LECTURE 15: UI PROGRAMMING PART 2

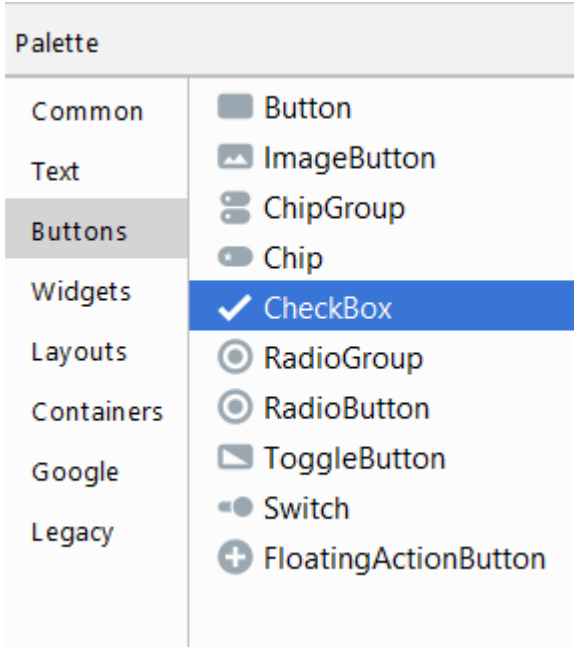
BY LINA HAMMAD & AHMAD BARGHASH

This Lecture, which will serve as your introduction to UI programming. After this class you will be familiar with: Check Boxes, Radio Buttons, Scroll View, and Spinner.

BUTTONS -- CHECKBOXES

- A checkbox is a specific type of two-states button with a check mark graphic and description text that has two states: checked or unchecked.
- Checkbox options allow to user select multiple items, so checkbox needs to manage a separately.
- key attributes:

android:checked="**bool**"	set to true to make it initially checked
android:clickable="***bool***"	set to false to disable the checkbox
android:id="@+id/***theID***"	unique ID for use in Kotlin code
android:onClick="***function***"	function to call in activity when clicked (must be public, void, and take a View arg)
ndroid:text="***text***"	text to put next to the checkbox



BUTTONS – CHECKBOXES WAY I

- XML code:

```
<CheckBox
    android:id="@+id/cb_single"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- Kotlin code:

```
val checkBox1 = findViewById<CheckBox>(R.id.cb_single)
bt1.setOnClickListener(View.OnClickListener {
      if (checkBox1.isChecked) {
          Toast.makeText(this,"Checked",Toast.LENGTH_SHORT).show()
      } else {
          Toast.makeText(this,"Unchecked",Toast.LENGTH_SHORT).show()
      }
   })
}
```

Remember to add the Koltin code inside the onCreate function.

BUTTONS – CHECKBOXES WAY2

- XML code:

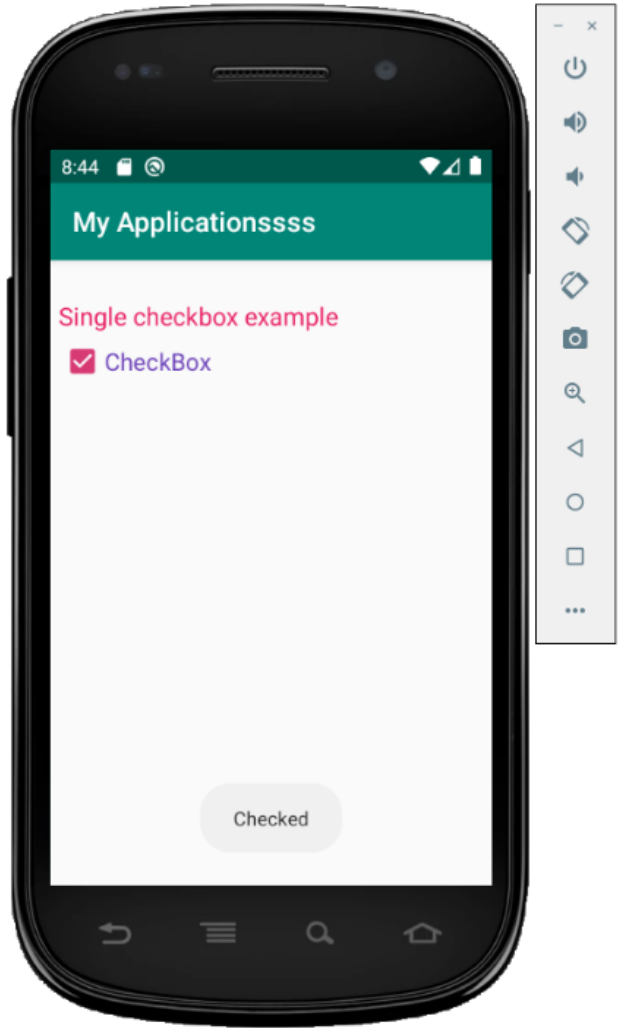
```
<CheckBox
    android:id="@+id/cb_single"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="checkBoxFun"
    android:text="CheckBox"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- Kotlin code:

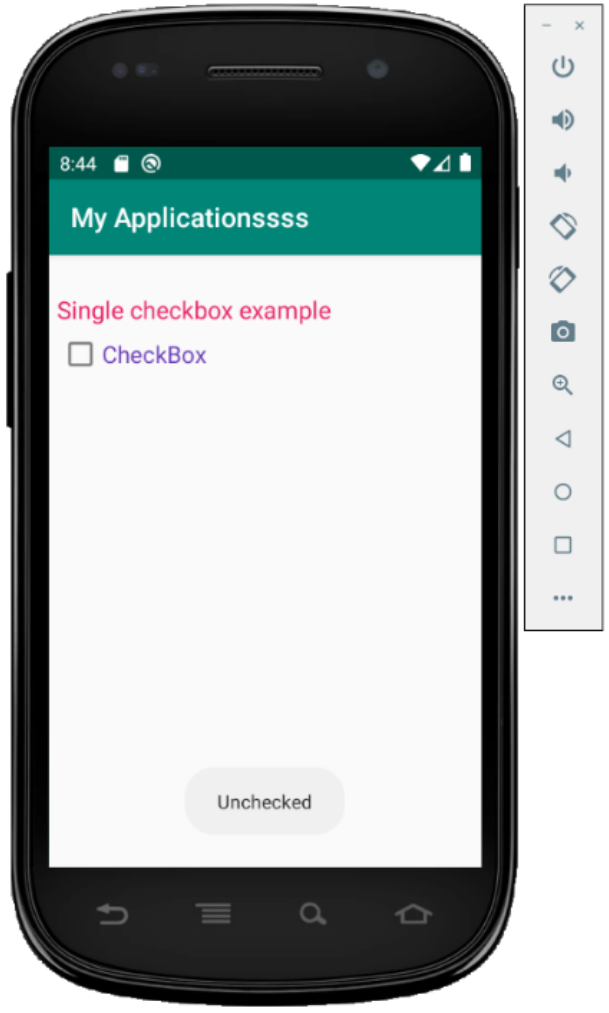
```
fun checkBoxFun(view: View) {
    if (checkBox1.isChecked) {
        Toast.makeText(this, "Checked", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "Unchecked", Toast.LENGTH_SHORT).show()
    }
}
```

