COMPONENT LIBRARY FOR ANDROID APPLICATION DEVELOPMENT

Component is a user interface builder library that eliminates the tedious works involved in building UI components programmatically and the need to think about developing for different screen sizes and screen densities.

Using Component library, UI components can completely and easily be built programmatically without using layout resources (xml). Though you can use layout resources (xml) alongside building components programmatically using Component library, but it's not necessary. Sticking to building UI components programmatically using Component library alone will help simplify your work.

Component is primarily based on the formula below, which is enclosed in a utility method named ‘dimen’:

dimension = pixel * (densityDPI / DENSITY_DEFAULT)

Where,

pixel = value you give to the ‘dimen’ method that gets resolved for different screen densities and screen sizes.

densityDPI = density dots per inch of android device gotten from the display metrics of the device.

DENSITY_DEFAULT = constant from DisplayMetrics.

dimension = actual value the pixel value given to the formula will resolve into after calculation for different screen densities and screen sizes. This value is assigned to any space between two points in a component or space between two components which include length, width, height, breadth and (left, top, right and bottom of padding and margin) of components.

Note: You do not need to care about this formula or the ‘dimen’ method if you are using this library because all the works you will need to do with the method have already been done for you in the library. It’s only described here as the core of the library.

ADDING COMPONENT TO ANDROID PROJECT

Using Gradle;

Add the repository below to repositories block in your build.gradle (Project: <Project Name>) file:

```
allprojects {
        repositories {
                …
                maven {url ‘https://jitpack.io’}
        }
}
```

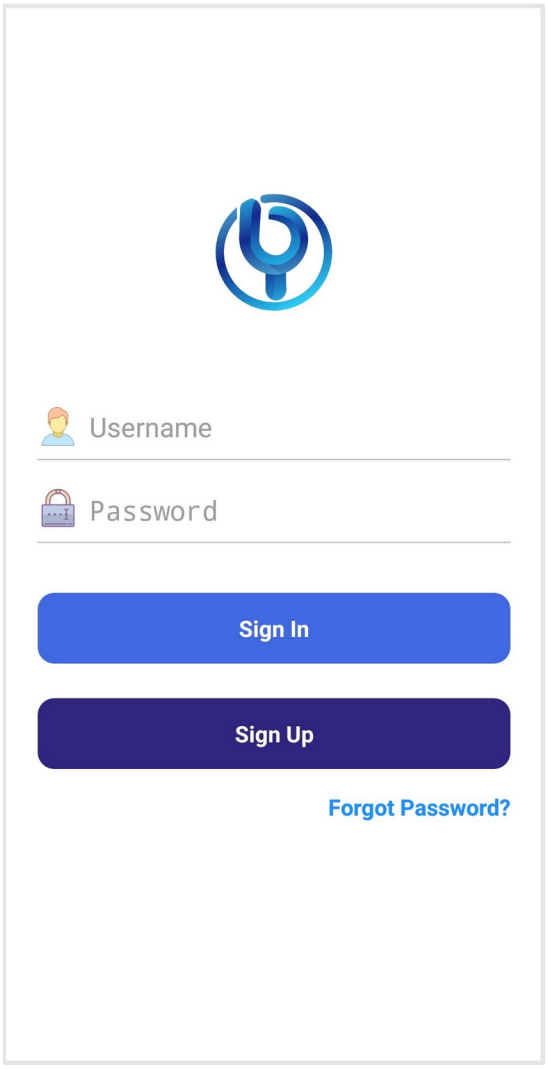
Add the dependency below to dependencies block in your build.gradle (Module: app) file:

```
dependencies {
        implementation 'com.github.chibuzoio:component:1.0.1'
}
```

Click on 'Sync Now' or go to File and click on 'Sync Project With Gradle Files'. The library will be added automatically into your project.

