Building a View and MVC

@somkiat

Topics

Building a View
Model-View-Controller (MVC)
Workshop
Homework

GUI Architecture of Android

Single thread (Main Thread)
Event-driven
Nestable components
Model-View-Controller pattern

MVC

Model View Controller

Model

Represent the **data** or data container
e.g. data store or database

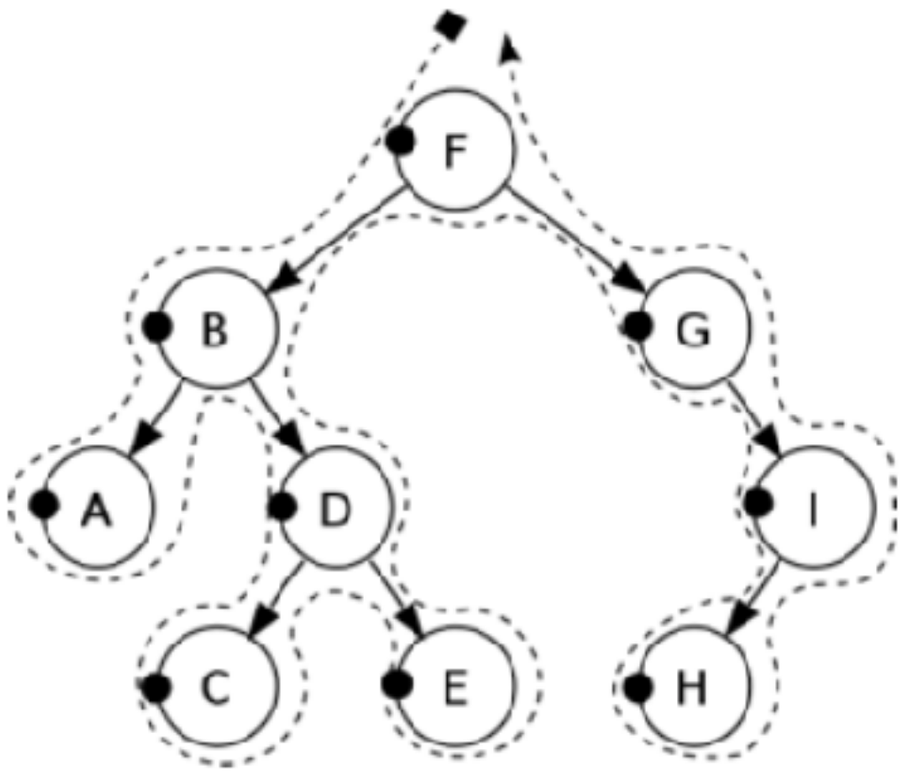
View

Responsible for rendering the display
Sending audio to speakers

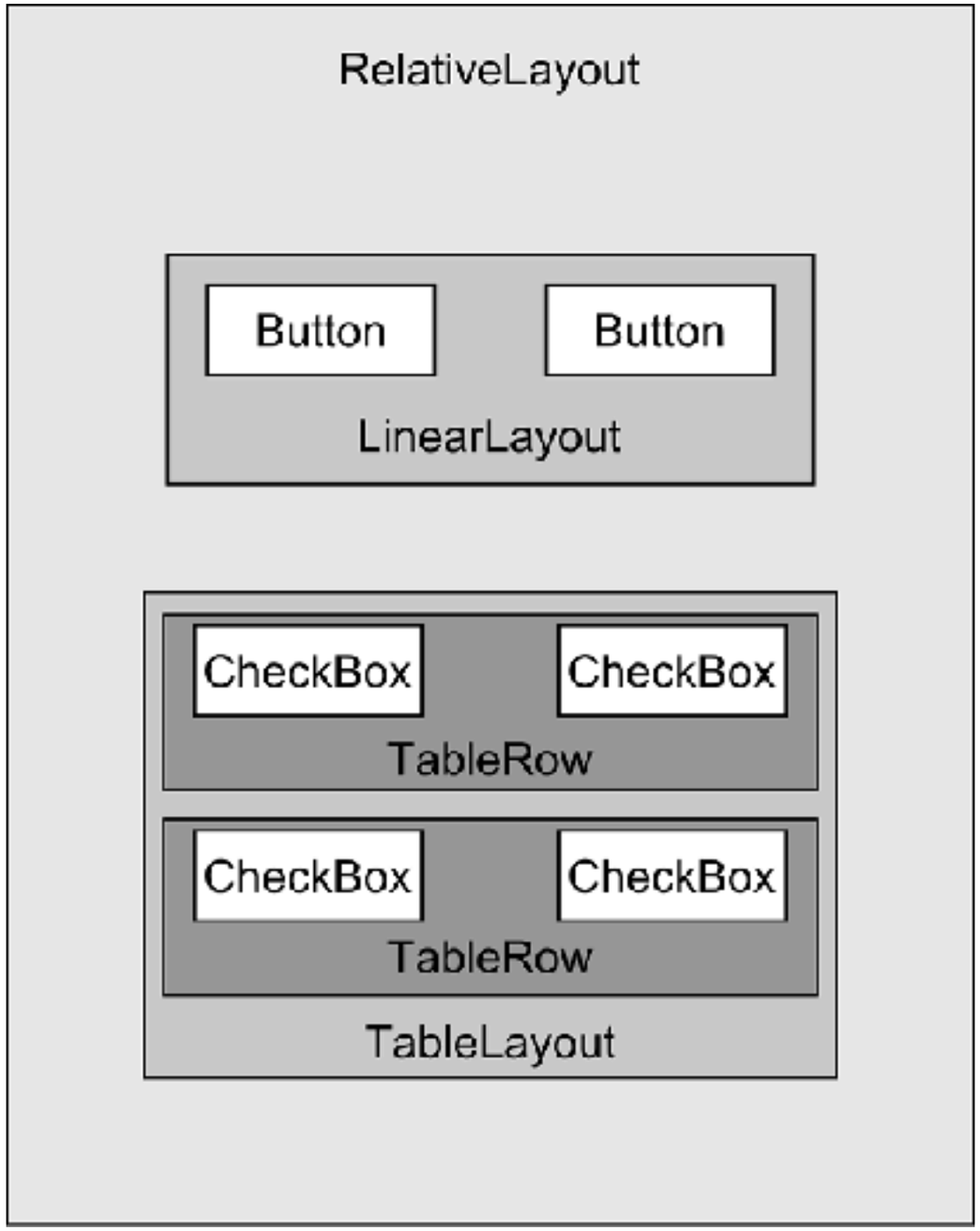
In android, it’s extend from **View class**

View

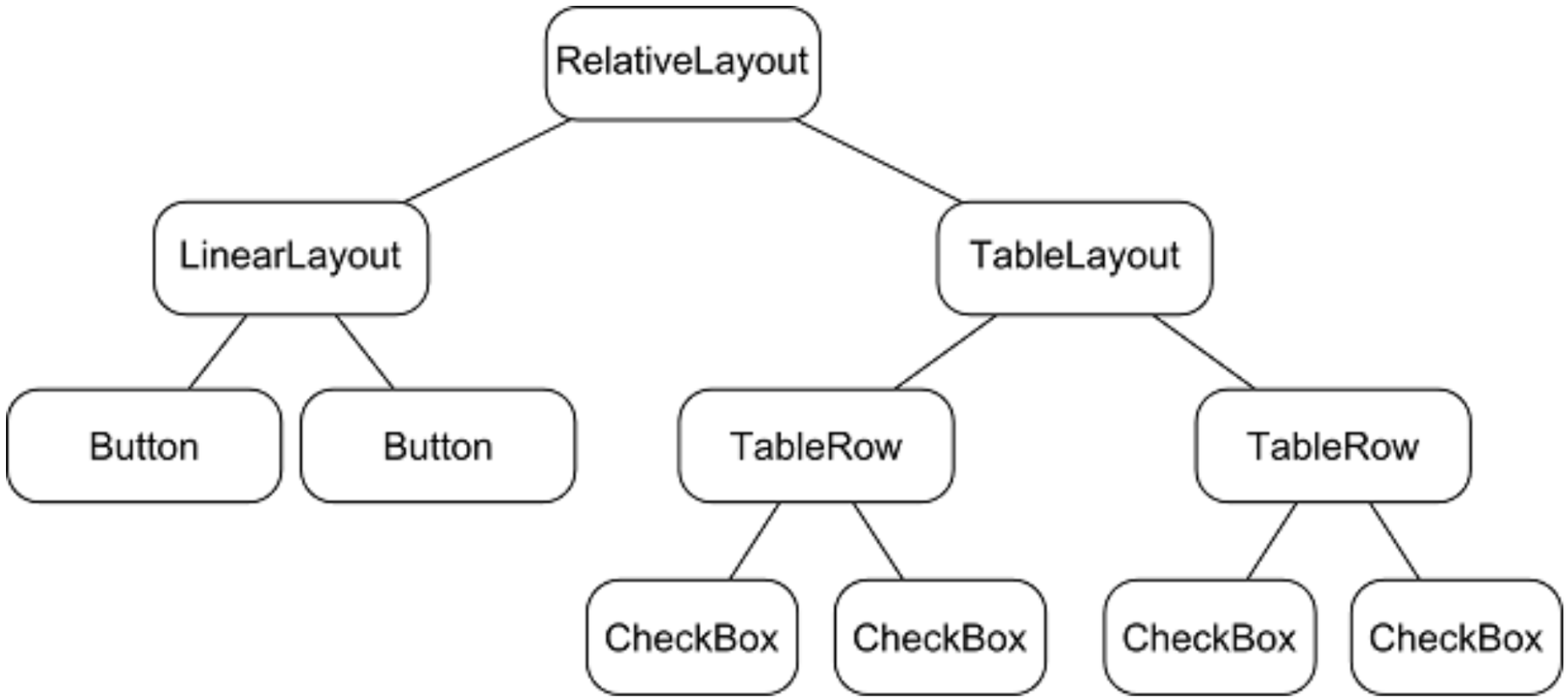
In Android, it’s extend from **View class**