BUTTONS – CHECKBOXES

When check the checkbox

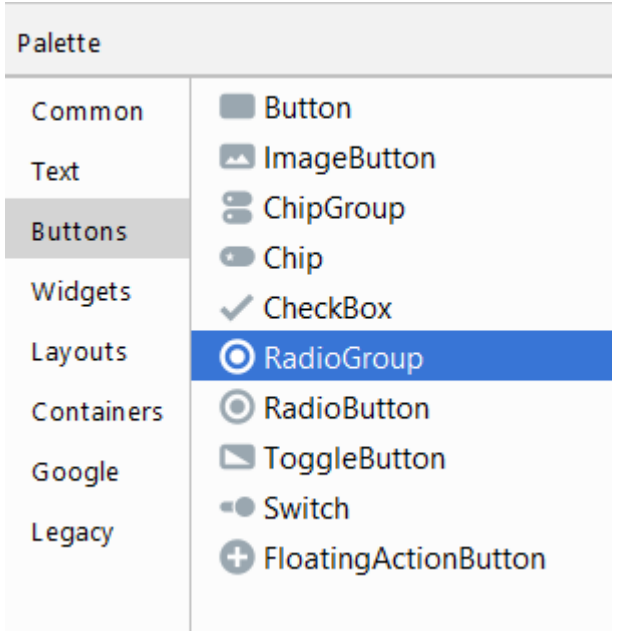


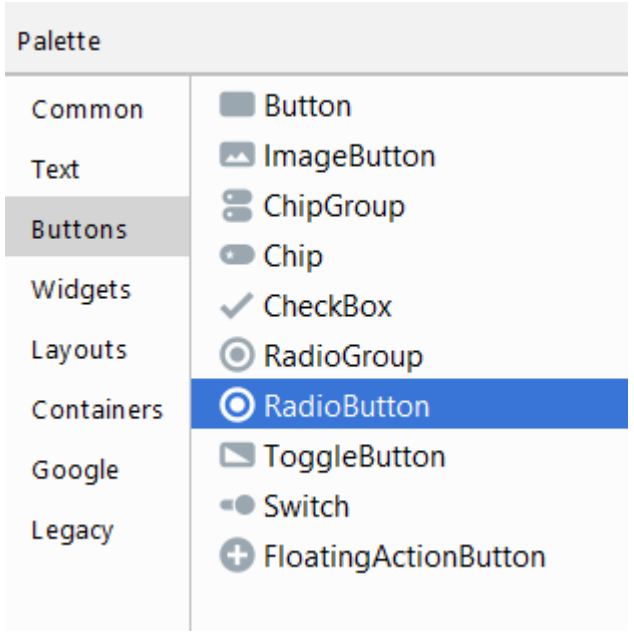
When uncheck the checkbox



BUTTONS -- RADIO BUTTONS

- The RadioButton has two states: either checked or unchecked. A RadioGroup is used to group one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.
- To use the RadioButton you will need at first to drag and drop RadioGroup, then to drag and drop RadioButton, and make sure that the RadioButton will be at iside the RadioGroup.





BUTTONS -- RADIO BUTTONS

- Key attributes:

android:checked="***bool***"	set to true to make it initially checked
android:clickable="***bool***"	set to false to disable the button
android:id="@+id/***theID***"	unique ID for use in Kotlin code
android:onClick="***function***"	function to call in activity when clicked (must be public, void, and take a View arg)
android:text="***text***"	text to put next to the button

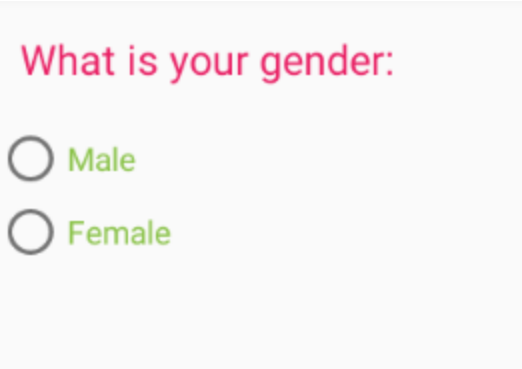
BUTTONS -- RADIO BUTTONS

- The Xml code to create the following view:

```
<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="165dp"
    android:layout_height="64dp"
    android:layout_marginTop="15dp"
    android:layout_marginLeft="15dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2">

    <RadioButton
        android:id="@+id/radio1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Male"
        android:textColor="#8BC34A" />

    <RadioButton
        android:id="@+id/radio2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Female"
        android:textColor="#8BC34A" />
```



Note: in this example there is no option automatically selected. If you want to select on the RadioGroup option, make sure to add the `android:checked="true"` attribute.

