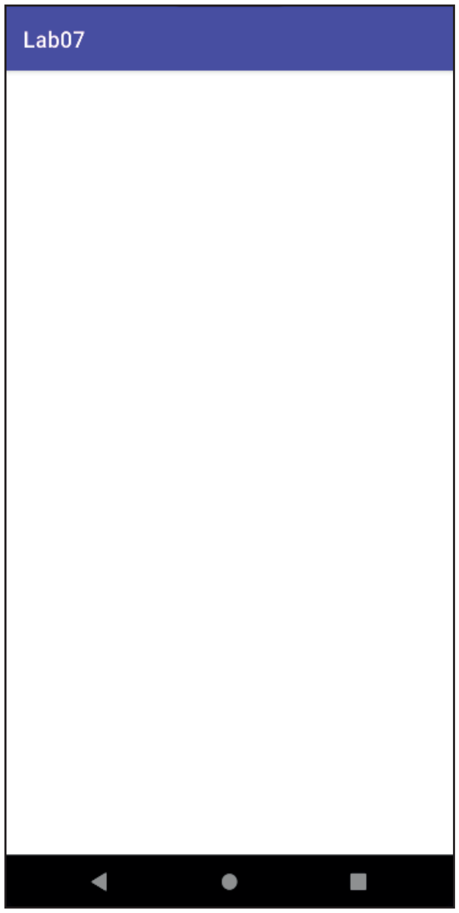
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun mySchedule(view: View){
        var intent  = Intent(this,Confirm::class.java)
        startActivity(intent)
    }
}
```

23- To link this **`mySchedule`** function operations with the: **Schedule** button, open the **activity_main.xml** file in the **Code** mode, and then add the "**onClick**" attribute to the **Button** tag as illustrated in the gray highlighted color in the following XML code:it

```
<Button
    android:id="@+id/scheduleBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="164dp"
    android:layout_marginTop="20dp"
    android:text="Schedule"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/countryId"
    android:onClick="mySchedule"
    />
```

24-To be sure everything is configured correctly until this step, click the **Run** button to test the **Schedule** button. When you tap this schedule button, you should open the **Confirm** activity as illustrated in the figure below:





25- Open the **MainActivity** file, and then add the: **`intent.putExtra`** method to the **`mySchedule`** function as illustrated in the code below:
`putExtra` method in this lab is responsible for passing the data from this main app activity form (Schedule form) to the **Confirm** activity. This data includes the **Plain Text** widgets whose **id** are: **`nameId`**, **`addressId`**, **`cityId`**, **`stateId`**, **`zipId`**, and **`countryId`**

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun mySchedule(view: View){
        var intent  = Intent(this,Confirm::class.java)
        intent.putExtra("name", nameId.text.toString())
        intent.putExtra("address", addressId.text.toString())
        intent.putExtra("city", cityId.text.toString())
        intent.putExtra("state", stateId.text.toString())
        intent.putExtra("zipCode", zipId.text.toString())
        intent.putExtra("country", countryId.text.toString())
        startActivity(intent)
    }
}
```

26- In the previous code, double click the: **nameId** which has a red color, click the red pop-up lamp, and click **Import**.

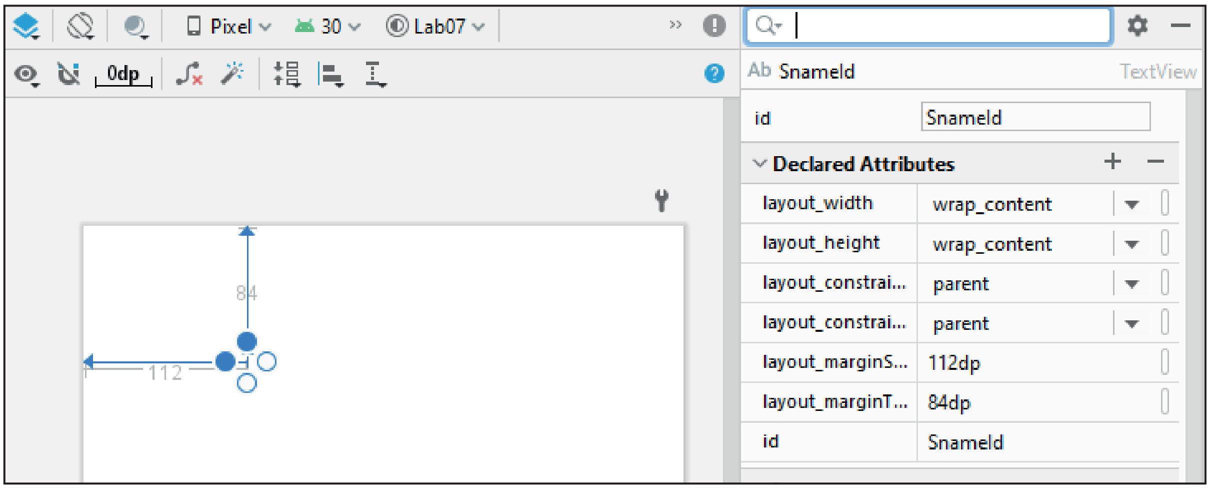
27- Now, you should configure the **getExtra** method in the **Confirm** activity (second activity) to retrieve the data which has been entered in the **putExtra** methods. You want to display the data in the **Confirm** activity in **TextView** widgets which have the following IDs: **`SnameId`**, **`SaddressId`**, **`ScityId`**, **`SstateId`**, **`SzipId`**, and **`ScountryId`**