USING COMPONENT LIBRARY

Using the LoginActivity below for illustration:



The code below is responsible for the arrangement of the Activity above:

```
AU.disableDefaultActionBar(this);

VerticalLinearLayout activityContainer =
        new VerticalLinearLayout(this, GenericLayoutParams.MATCH_PARENT,
GenericLayoutParams.MATCH_PARENT);
activityContainer.setComponentColor(R.color.whiteColor);

setContentView(activityContainer);

ScrollViewComponent scrollViewComponent =
```

```
        new ScrollViewComponent(activityContainer);

VerticalLinearLayout mainLayoutContainer =
        new VerticalLinearLayout(this, scrollViewComponent,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.MATCH_PARENT);
mainLayoutContainer.setLayoutGravity(Gravity.CENTER);
mainLayoutContainer.setGravity(Gravity.CENTER);

ImageViewComponent companyLogo =
        new ImageViewComponent(mainLayoutContainer, R.drawable.ymcmart);
companyLogo.setCircularCenterImage(R.drawable.ymcmart);
companyLogo.setImageSize(77.777f);

BorderlessEditText usernameEditText =
        new BorderlessEditText(mainLayoutContainer,
                R.drawable.icon_user, "Username");
usernameEditText.setMargins(23, 55.555f, 23, 0);

BorderlessEditText passwordEditText =
        new BorderlessEditText(mainLayoutContainer,
                R.drawable.icon_password, "Password");
passwordEditText.getBorderlessEditTextView()
        .setEditorInputType(EditTextComponent.INPUT_TYPE_TEXT_PASSWORD);
passwordEditText.setMargins(23, 11.111f, 23, 0);

ButtonComponent loginButton =
        new ButtonComponent(mainLayoutContainer, "Sign In");
loginButton.setMargins(23, 33.333f, 23, 23);

ButtonComponent signUpButton =
        new ButtonComponent(mainLayoutContainer, "Sign Up");
signUpButton.setDrawable(AU.curveBackgroundCorner(this, 11.111f,
R.color.colorPrimaryDarker));
signUpButton.setMargins(23, 0, 23, 0);

TextViewComponent forgotPasswordTextLink =
        new TextViewComponent(mainLayoutContainer, "Forgot Password?",
5);
forgotPasswordTextLink.setMargins(23, 15.333f, 23, 33.333f);
forgotPasswordTextLink.setAlignment(TextViewComponent.TEXT_ALIGN_RIGHT);
forgotPasswordTextLink.setTextStyle(TextViewComponent.BOLD_TEXT);
forgotPasswordTextLink.setTextViewColor(R.color.genericLink);
```

The code snippet above was written in the onCreate method of LoginActivity, but can be isolated as a private method of the activity and called in the onCreate method. Description of the code snippet is as below:

disableDefaultActionBar of AU (Activity Utility) class removes the activity's default ActionBar.

activityContainer is the base component that holds all the components of LoginActivity in place. That's why it's set to render all other components of the LoginActivity by setContentView method of LoginActivity. The constructor of **activityContainer** object contains two other parameters which are constants of GenericLayoutParams class. **(Note: Always choose layout params constants (WRAP_CONTENT and MATCH_PARENT) from GenericLayoutParams and not from the LayoutParams class if you are using this library).**

Note: Every component takes layout component (like FrameLayout, VerticalLinearLayout, HorizontalLinearLayout, ScrollView and so on) as one of its parameters because the layout component holds the component in position except the first (or base) layout component that is set for the activity by the setContentView method of the activity.

GenericLayoutParams class takes care of layout management of components. You do not have to concern yourself with what is happening in this class if you are only using the library, unless you are contributing to the development of the library.

scrollViewComponent takes care of scrolling should in case the components of the LoginActivity grows past the screen size.

scrollViewComponent by convention can only contain one component. So, **mainLayoutContainer** is the only component contained by **scrollViewComponent** and it forms the base container for all other components contained by LoginActivity.

Note: Instead of setting orientation off from LinearLayout class, the LinearLayout got divided into two types of LinearLayout; HorizontalLinearLayout and VerticalLinearLayout.