BUTTONS -- RADIO BUTTONS

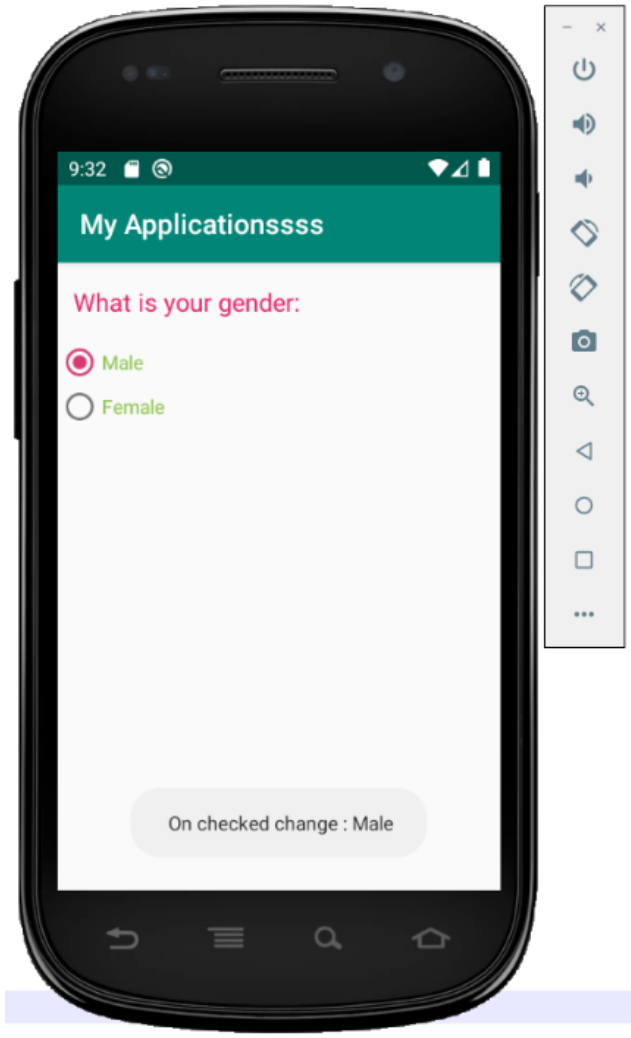
- Kotlin code:

```
// Get radio group selected item using on checked change listener
 val radioGroup1 = findViewById<RadioButton>(R.id.radioGroup1)
 radioGroup1.setOnCheckedChangeListener(
    RadioGroup.OnCheckedChangeListener { group, checkedId ->
        val radio: RadioButton = findViewById(checkedId)
        Toast.makeText(applicationContext, " On checked change : ${radio.text}",Toast.LENGTH_SHORT).show()
    })
```