View Hierarchy

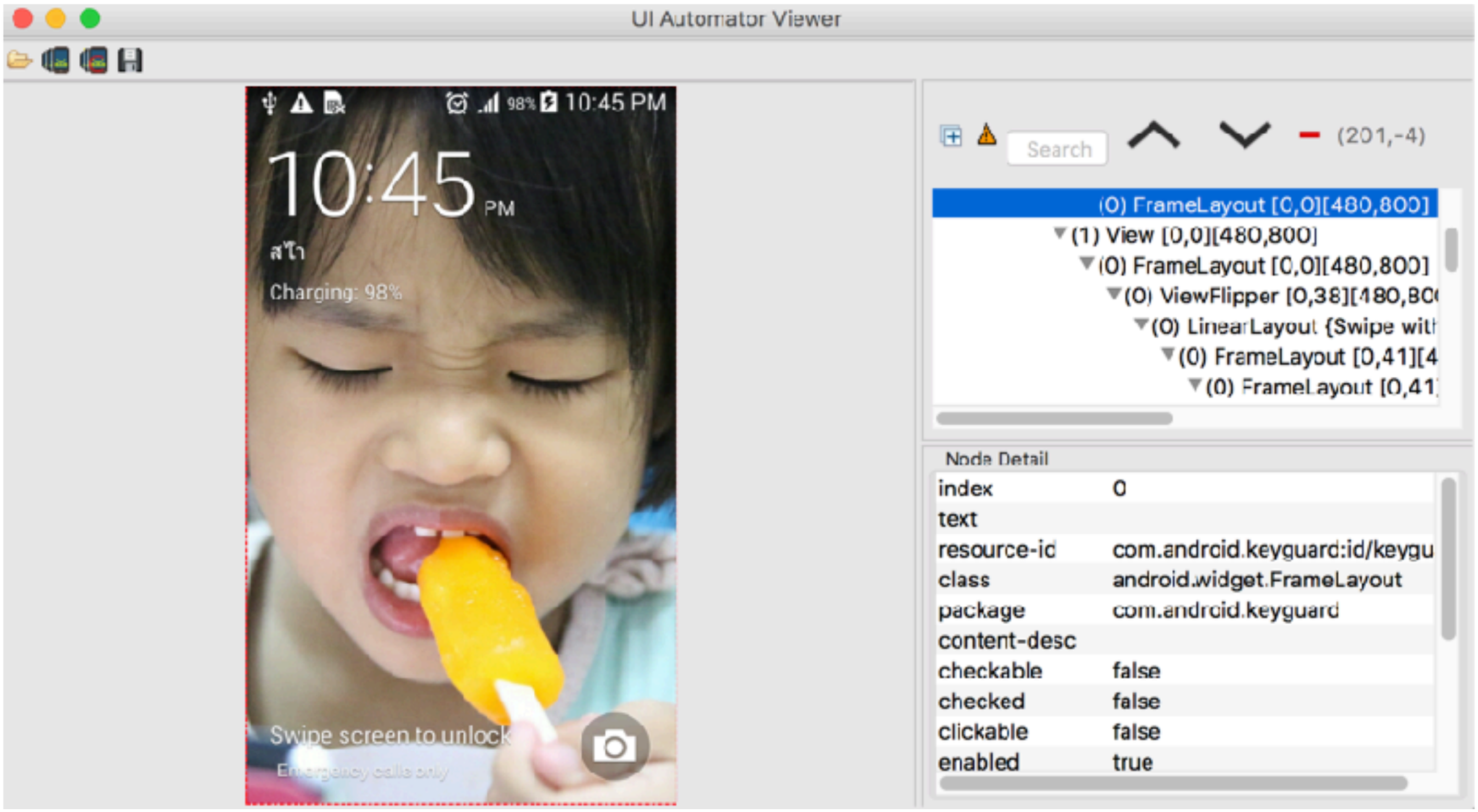


View Hierarchy



UIAutomator Viewer

$ANDROID_HOME/tools/bin/uiautomatorviewer



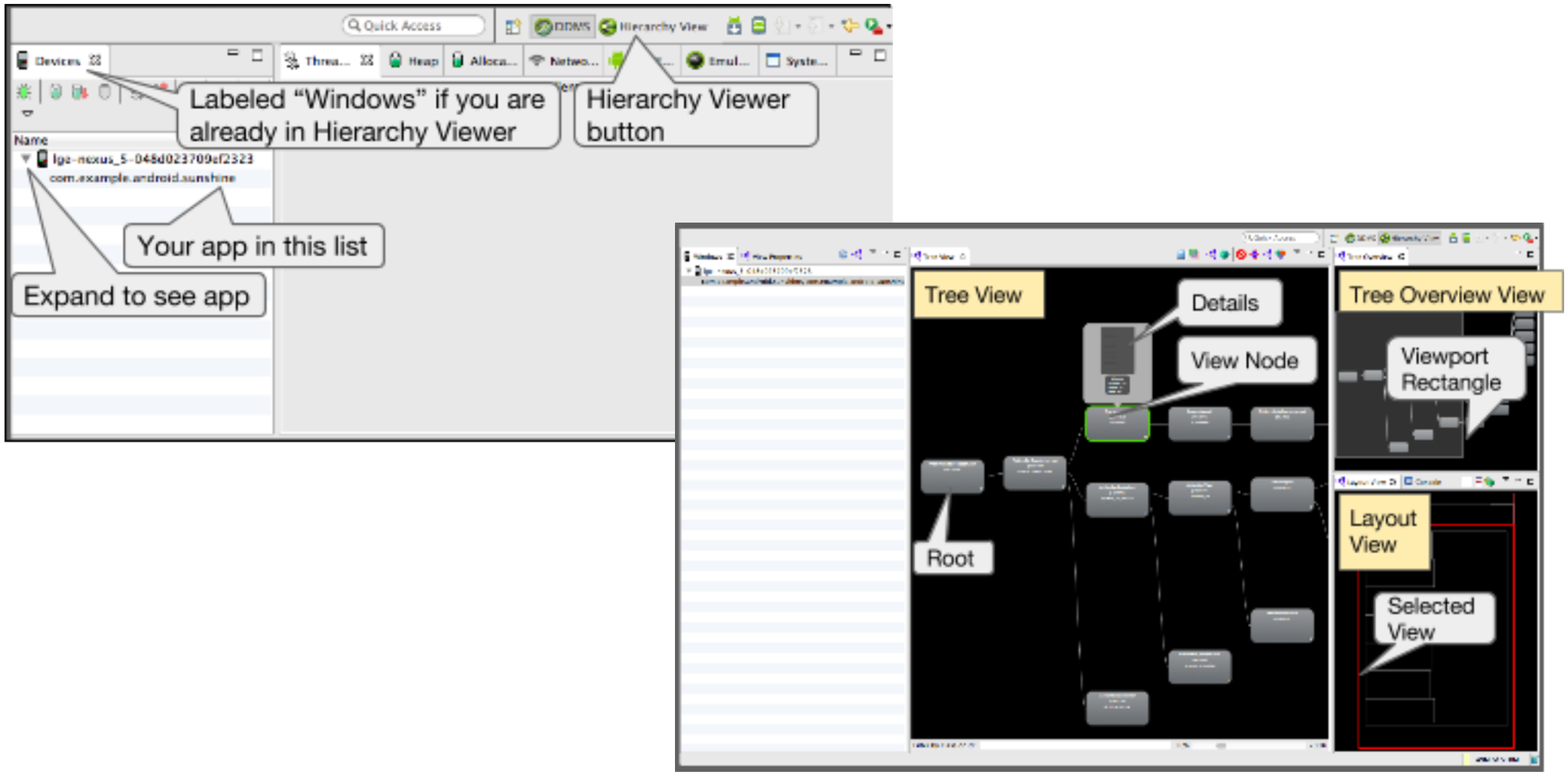
Improving your layout

Inspect your layout
Revise your layout
Use Lint
Loading view on demand
more ...

https://developer.android.com/training/improving-layouts/index.html

Profiling your layout

Hierarchy Viewer



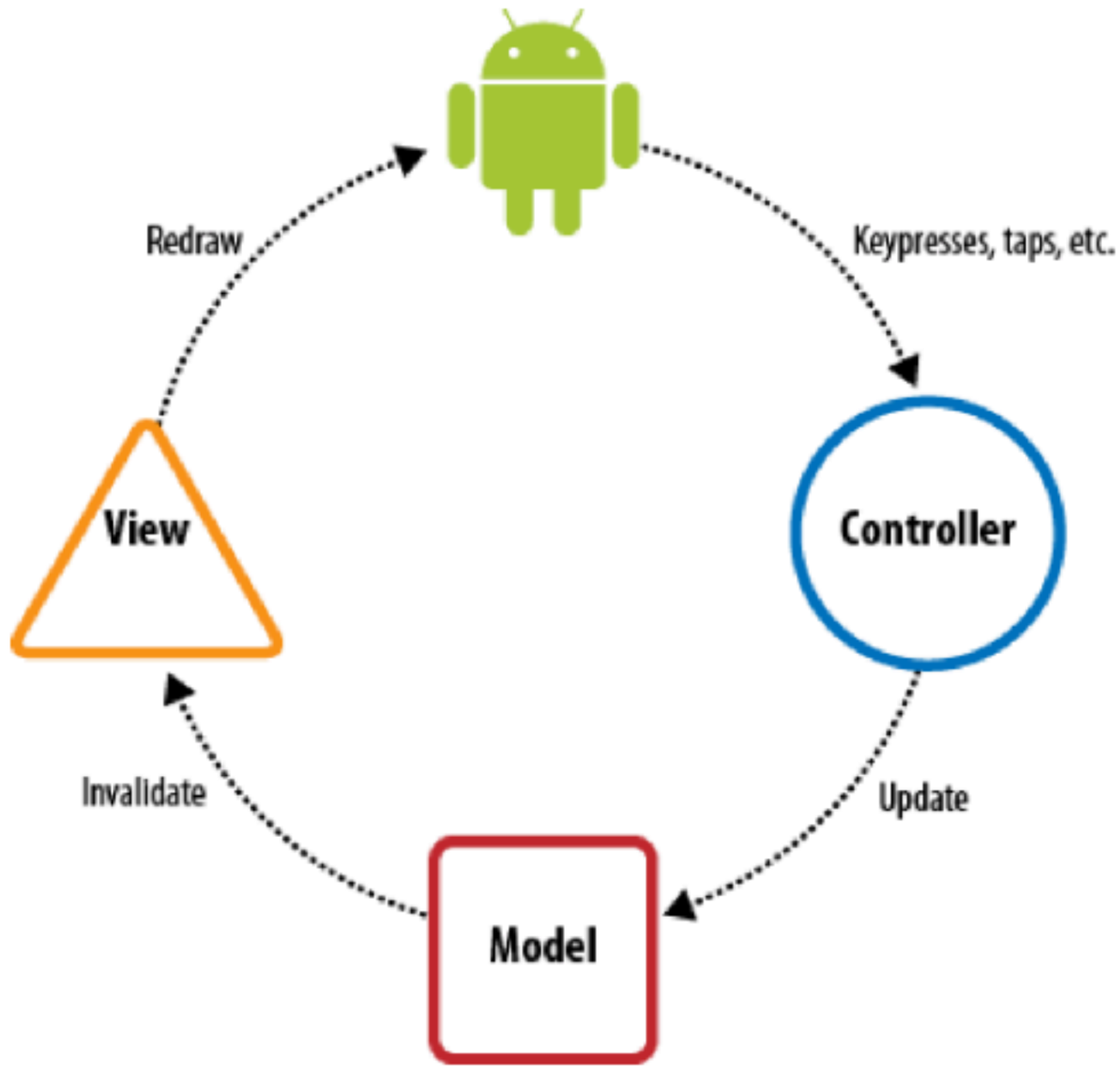
https://developer.android.com/studio/profile/hierarchy-viewer.html

Controller

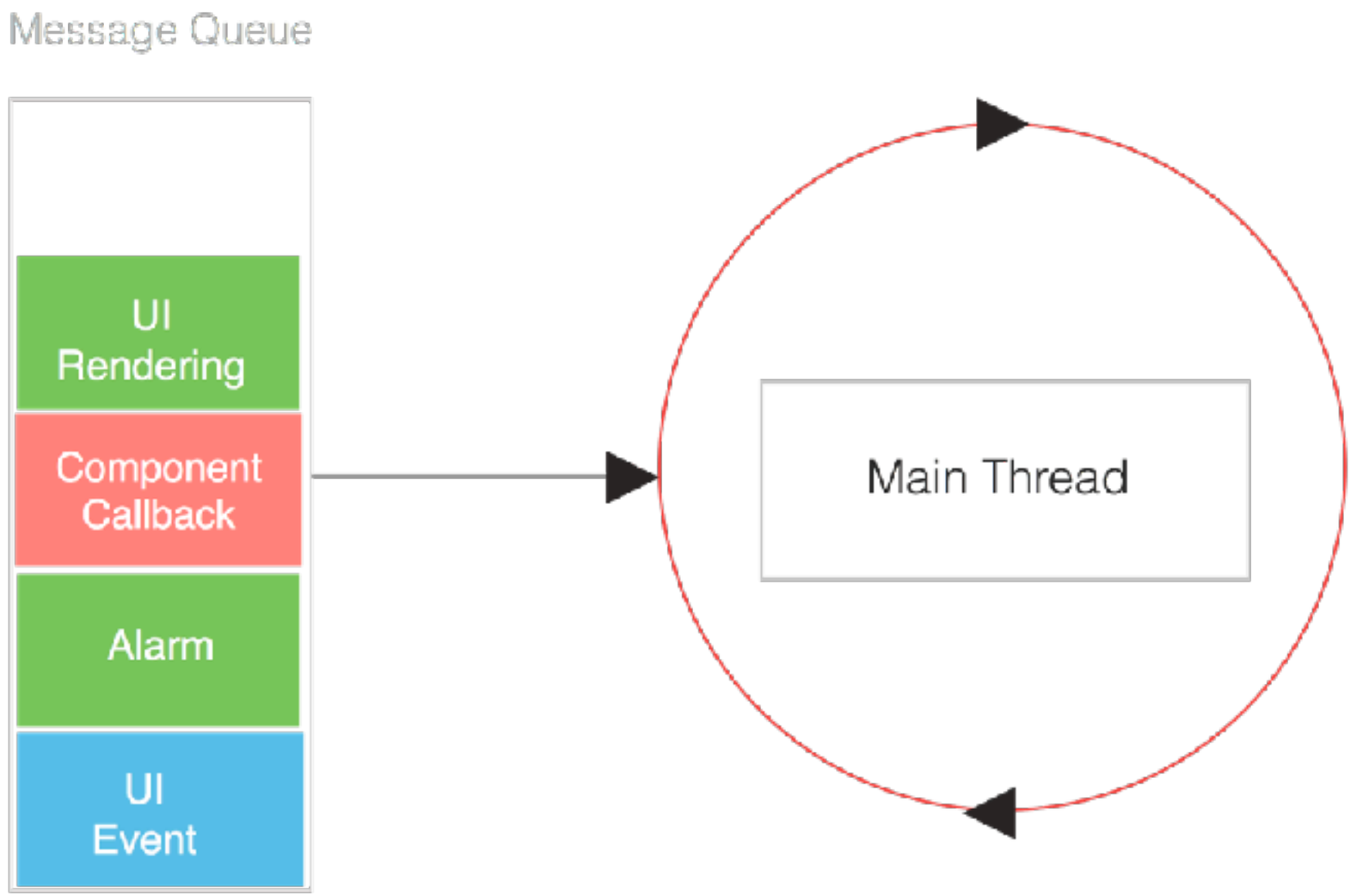
Responds to external actions
e.g. keystone, swipe, tab, incoming call