Your address should be displayed at the Confirm activity in the following format (Standard address format):

Name
Address
City, State, Zip Code
Country

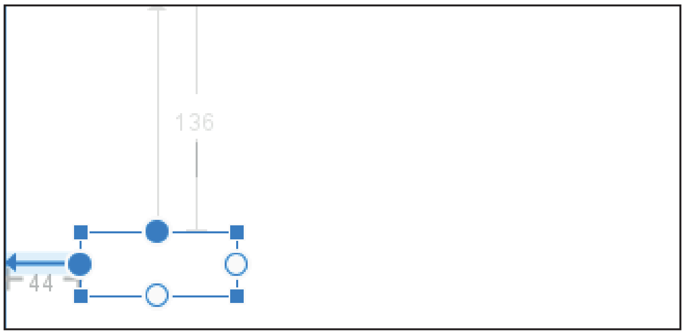
To do this, you will open the **activity_confirm.xml** file in the **Design** mode, then add **TextView** widget for each of the address parts. Here, you will add five **TextViews** and arrange them to have the same address format above.

28- Now, open **activity_confirm.xml** file in the **Design** mode. Add a **TextView** to this **Confirm** activity, set its constraints, delete its **text** tag value, and set its **id: `SnameId`** as illustrated in the following figure:

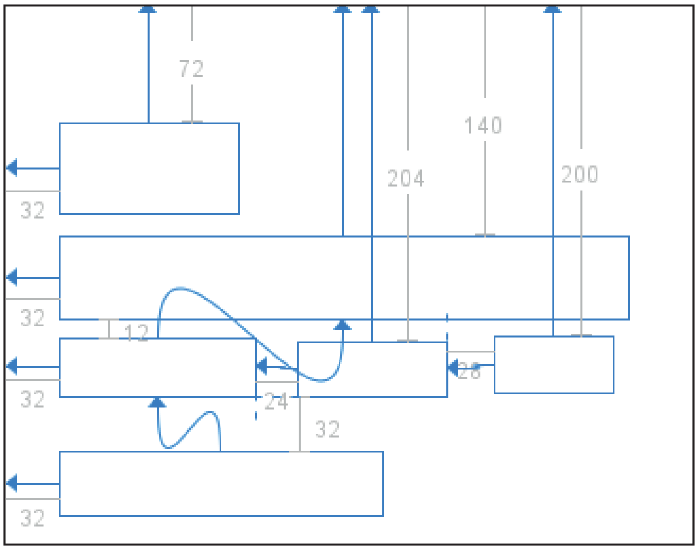


id	SnameId
layout_width	wrap_content
layout_height	wrap_content
layout_constrai...	parent
layout_constrai...	parent
layout_marginS...	112dp
layout_marginT...	84dp
id	SnameId

Try to expand the size of the **Name** to have enough space to appear on this activity as illustrated in the following figure:



29- Repeat the same step as before to add a **TextView** to the following IDs: **`SaddressId`**, S**`cityId`**, **`SstateId`**, **`SzipId`**, and **`ScountryId.`** Try to arrange them in the same previous standard address format. You should have an interface similar to the following figure:

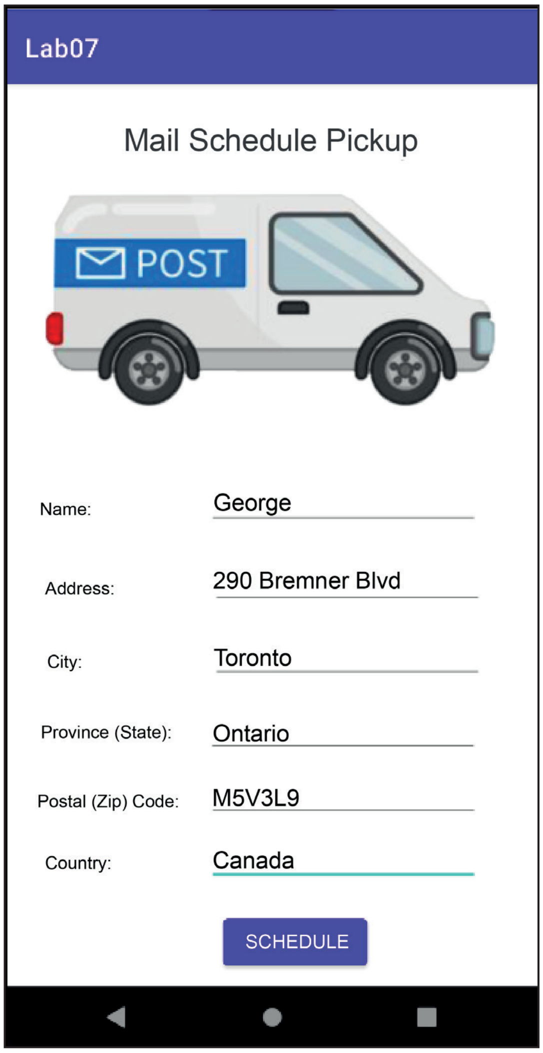


30- Now, you should configure the **getExtra** method in the second activity (Confirm activity) to retrieve the data which has been entered in the **putExtra** method. You want to display the data in the Confirm activity in the **TextView** widgets which have the IDs: **`SnamdId`**, **`SaddressId`**, **`ScityId`**, **`SstateId`**, **`SzipId`**, and **`ScountryId.`**

Open the **Confirm** file and add the following code:

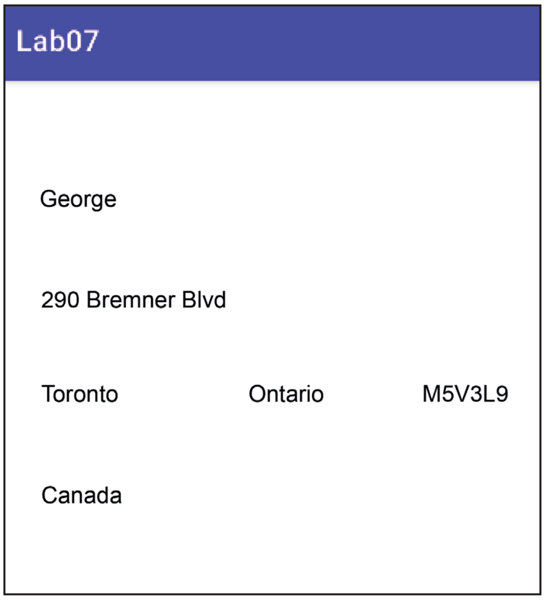
```
class Confirm : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_confirm)
        SnameId.text = intent.getStringExtra("name")
        SaddressId.text = intent.getStringExtra("address")
        ScityId.text = intent.getStringExtra("city")
        SstateId.text = intent.getStringExtra("state")
        SzipId.text = intent.getStringExtra("zipCode")
        ScountryId.text = intent.getStringExtra("country")

    }
}
```

30- To test your app, **Stop**, then **Run** your app. Fill out your mail schedule pickup form, and then tap the **Schedule** button.


Lab07

Mail Schedule Pickup

Name:	George
Address:	290 Bremner Blvd
City:	Toronto
Province (State):	Ontario
Postal (Zip) Code:	M5V3L9
Country:	Canada