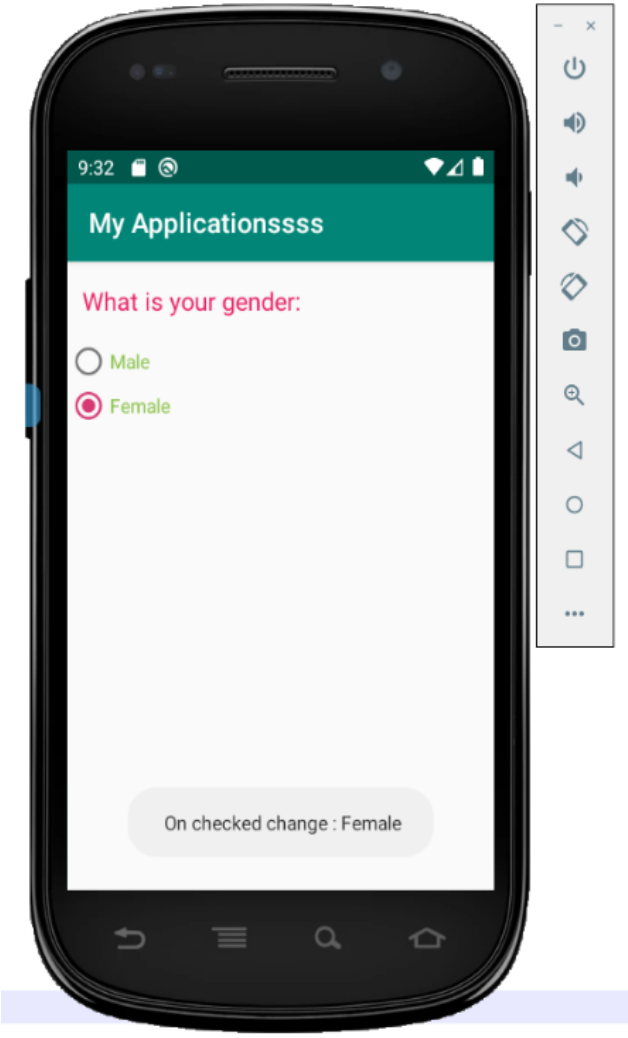
Remember to add the Koltin code inside the onCreate function.

BUTTONS -- RADIO BUTTONS

When choose the first option



When choose the second option



CONTAINERS – VERTICAL SCROLL VIEW

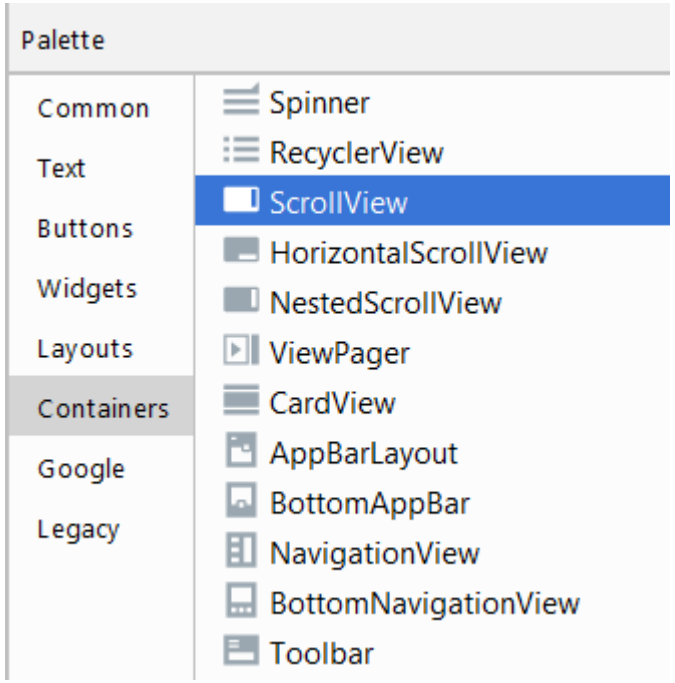
- A ScrollView is a special type of FrameLayout in that it enables users to scroll through a list of views that occupy more space than the physical display. The ScrollView can contain only one child view or ViewGroup, which normally is a LinearLayout.