Implemented as a **event queue**

Update model from actions/events

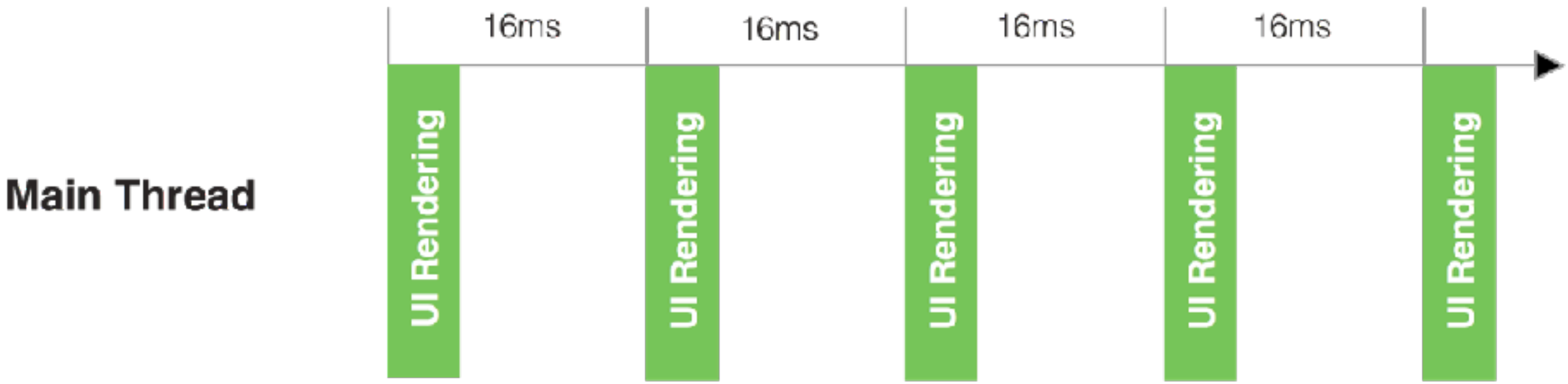

Redraw
Keypresses, taps, etc.
View
Controller
Invalidate
Update
Model

Main Thread Queue



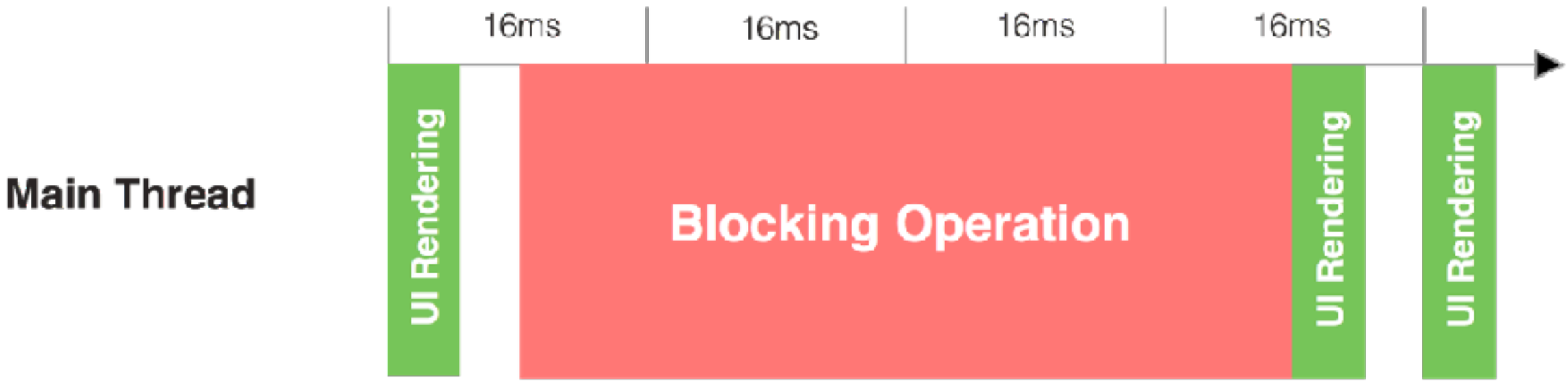
http://hvasconcelos.github.io/articles/Offloading-work-from-the-UI-Thread

Main Thread Rendering

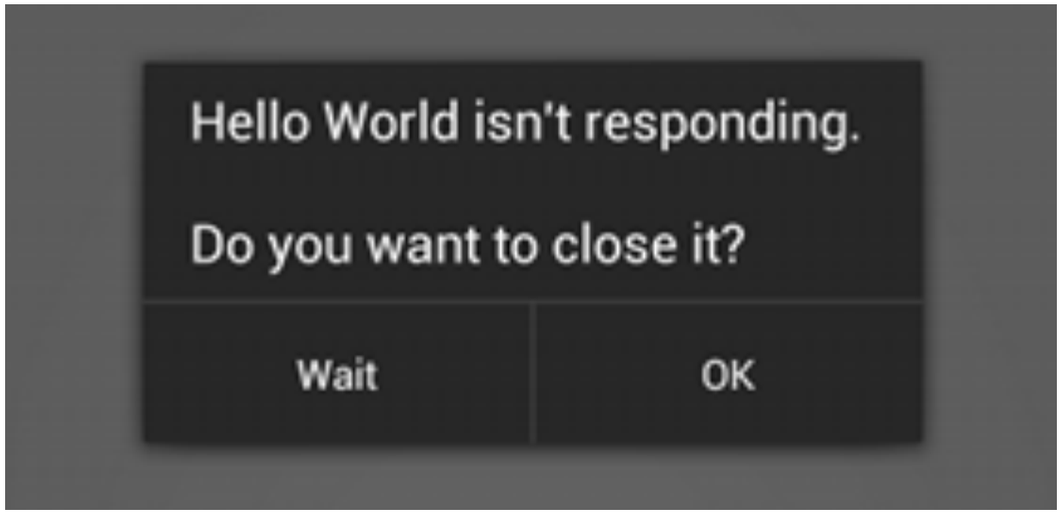


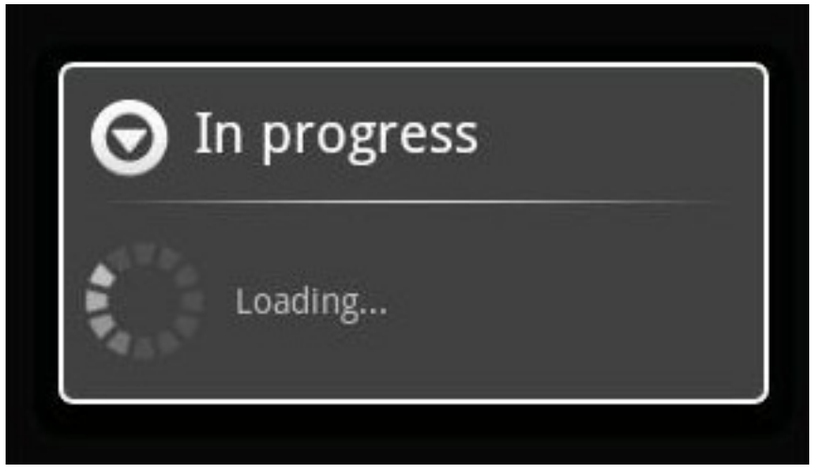
Most android devices refresh the screen 60 times/second, every 16 ms

Blocking Operation !!

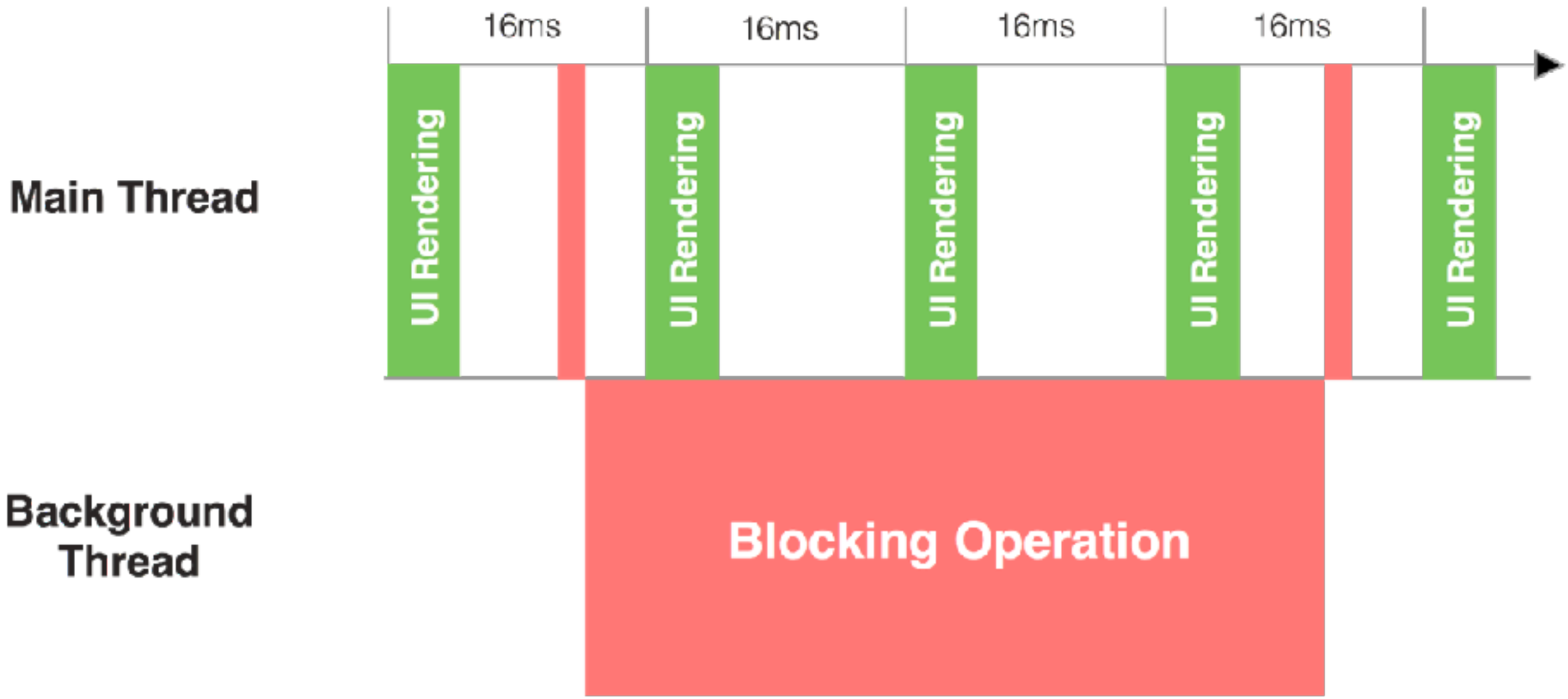


Blocking Operation !!





Background Thread



Asynchronous Techniques

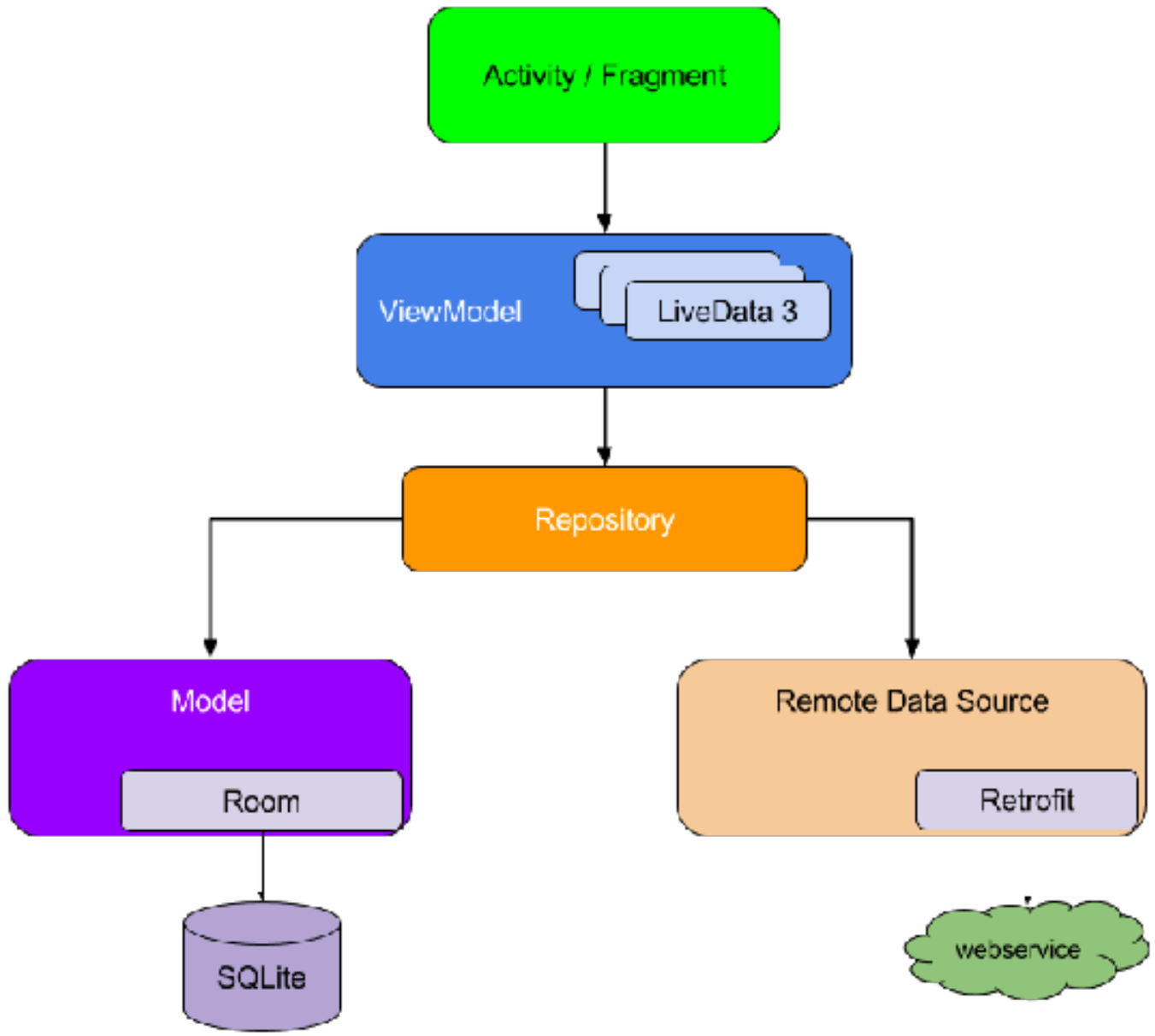
Thread
AsyncTask
Loader
IntentService
JobScheduler
RxJava

More !!!