Setting gravity and layout gravity using this library has been simplified into a single method call. E.g:

mainLayoutContainer.setGravity(Gravity.CENTER);
mainLayoutContainer.setLayoutGravity(Gravity.CENTER);

The two method calls above for setting gravity and layout gravity respectively are responsible for aligning the **mainLayoutContainer** which contains other components to the center of the **scrollViewComponent.**

From the **mainLayoutContainer** going down, the trend (the simplicity involved in creating UI components) continues downward because **the design goal of Component library is that every component should be designed as a single entity (as one class) throughout the entirety of the project and have its object created and used where ever it's needed.** Complex components are created by composing other simple or complex components into one component. Some examples of simple components in this library include: DrawerLayoutComponent, FrameLayoutComponent, ViewPagerComponent, EditTextComponent, ScrollViewComponent, TextViewComponent, and ViewComponent. These components are termed simple components because they inherit directly from the known Android UI classes which respectively are: DrawerLayout, FrameLayout, ViewPager, AppCompatEditText, ScrollView, AppCompatTextView and View. On the other hand, some examples of complex components include: HorizontalLinearComponent, VerticalLinearComponent, BorderlessEditText, ButtonComponent and FormFieldComponent. The first two complex components inherit directly from LinearLayoutComponent, while the last three inherit from VerticalLinearComponent. These complex components are also composed of other simple or complex components.

HOW TO DESIGN REUSABLE COMPONENTS

To ensure flexibility in the use of Component library, components have to be designed once and used wherever it is needed in your project. Using BorderlessEditText as example below, components are designed by inheriting from the outermost layout component (which is VerticalLinearLayout in the case of BorderlessEditText component), which forms the root View of

the reusable component. This outermost layout component adds children components, which includes other layout components (which in turn adds other components and the trend continues with respect to the complexity of the reusable component).

Analysis of BorderlessEditText component as an example:

The **BorderlessEditText** component above was designed from the combination of **HorizontalLinearLayout** and **VerticalLinearLayout.**

This component can also be designed from the combination of other types of layout components like **FrameLayout, RelativeLayout** or **ConstraintLayout** depending on what is most convenient for you**.** I chose **LinearLayout** component because it is the layout component most convenient for me.

Looking at the component above, only one **VerticalLinearComponent** and one **HorizontalLinearComponent** will be required to arrange the components in it as illustrated in the diagram below:

The outer orange layout is a **VerticalLinearLayout,** while the inner purple layout is a **HorizontalLinearLayout.** Since the outer layout, which is also the outermost layout of this component is a **VerticalLinearLayout,** the **BorderlessEditText** component will extend **VerticalLinearLayout** so that all the components in **BorderlessEditText** component will be added to the **VerticalLinearLayout,** which now forms the root view of **BorderlessEditText** component.

```
public class BorderlessEditText extends VerticalLinearLayout {

    public BorderlessEditText(ViewGroup viewGroup) {
        super(viewGroup.getContext(), viewGroup,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);
    }
}
```

BorderlessEditText calls the constructor of the super class, **VerticalLinearLayout** and gives it context from **ViewGroup, ViewGroup, MATCH_PARENT** and **WRAP_CONTENT** of **GenericLayoutParams** class. The last two parameters of the super constructor have it that **BorderlessEditText** component should inherit the width of the layout that will contain it and assume the height of the content(s) that has the highest height.

NOTE: MATCH_PARENT and WRAP_CONTENT constants must be gotten from GenericLayoutParams class.

NOTE: Child components of component classes are created in private methods of the classes.