Lab07

George

290 Bremner Blvd

Toronto Ontario M5V3L9

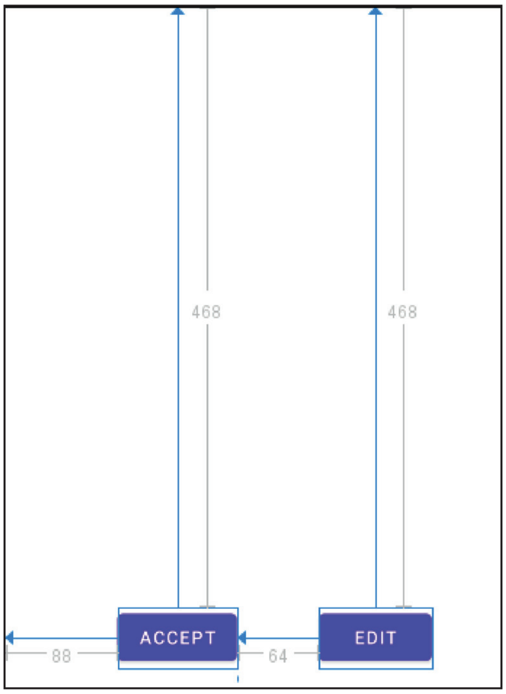
Canada

To get a better output result, you may open the **activity_confirm.xml** file and decrease the size of the **TextView** boxes of this address, and add commas (**TextView** widget with a text attribute value: **,**) between the city, state and the zip code. This comma helps in arranging the app user's address in the following format:

Name
Address
City, State, Zip Code
Country

31- Now, you should add two buttons (**Accept** and **Edit**) to the Confirm activity. If the app user taps the Accept button, he/she will open another new activity (Thank You activity) including a thank you message from the mail shipping company; however, if the app user taps the **EDIT** button, he/she will go back to the first activity to edit his/her address information.

To do this, open the **activity_confirm.xml** activity in **Design** mode, add two buttons, set their constraints, set their attribute **text** values **Accept** and **Edit**. As illustrated in the figure below:

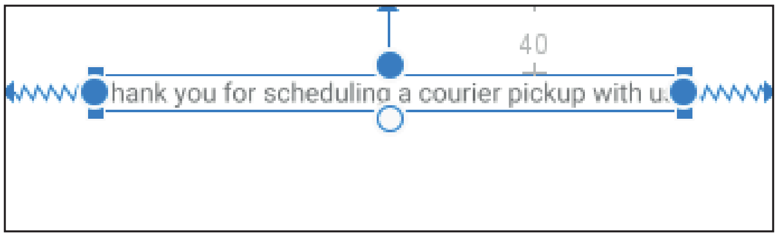


32- To create the Thank You activity, right click **com.example.lab07** → **New** → **Activity** → **Empty Activity**.

33- Enter the new activity name as **ThankYou**, and click **Finish**

34- Open the **activity_thank_you.xml** file in the **Design** mode, add a **TextView** widget, set its constraints, change the **text** attribute value to: Thank you for scheduling a courier pickup with us.

As illustrated in the following figure:



35- Now, open the **Confirm** file. Add two functions **Accept** and **Edit**. When function **Accept** runs, your app will open the **activity_thank_you.xml** activity, and your app user will get the previous thank you message.
However, if the **Edit** function runs, the app will go back to open the **activity_main.xml** file to edit the app user address.

To move from the **Confirm** activity to the **ThankYou** activity you will use the **Intent** class with the **Accept** function configurations as illustrated in the code below. However, if you used the **Intent** class with the **Edit** function, your app user will go back to the main activity (previous activity), but he/she will find his/her address form information has been cleared (blank form). Therefore, you will just use with your **Edit** function the `onBackPressed()` method. This method has the same job like the Back button in your phone. The code is as follows:

```
class Confirm : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_confirm)
        SnameId.text = intent.getStringExtra("name")
        SaddressId.text = intent.getStringExtra("address")
        ScityId.text = intent.getStringExtra("city")
        SstateId.text = intent.getStringExtra("state")
        SzipId.text = intent.getStringExtra("zipCode")
        ScountryId.text = intent.getStringExtra("country")
    }
    fun Accept(view:View){
        var agree  = Intent(this,ThankYou::class.java)
        startActivity(agree)
    }

    fun Edit(view:View){
        onBackPressed()
    }
}
```

36- Open the **activity_confirm.xml** in the **Code** mode, add the **onClick** attribute to the two **Button** tags (Accept and Edit) as illustrated in the gray highlighted color in the following XML code:

```
<Button
    android:id="@+id/acceptId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="88dp"
    android:layout_marginTop="468dp"
    android:text="Accept"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="Accept"/>

<Button
    android:id="@+id/editId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="64dp"
    android:layout_marginTop="468dp"
    android:text="Edit"
    app:layout_constraintStart_toEndOf="@+id/acceptId"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="Edit"/>
```

37- **Stop**, then **Run** your app. Fill your address information, tap **Schedule** button, tap **Edit** button to edit your address information, tap **Schedule** button again, and finally tap **Accept** button to get the thank you message.