Model View Presenter
Model View ViewModel
View Interactor Presenter Entity Router
Clean Architecture

more ^_^

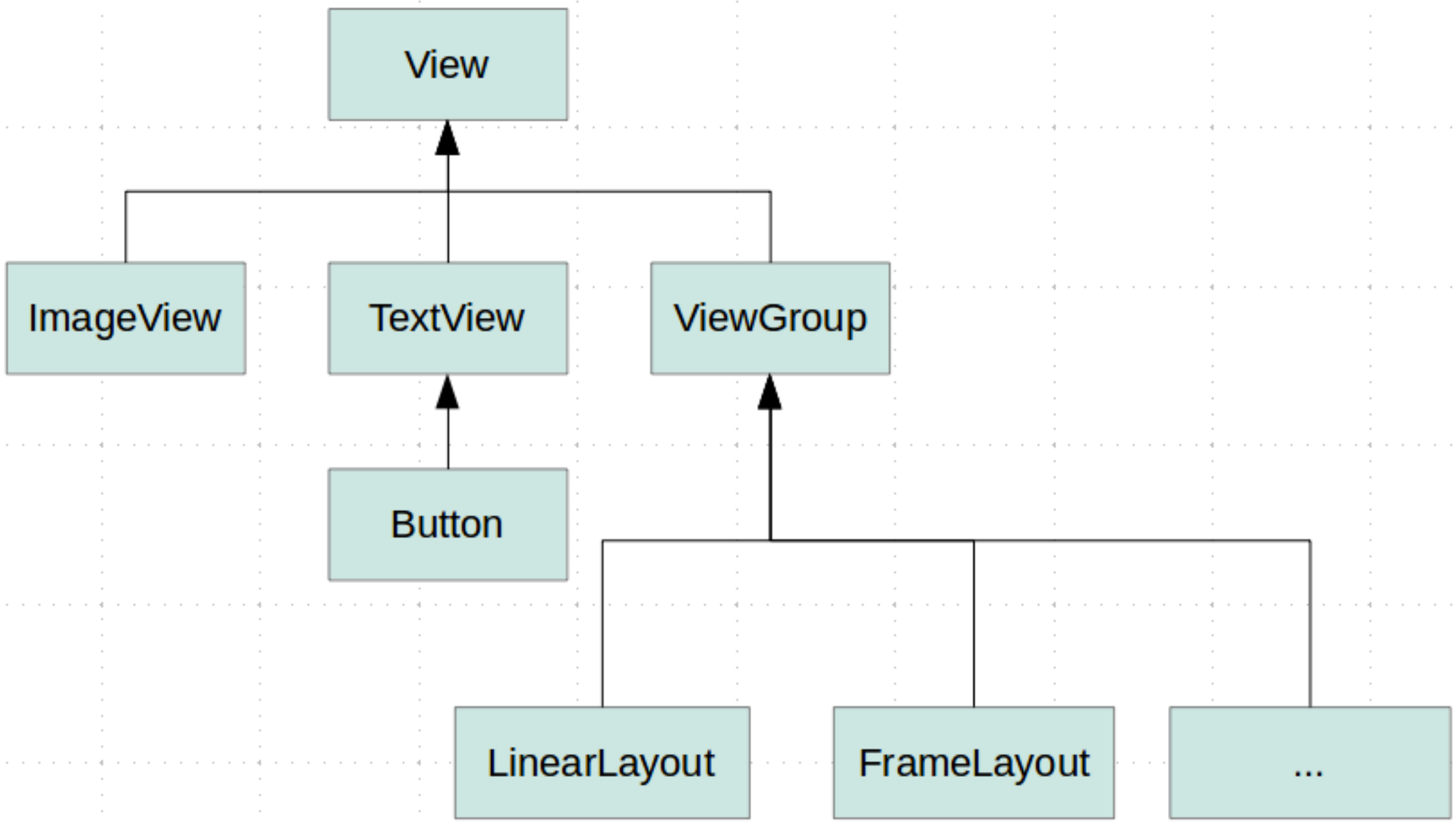
Architecture Components



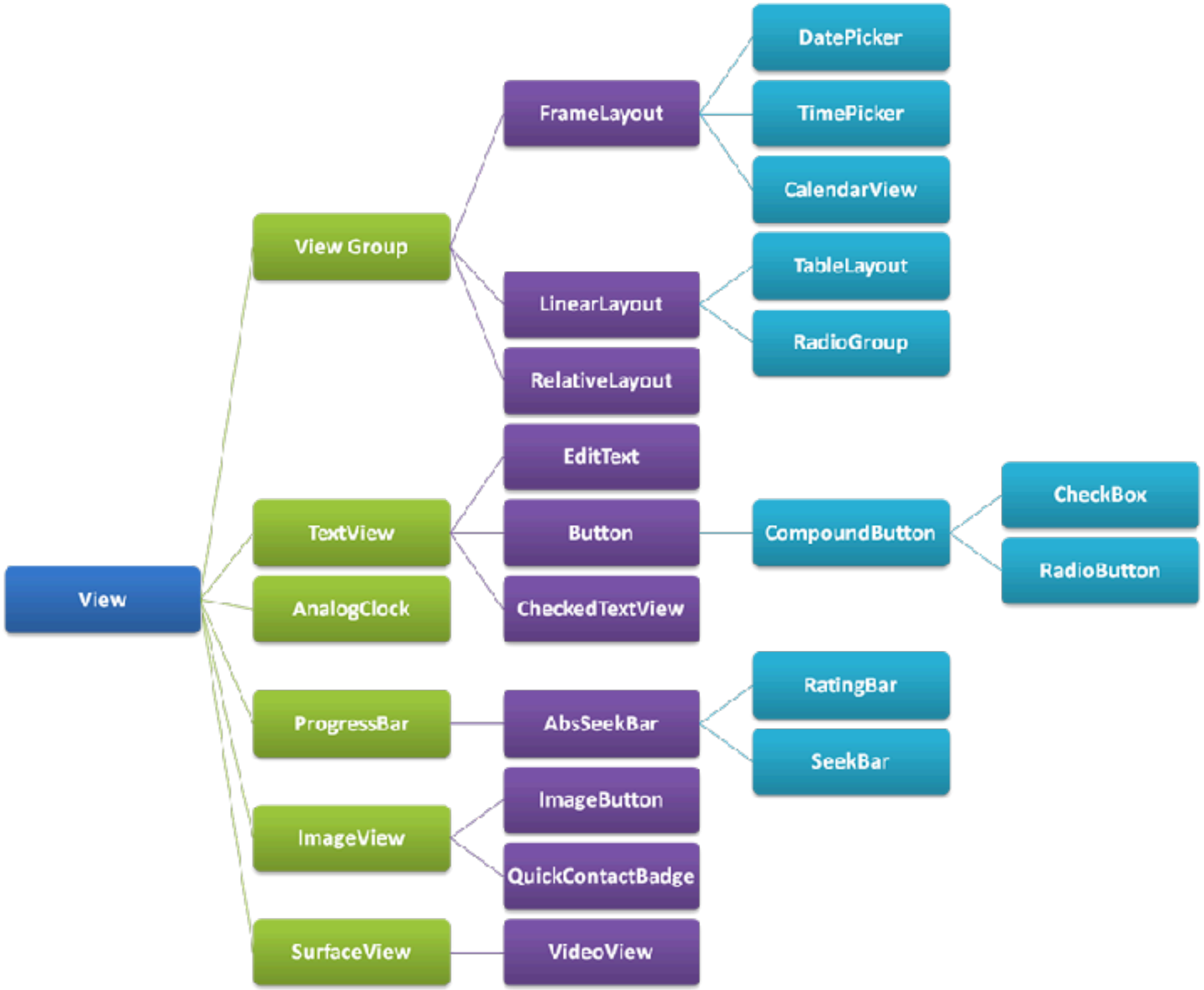
https://developer.android.com/topic/libraries/architecture/guide.html

Start workshop

Android View



Android View



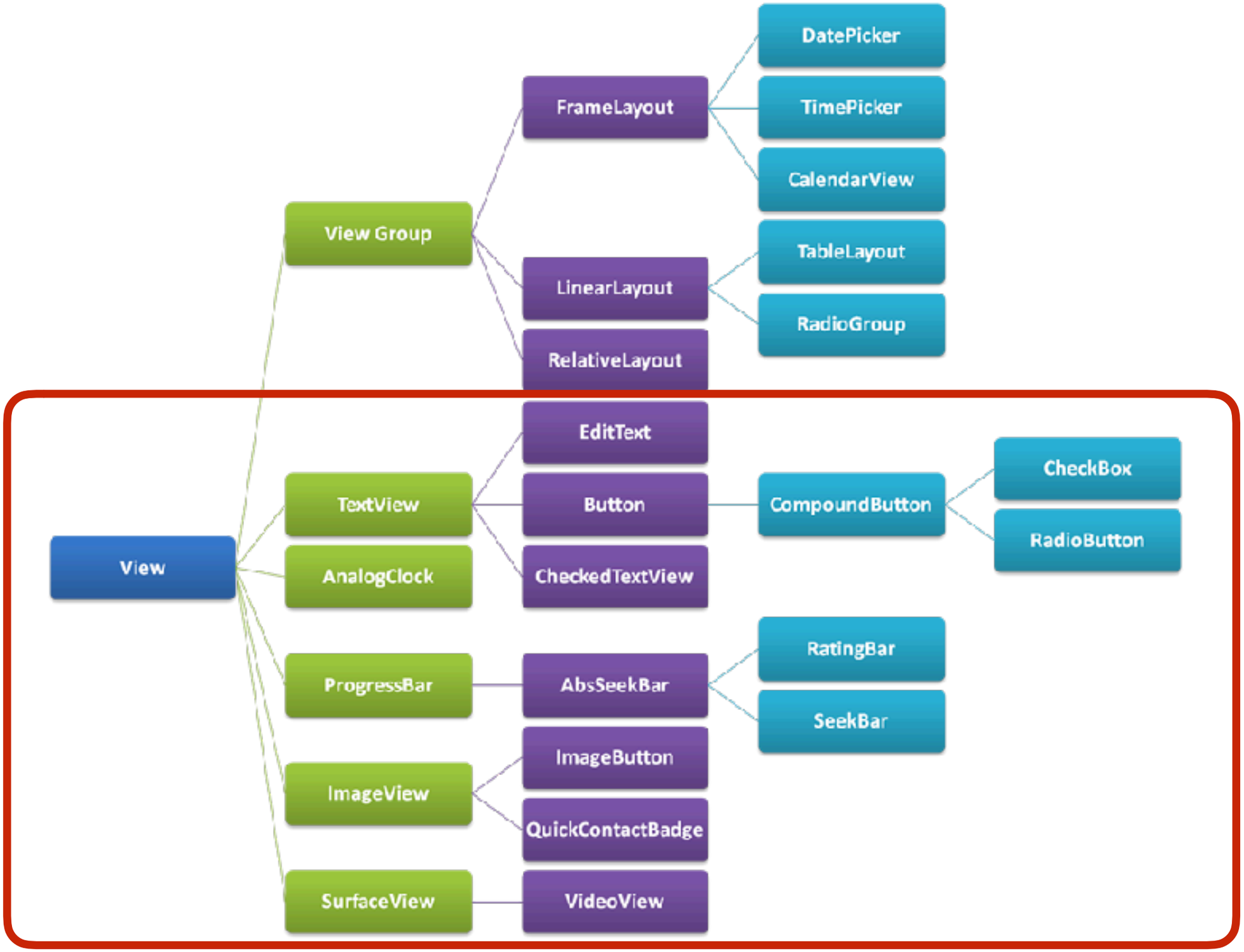
Android View

Widgets
Container views

Widgets

Displayed on screen
Extended from **android.view.View**
e.g. Button, TextView, EditText, ListView

Widgets


View
View Group
FrameLayout
DatePicker
TimePicker
CalendarView
LinearLayout
TableLayout
RadioGroup
RelativeLayout
TextView
EditText
Button
CompoundButton
CheckBox
RadioButton
CheckedTextView
AnalogClock
ProgressBar
AbsSeekBar
RatingBar
SeekBar
ImageView
ImageButton
QuickContactBadge
SurfaceView
VideoView