Next to be created is a private method for the **HorizontalLinearLayout** that contains the icon and **EditTextComponent** enclosed in the purple rectangle as illustrated in the diagram above. Also, to be created is a public get method that gets the **HorizontalLinearLayout,** while adding a private field for **HorizontalLinearLayout.**

```
public class BorderlessEditText extends VerticalLinearLayout {
    private HorizontalLinearLayout borderlessEditTextLayout;

    public BorderlessEditText(ViewGroup viewGroup) {
        super(viewGroup.getContext(), viewGroup,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);

        setBorderlessEditTextLayout();
    }

    public HorizontalLinearLayout getBorderlessEditTextLayout() {
        return borderlessEditTextLayout;
    }

    private void setBorderlessEditTextLayout() {
        borderlessEditTextLayout =
           new HorizontalLinearLayout(getContext(), this,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);
        setEditTextIconView();
        setBorderlessEditTextView();
    }
}
```

Looking at **setBorderlessEditTextLayout()** method above, **setEditTextIconView()** and **setBorderlessEditTextView()** were called in it, with the later coming before the former. Thus, ImageView for the icon is first added before adding EditText view into **borderlessEditTextLayout.** The constructor of the **HorizontalLinearLayout** assigned to **bordelessEditTextLayout** in **setBorderlessEditTextLayout()** method takes **'this'** as one of its parameters because it uses it to add **borderlessEditTextLayout** to the root view of **BorderlessEditText** component. **BorderlessEditTextLayout** inherits the width of its parent **(VerticalLinearLayout) (MATCH_PARENT)** and assumes the maximum height formed by its contents **(WRAP_CONTENT)**. Finally, **setBorderlessEditTextLayout()** method is called in the constructor of **BorderlessEditText** component.

NOTE: setBorderlessEditTextLayout() method is private, while getBorderlessEditTextLayout() is public. This is so in order to ensure consistency in the use of the component by preventing the user of the component from disorganizing the component (BorderlessEditText) by adding more components to it.

Next to be created is **editTextUnderline** as illustrated in the green box below**:**


```
public class BorderlessEditText extends VerticalLinearLayout {
    private ViewComponent editTextUnderline;
    private HorizontalLinearLayout borderlessEditTextLayout;

    public BorderlessEditText(ViewGroup viewGroup) {
        super(viewGroup.getContext(), viewGroup,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);

        setBorderlessEditTextLayout();
        setEditTextUnderline();
    }

    public HorizontalLinearLayout getBorderlessEditTextLayout() {
        return borderlessEditTextLayout;
    }

    private void setBorderlessEditTextLayout() {
        borderlessEditTextLayout =
           new HorizontalLinearLayout(getContext(), this,
                GenericLayoutParams.MATCH_PARENT,
               GenericLayoutParams.WRAP_CONTENT);
        setEditTextIconView();
        setBorderlessEditTextView();
    }

    public ViewComponent getEditTextUnderline() {
        return editTextUnderline;
    }

    private void setEditTextUnderline() {
        editTextUnderline =
                new ViewComponent(this, R.color.faintLine, 1);
    }
}
```

As expected; the set method, **setEditTextUnderline()** is private to **BorderlessEditText** class, while the get method, **getEditTextUnderline()** is public for classes that instantiate **BorderlessEditText** class to access it. The constructor of the **ViewComponent** class in **setEditTextUnderline()** method takes **'this',** which adds **editTextUnderline** to the root view of **BorderlessEditText** component, the second parameter is the color of the **editTextUnderline** and 1 which is the last parameter of the **ViewComponent** constructor is the height of the **editTextUnderline.** Finally, **setEditTextUnderline()** method is added to the constructor of **BorderlessEditText** for creation**.**

The outer layout is filled. Now, let's look into the inner layout which encloses the icon and the EditText components.