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

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

</LinearLayout>
</ScrollView>
```

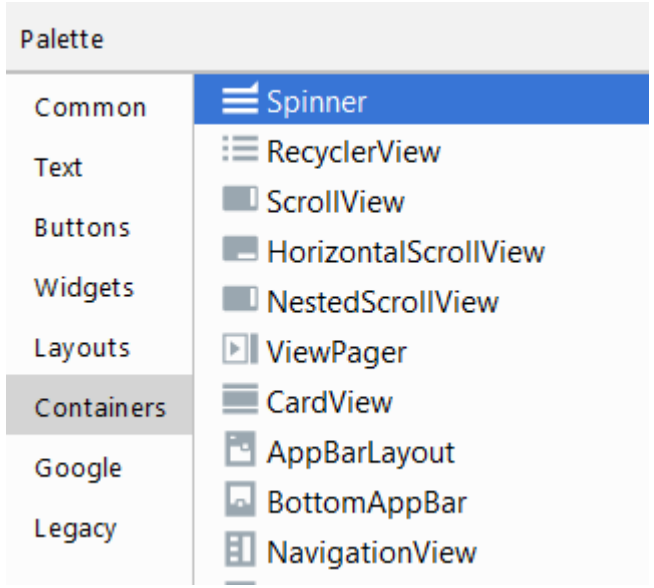
All activity elements should be inside the LinearLayout tag.



CONTAINERS -- SPINNER

- A drop-down menu is called spinner. The SpinnerView displays one item at a time from a list and enables users to choose from them.
- Note: Radio buttons allow the user to select one option from a set. You should use radio buttons for optional sets that are mutually exclusive if you think that the user needs to see all available options side-by-side. If it's not necessary to show all options side-by-side, use a **spinner** instead.
- key attributes:

android:clickable="***bool***"	set to false to disable the spinner
android:id="@+id/***theID***"	unique ID for use in Kotlin code
android:entries="@array/***array***"	set of options to appear in spinner (must match an array in strings.xml)
android:prompt="@string/***text***"	title text when dialog of choices pops up



CONTAINERS -- SPINNER

- XML code:

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dp"
    android:layout_marginLeft="20dp"
    app:layout_constraintTop_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    />
```

CONTAINERS -- SPINNER

- Follow the steps below to fill the SPINNER and grant it actions
  - Create an array
  - Create an array adapter that to perform a *simple_spinner_item* with items from the array you previously created
  - Attach the array to the adapter of the spinner
  - Configure the onItemSelectedListener
    - AdapterView.OnItemSelectedListener
    - override fun onItemSelected