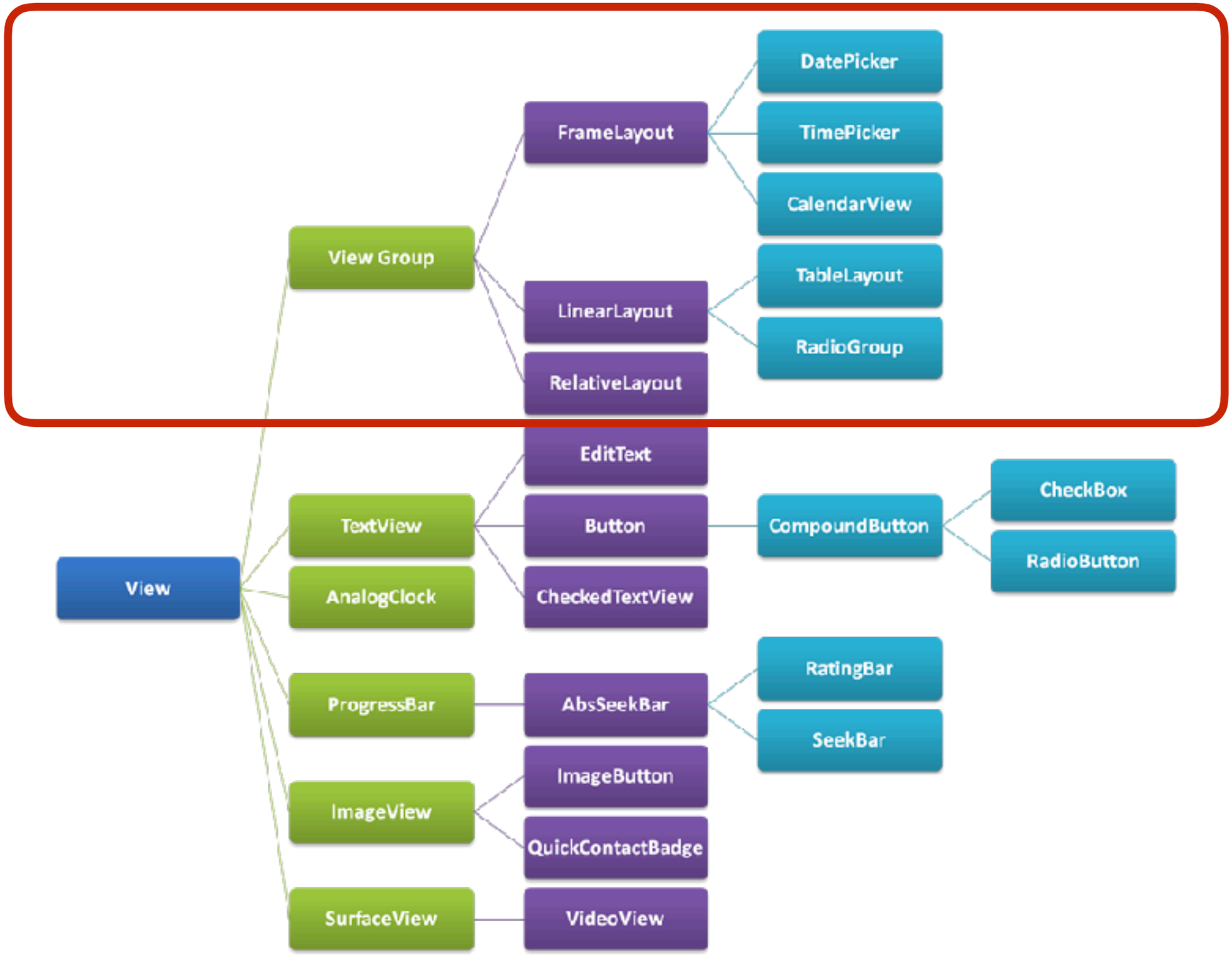
Container Views

User can’t see this view

Extended from **android.view.ViewGroup**

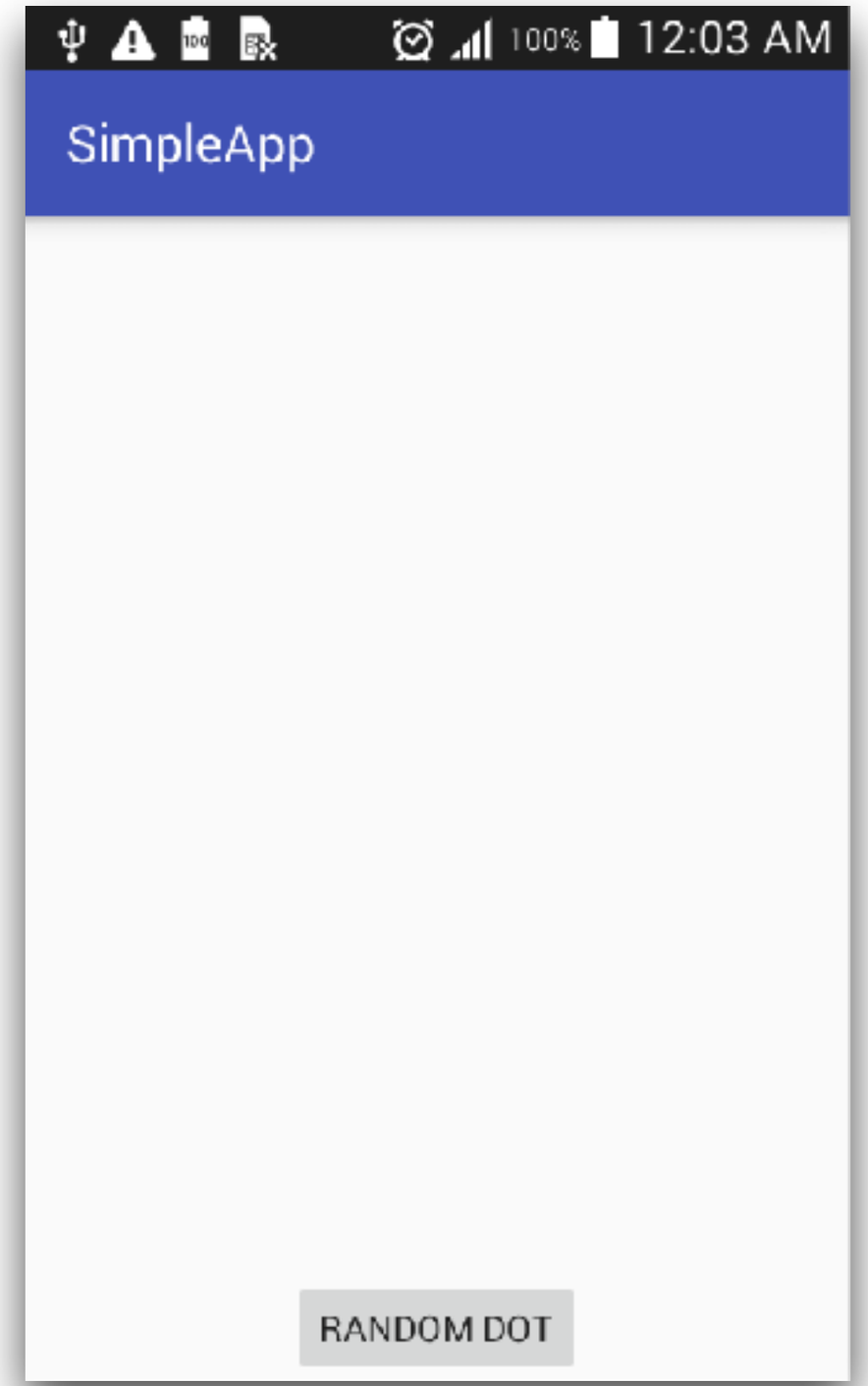
e.g. Button, TextView, EditText, ListView

Container Views

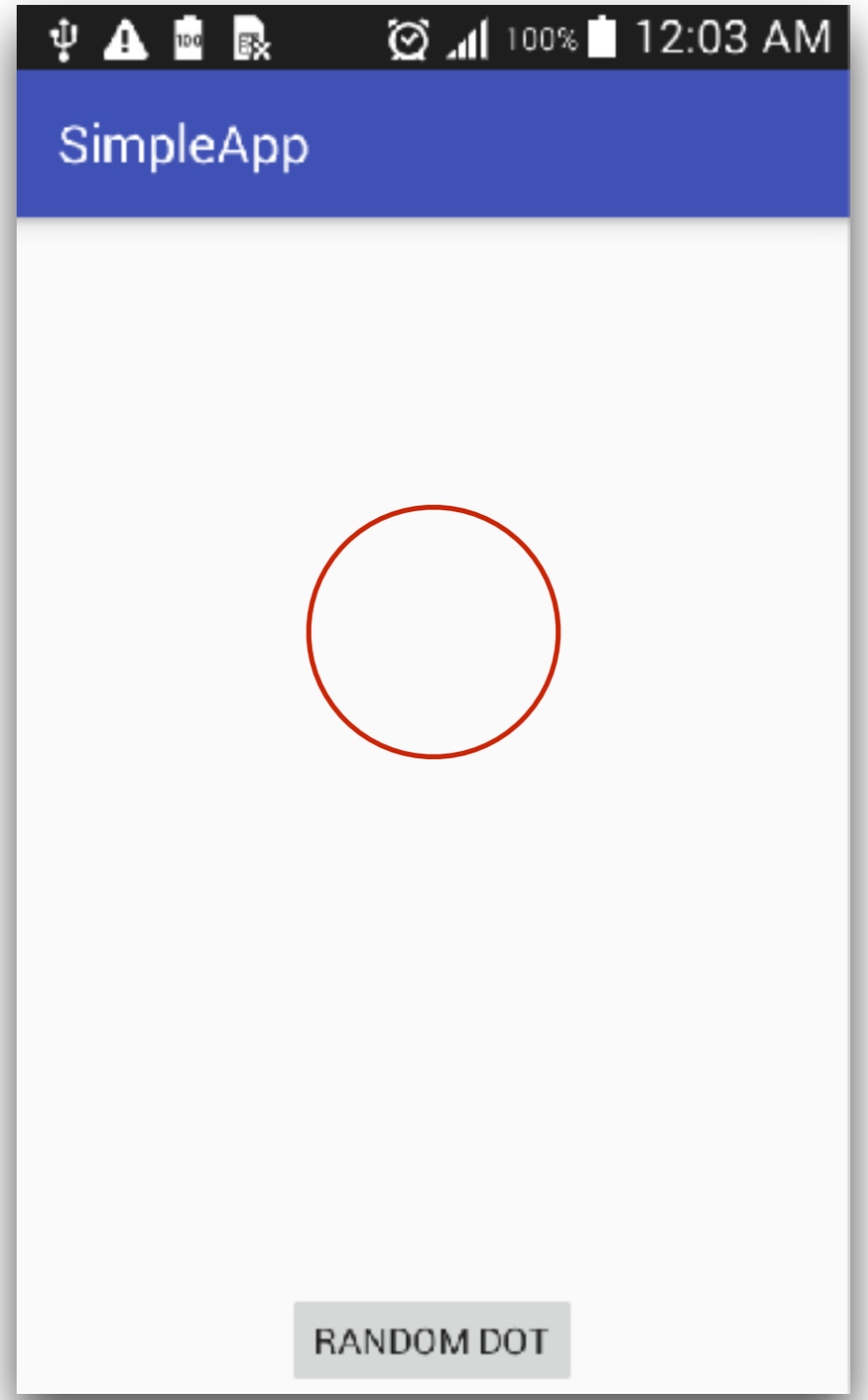


Start workshop

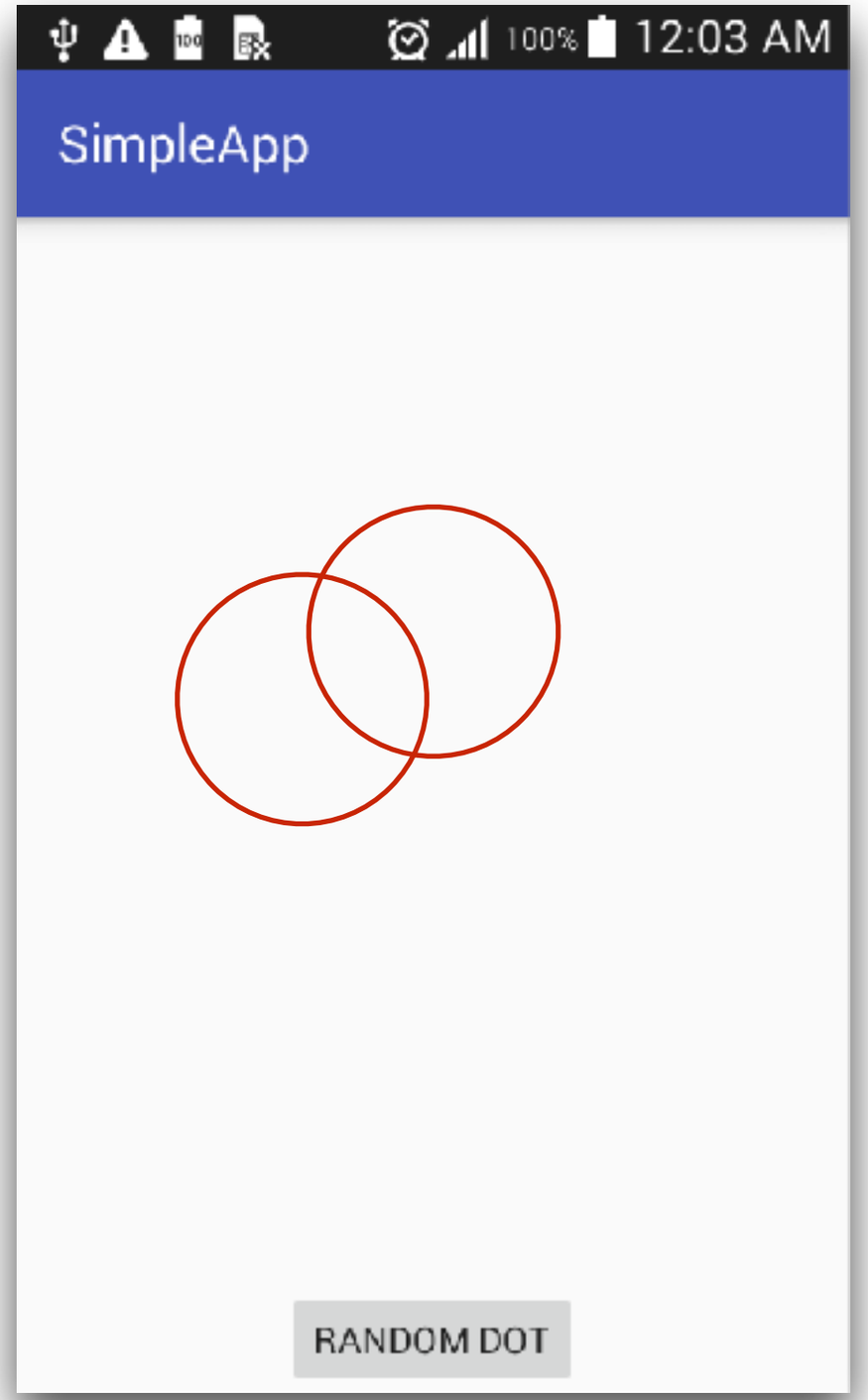
My Dot App



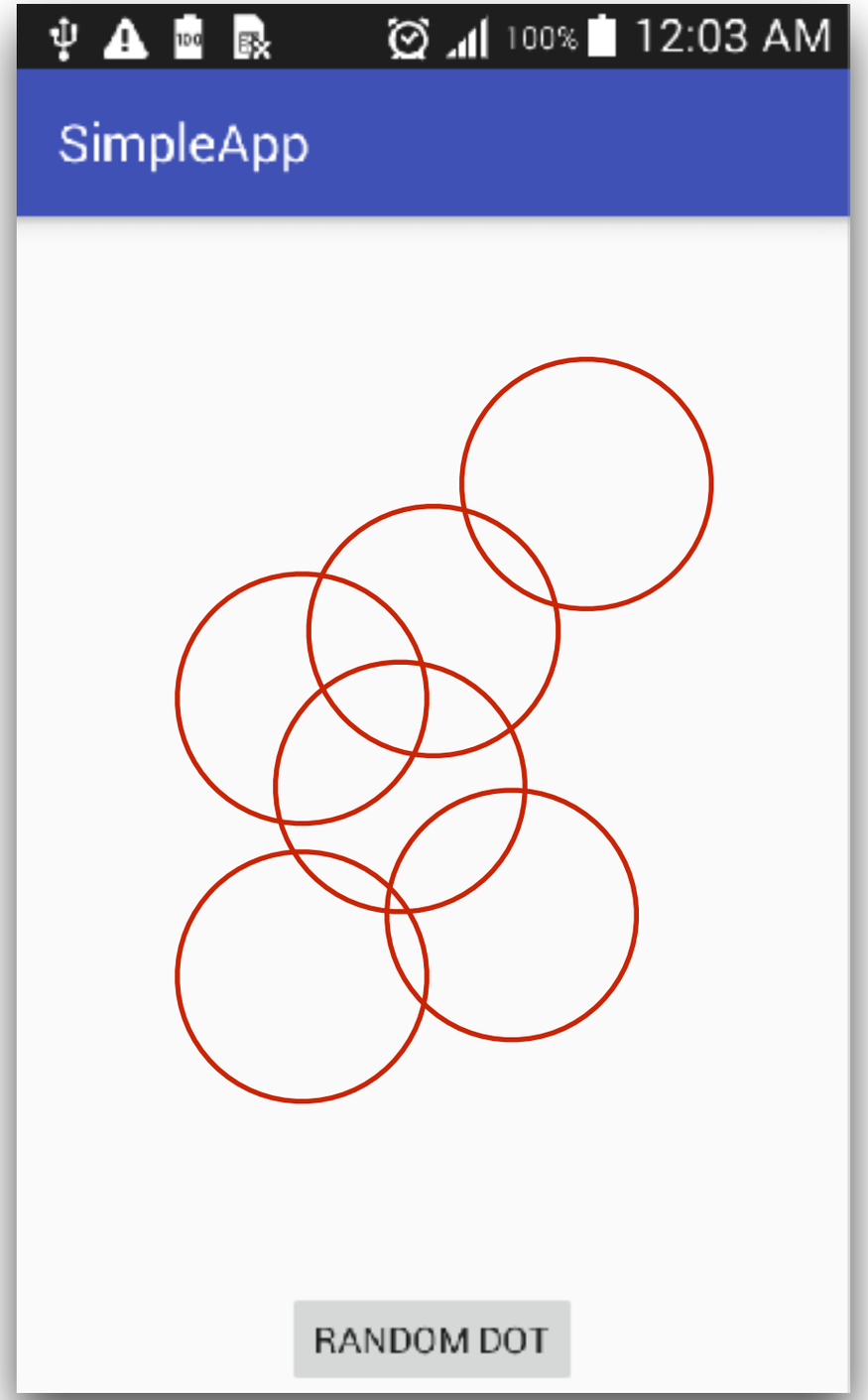
My Dot App



My Dot App



My Dot App

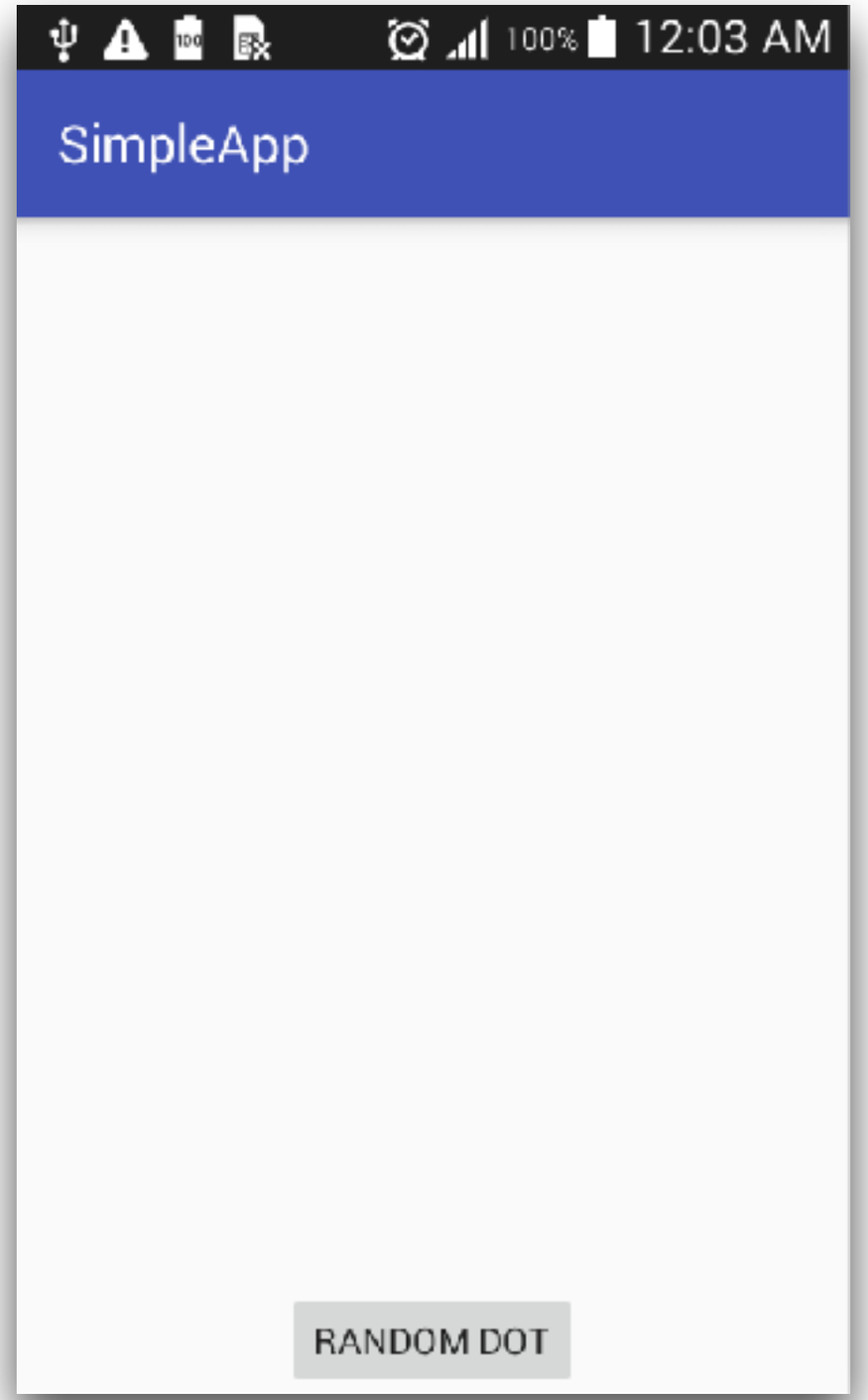


Let’s coding

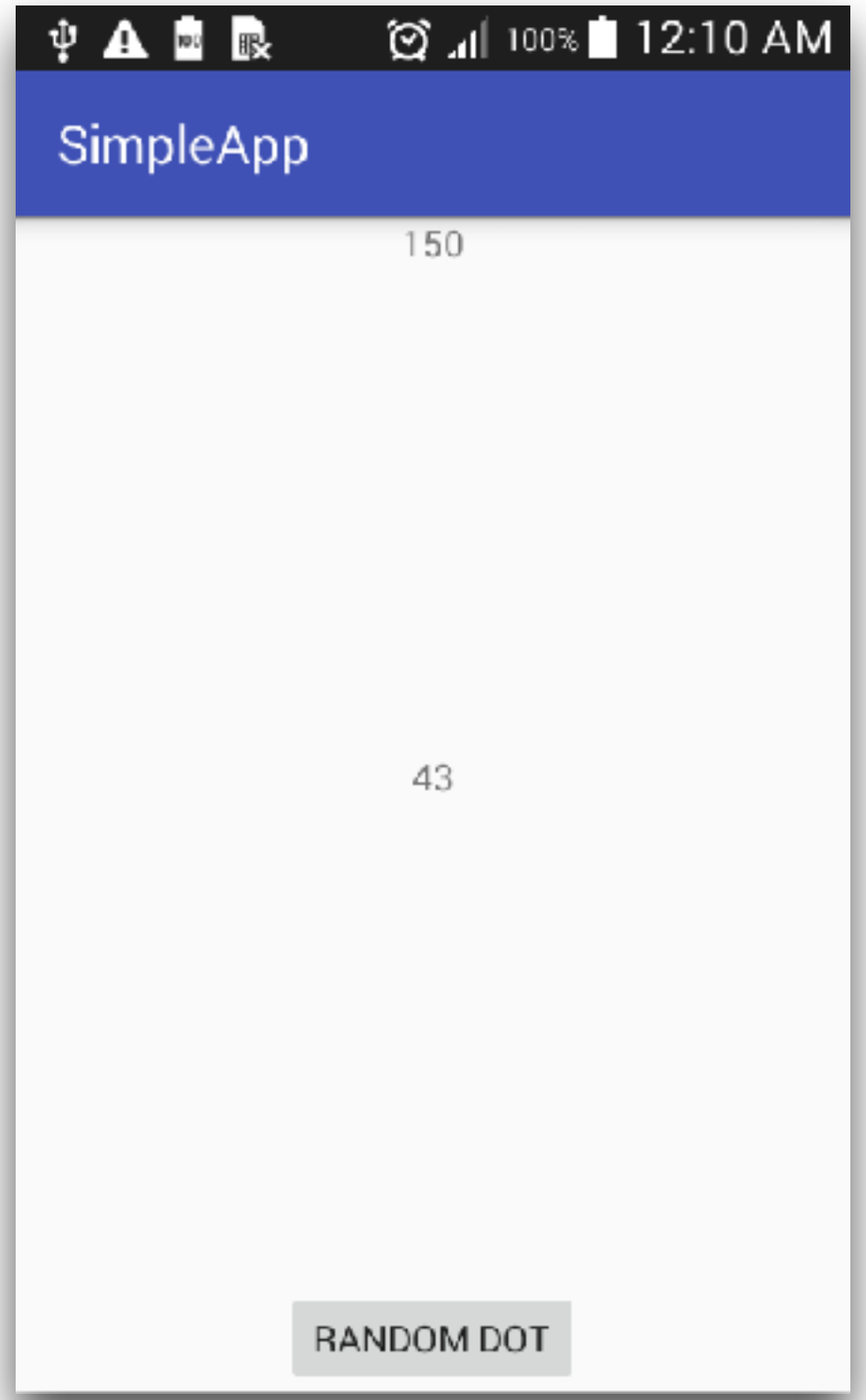
1. Create new project

project name = **SimpleMyDot**
package name = **kmitl.lab03.<your name>**
folder in GitHub = **lab/lab03**