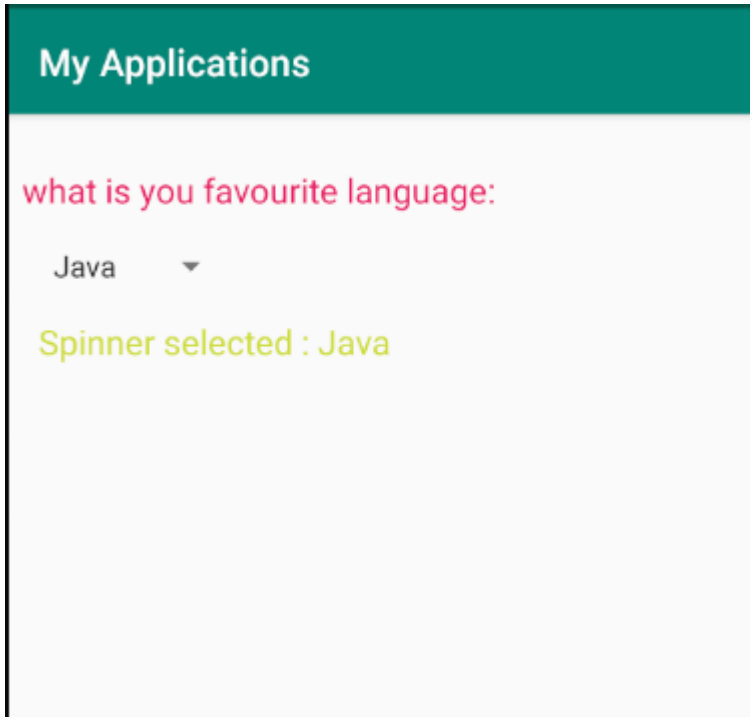
CONTAINERS -- SPINNER

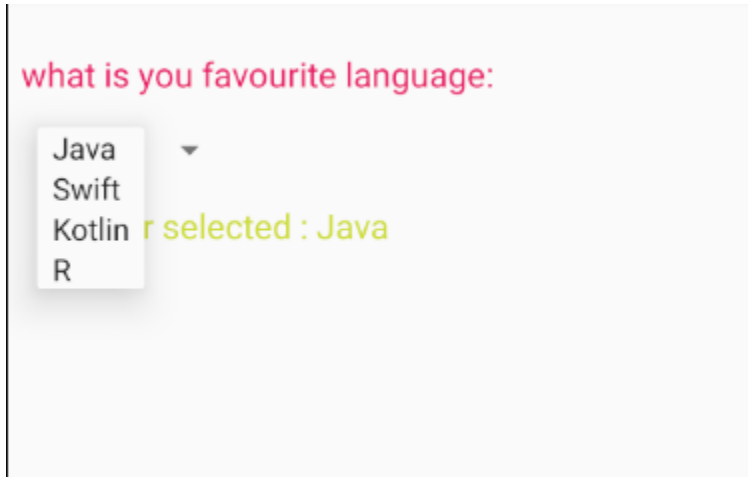
- Kotlin code:

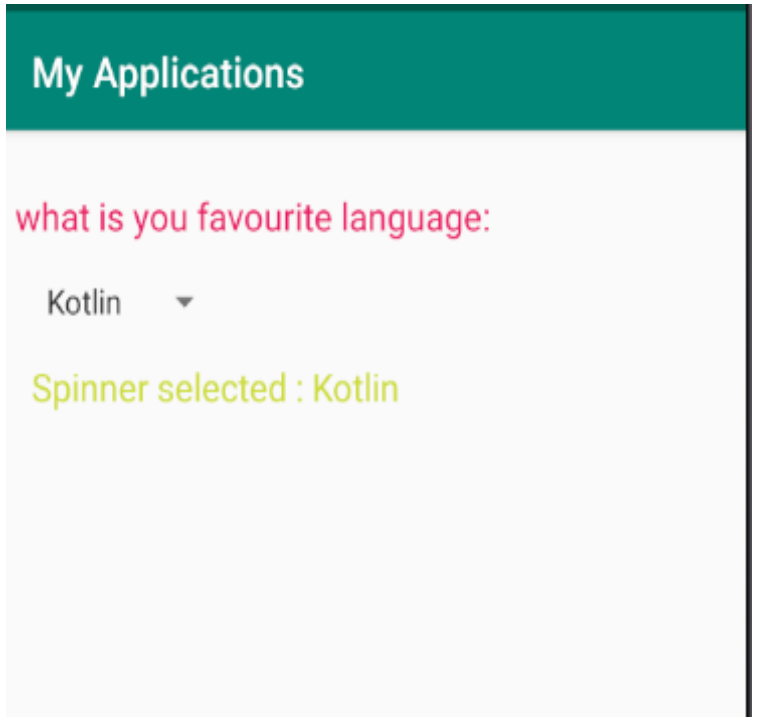
```
val programmingLang= arrayOf("R","Kotlin","Java","Z")//Array creation
val arrAda= ArrayAdapter(this,android.R.layout.simple_spinner_item,programmingLang)//Creating an array adapter
val spinner1= findViewById<Spinner>(R.id.spinner1)
val textView4= findViewById<TextView>(R.id.textView4)

spinner.adapter=arrAda  //to attach array adapter to the spinner
spinner.onItemSelectedListener = object:
    AdapterView.OnItemSelectedListener {
    override fun onItemSelected(
        parent: AdapterView<*>,
        view: View,
        position: Int,
        id: Long
    ) {
        textView.text ="Spinner selected : ${parent.getItemAtPosition(position).toString()}"
    }
    override fun onNothingSelected(parent: AdapterView<*>) {
    }
}
```

CONTAINERS -- SPINNER







After run the program