2. Create main layout



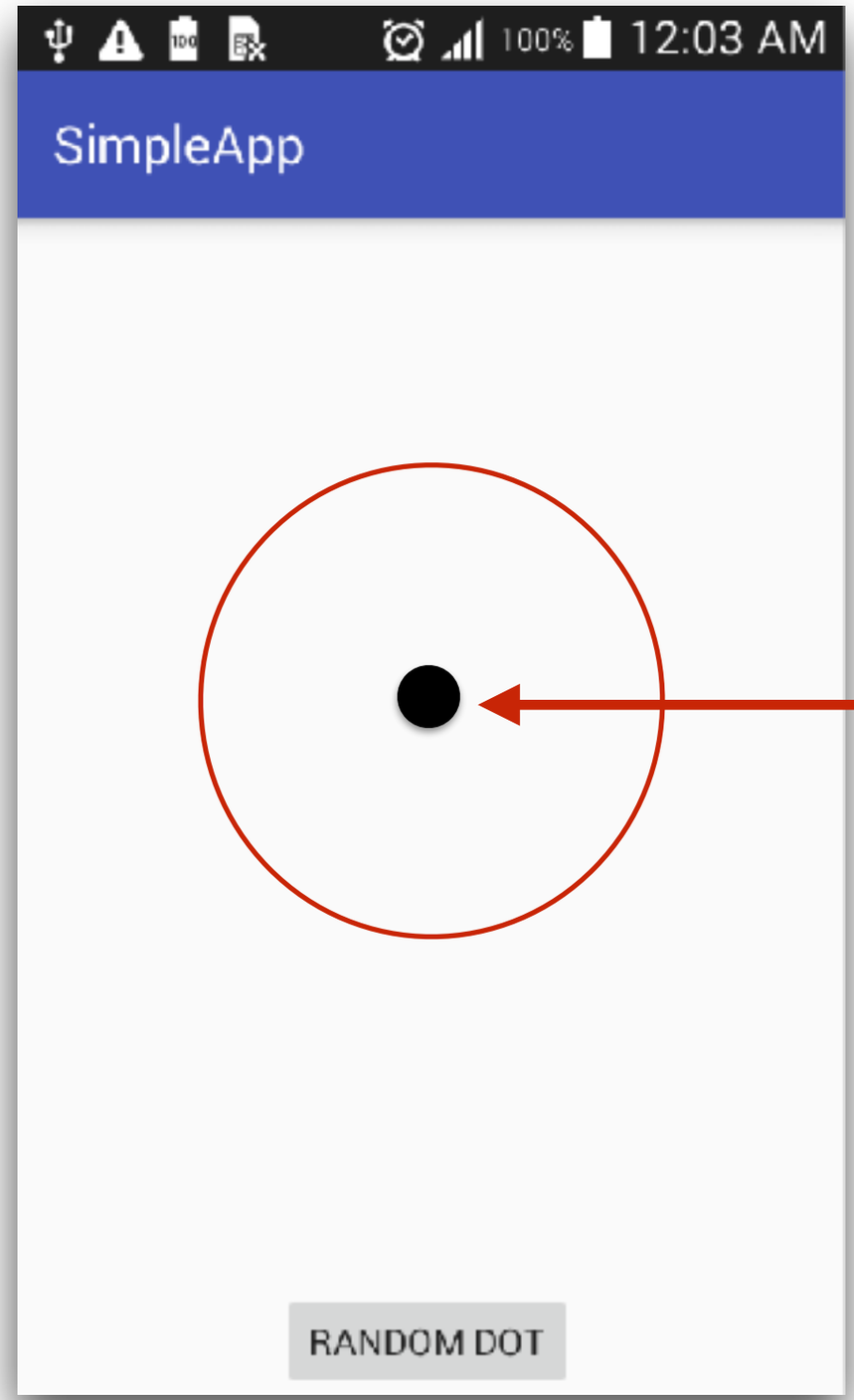
3. Handle action when clicked



Review your code

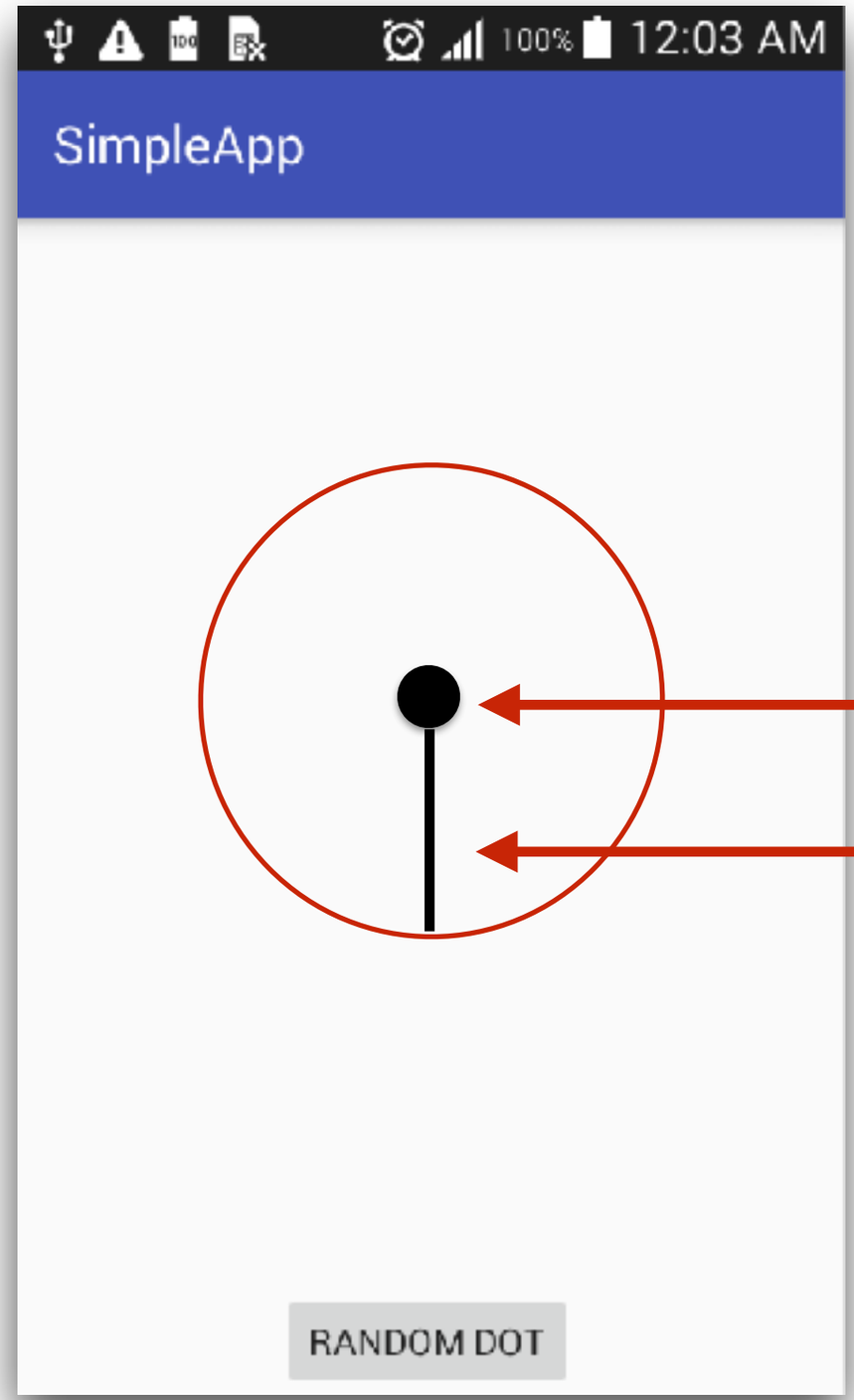
What is your model ?
What is your view ?
What is your controller ?

Dot model



Center point => x, y

Dot model



Center point => x, y

Radius

4. Create Dot class

In package name = **model**

4. Create Dot class

```
package lab03.kmitl.simpleapp.model;

public class Dot {

    private int centerX;
    private int centerY;
    private int radius;

}
```

5. Update Main Activity

Create new dot object
Display data of dot object to View

5. Update activity

```
public void randomDot(View view) {
    //Random a Dot
    Random random = new Random();
    int x = random.nextInt(200);
    int y = random.nextInt(200);
    Dot randomDot = new Dot(x, y, 20);

    //Draw dot model to view
    TextView coordX = (TextView) findViewById(R.id.txtCoordX);
    TextView coordY = (TextView) findViewById(R.id.txtCoordY);
    coordX.setText(String.valueOf(randomDot.getCenterX()));
    coordY.setText(String.valueOf(randomDot.getCenterY()));
}
```

Review your code

When a dot will display ?

6. Display dot when data changed

Create new **interface** in Dot class

Interface name = **DotChangedListener**

6. Display dot when data changed

```java
package lab03.kmitl.simpleapp.model;

public class Dot {

    public interface DotChangedListener {
        void onDotChanged(Dot dot);
    }
```

7. Create new variable in Dot class

```
package lab03.kmitl.simpleapp.model;

public class Dot {

    public interface DotChangedListener {
        void onDotChanged(Dot dot);
    }

    private DotChangedListener dotChangedListener;

    public void setDotChangedListener(
            DotChangedListener dotChangedListener) {
        this.dotChangedListener = dotChangedListener;
    }
```

Question

When a dot will display ?

8. Notify when data changed

```
public void setCenterX(int centerX) {
    this.centerX = centerX;
    this.dotChangedListener.onDotChanged(this);
}

public void setCenterY(int centerY) {
    this.centerY = centerY;
    this.dotChangedListener.onDotChanged(this);
}
```

Review your code

Activity should not be display a dot by yourself

Activity should be display a dot when it’s changed

9. Update MainActivity

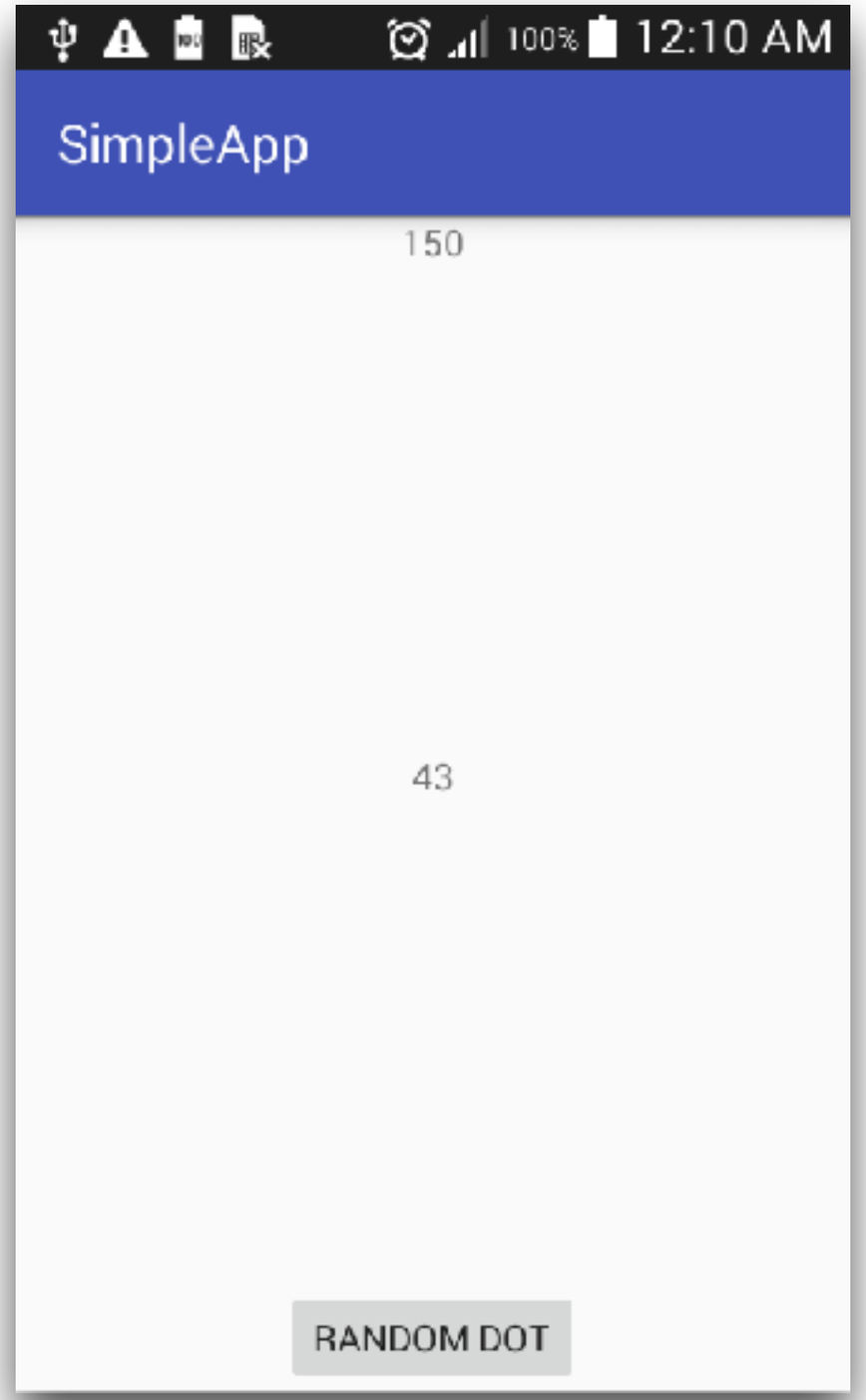
Implements with DotChangedListener

```
public class MainActivity extends AppCompatActivity
implements Dot.DotChangedListener{

    private Dot dot;

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

        // Set default value
        dot = new Dot(0, 0, 20);
        dot.setDotChangedListener(this);
    }
```

9. Update MainActivity

Separate random dot and display

```
public void randomDot(View view) {
    //Random a Dot
    Random random = new Random();
    dot.setCenterX(random.nextInt(200));
    dot.setCenterY(random.nextInt(200));
}

@Override
public void onDotChanged(Dot dot) {
    //Draw dot model to view
    TextView coordX = (TextView) findViewById(R.id.txtCoordX);
    TextView coordY = (TextView) findViewById(R.id.txtCoordY);
    coordX.setText(String.valueOf(dot.getCenterX()));
    coordY.setText(String.valueOf(dot.getCenterY()));
}
```


100%
12:10 AM
SimpleApp
150
43
RANDOM DOT

What’s next ?

Display dot on View

Build custom View

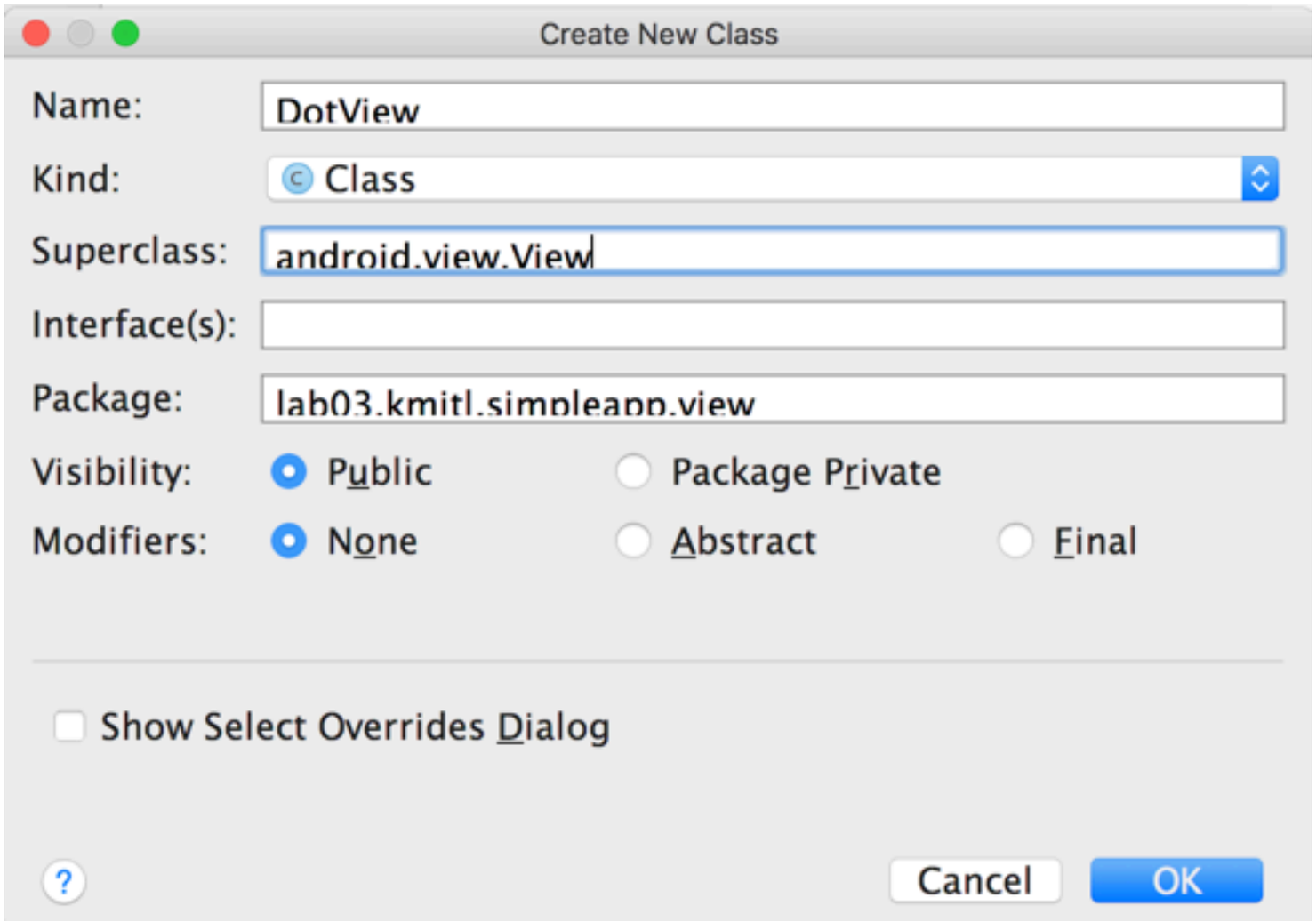
Build Custom View

Extends from android.view.View

Create new package = **view**

Create new class = **DotView**

1. Create class DotView.java



2. Add more constructor

```
private Paint paint;
public DotView(Context context) {
    super(context);
    paint = new Paint();
}

public DotView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
    paint = new Paint();
}

public DotView(Context context, AttributeSet attrs) {
    super(context, attrs);
    paint = new Paint();
}
```

3. Add onDraw()

```java
package lab03.kmitl.simpleapp.view;

import android.content.Context;
import android.graphics.Canvas;
import android.view.View;

public class DotView extends View {
    public DotView(Context context) {
        super(context);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
    }
}
```

4. Implement onDraw()

```
private Dot dot;

@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    if(dot != null) {
        paint.setColor(Color.RED);
        canvas.drawCircle(dot.getCenterX(),
                dot.getCenterY(),
                dot.getRadius(), paint);
    }
}

public void setDot(Dot dot) {
    this.dot = dot;
}
```

5. Update main layout to add view

```
<lab03.kmitl.simpleapp.view.DotView
    android:id="@+id/dotView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="10" />
```

6. Update MainActivity

Create instance of DotView

```
public class MainActivity extends AppCompatActivity
implements Dot.DotChangedListener{

    private Dot dot;
    private DotView dotView;

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

        // Set default value
        dot = new Dot(0, 0, 20);
        dot.setDotChangedListener(this);

        dotView = (DotView) findViewById(R.id.dotView);
    }
```

7. Update MainActivity

Draw a circle when data changed

```
public void randomDot(View view) {
    //Random a Dot
    Random random = new Random();
    dot.setCenterX(random.nextInt(200));
    dot.setCenterY(random.nextInt(200));
}
```

```
@Override
public void onDotChanged(Dot dot) {
    //Draw dot model to view
    dotView.setDot(dot);
    dotView.invalidate();
}
```

8. Remove hard code

Replace 200 with the actual size of View

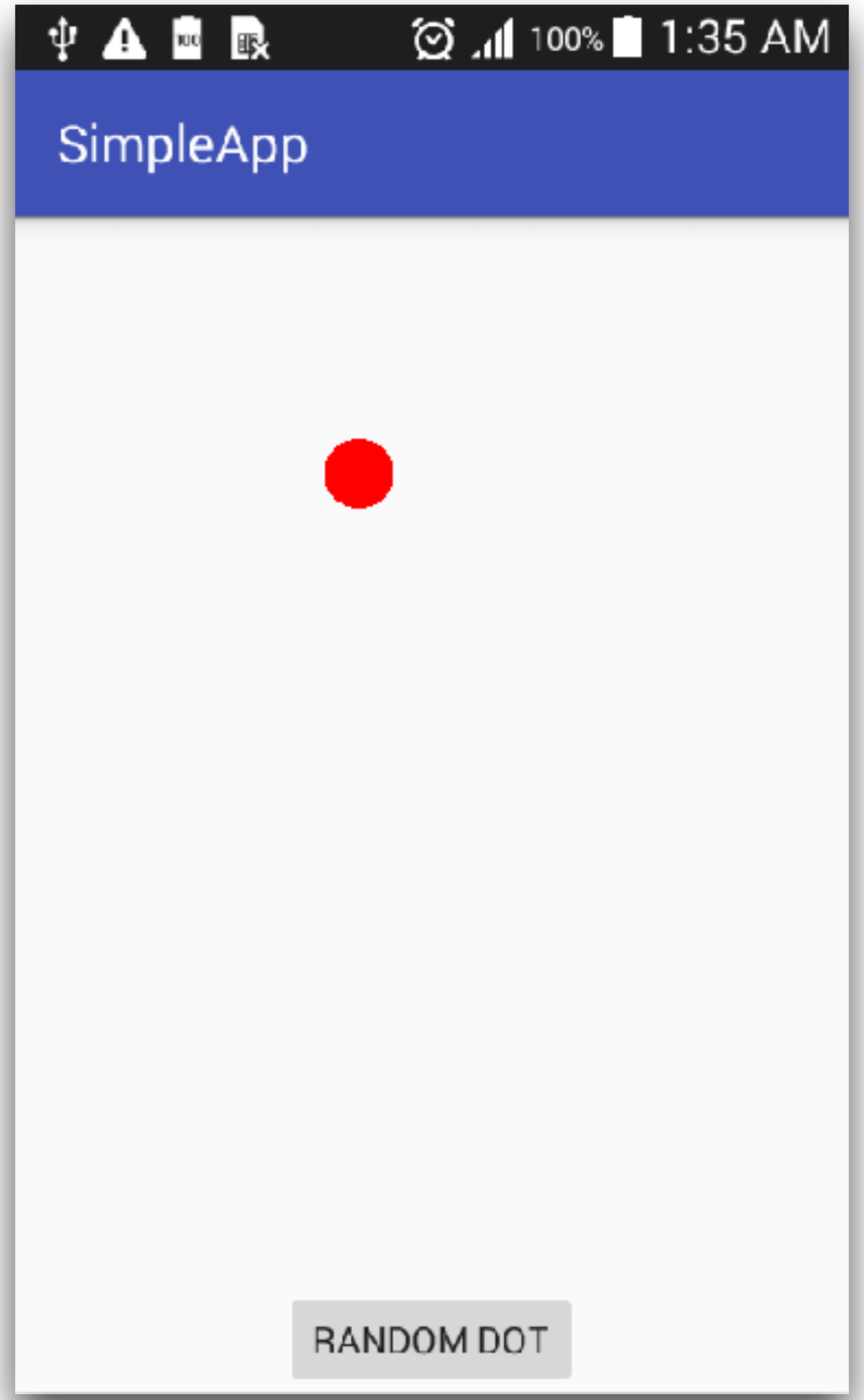
```
public void randomDot(View view) {
    //Random a Dot
    Random random = new Random();
    dot.setCenterX(random.nextInt(200));
    dot.setCenterY(random.nextInt(200));
}
```

```
@Override
public void onDotChanged(Dot dot) {
    //Draw dot model to view
    dotView.setDot(dot);
    dotView.invalidate();
}
```

8. Remove hard code

Replace 200 with the actual size of View

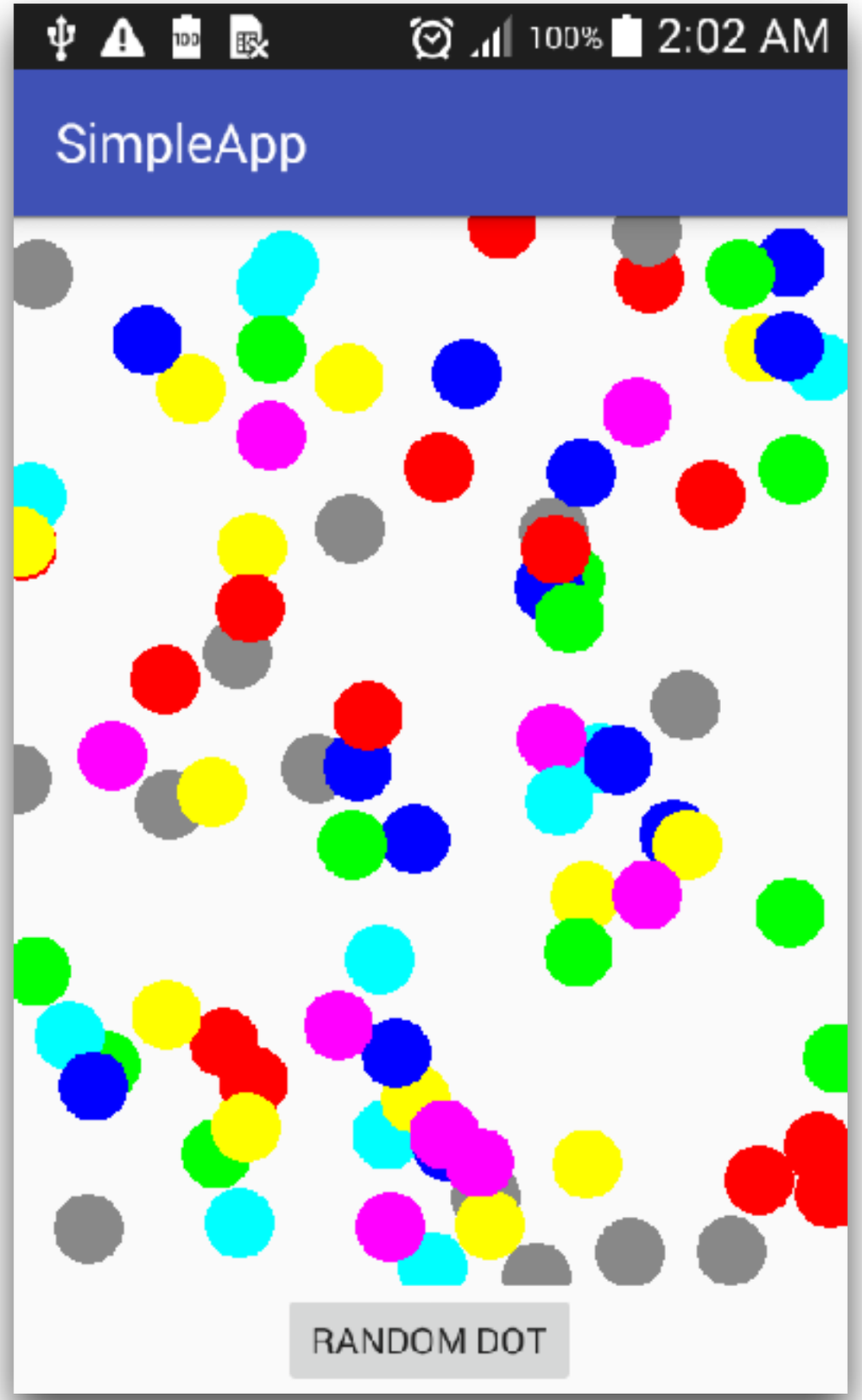
```
public void randomDot(View view) {
    Random random = new Random();
    int x = random.nextInt(this.dotView.getWidth());
    int y = random.nextInt(this.dotView.getHeight());
```


100%
1:35 AM
SimpleApp
RANDOM DOT

What’s next ?

Workshop

Need more dot !!
Random color of dot !!


100%
2:02 AM
SimpleApp
RANDOM DOT

What’s next ?

Homework

Edit dot
Delete dot
Clear all dot