From the diagram above, the icon and the EditText components are horizontally aligned. So, **HorizontalLinearLayout** component is required to arrange them in position. In this case, both the icon and the EditText components will be contained by **borderlessEditTextLayout** as expressed in the code below:

```
public class BorderlessEditText extends VerticalLinearLayout {
    private ViewComponent editTextUnderline;
    private ImageViewComponent editTextIconView;
    private EditTextComponent borderlessEditTextView;
    private HorizontalLinearLayout borderlessEditTextLayout;

    public BorderlessEditText(ViewGroup viewGroup,
           int editTextIcon, String editTextHint) {
        super(viewGroup.getContext(), viewGroup,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);

        this.editTextIcon = editTextIcon;
        this.editTextHint = editTextHint;

        setBorderlessEditTextLayout();
        setEditTextUnderline();
    }

    public HorizontalLinearLayout getBorderlessEditTextLayout() {
        return borderlessEditTextLayout;
    }

    private void setBorderlessEditTextLayout() {
        borderlessEditTextLayout =
                new HorizontalLinearLayout(getContext(), this,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);
        setEditTextIconView();
        setBorderlessEditTextView();
    }

    public ViewComponent getEditTextUnderline() {
        return editTextUnderline;
    }

    private void setEditTextUnderline() {
        editTextUnderline =
                new ViewComponent(this, R.color.faintLine, 1);
    }

    public ImageViewComponent getEditTextIconView() {
        return editTextIconView;
```

```
    }

    private void setEditTextIconView() {
        editTextIconView =
                new ImageViewComponent(borderlessEditTextLayout,
                     editTextIcon);
        editTextIconView.setLayoutGravity(Gravity.CENTER_VERTICAL);
        editTextIconView.setImageObject(editTextIcon);
        editTextIconView.setImageSize(27);
    }

    public EditTextComponent getBorderlessEditTextView() {
        return borderlessEditTextView;
    }

    private void setBorderlessEditTextView() {
        borderlessEditTextView =
                new EditTextComponent(borderlessEditTextLayout,
                     editTextHint);
        borderlessEditTextView.setMargins(5, 0, 0, 0);
        setEditTextBackgroundColor(R.color.whiteColor);
    }
}
```

In the code above, **setEditTextIconView(), getEditTextIconView(), setBorderlessEditTextView()** and **getBorderlessEditTextView()** methods were introduced to the **BorderlessEditText** component class, which alongside added private fields to **BorderlessEditText** component class and altered its constructor by adding two more parameters to it, which include: **editTextIcon** and **editTextHint.** In the constructor, both **editTextIcon** and **editTextHint** are set in order to be used by **setEditTextIconView()** and **setBorderlessEditTextView()** methods respectively. Observe that in these two set methods, **borderlessEditTextLayout** is given as the first parameter of the constructors of **ImageViewComponent** and **EditTextComponent** because both of them will be added to **borderlessEditTextLayout** which is a **HorizontalLinearLayout.** Finally, remember that these two set methods (**setEditTextIconView()** and **setBorderlessEditTextView()**) are called in **setBorderlessEditTextLayout()** method to make them available and visible to the **BorderlessEditText** component class as part of it.

To complete the development of **BorderlessEditText** component class, **getEditTextHint(), setEditTextHint(String editTextHint), getEditTextIcon(), setEditTextIcon(int editTextIcon), getEditTextBackgroundColor()** and **setEditTextBackgroundColor(int editTextBackgroundColor)** methods are required alongside their private fields respectively. Below is the complete code for the development of **BorderlessEditText** component:

```
public class BorderlessEditText extends VerticalLinearLayout {
    private String editTextHint;
    private ViewComponent editTextUnderline;
    private ImageViewComponent editTextIconView;
    private EditTextComponent borderlessEditTextView;
    private int editTextIcon, editTextBackgroundColor;
    private HorizontalLinearLayout borderlessEditTextLayout;

    public BorderlessEditText(ViewGroup viewGroup,
           int editTextIcon, String editTextHint) {
        super(viewGroup.getContext(), viewGroup,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);
```

```
        this.editTextIcon = editTextIcon;
        this.editTextHint = editTextHint;

        setBorderlessEditTextLayout();
        setEditTextUnderline();
    }

    public String getEditTextHint() {
        return editTextHint;
    }

    public void setEditTextHint(String editTextHint) {
        borderlessEditTextView.setHint(editTextHint);
        this.editTextHint = editTextHint;
    }

    public int getEditTextIcon() {
        return editTextIcon;
    }

    public void setEditTextIcon(int editTextIcon) {
        editTextIconView.setImageObject(editTextIcon);
        this.editTextIcon = editTextIcon;
    }

    public int getEditTextBackgroundColor() {
        return editTextBackgroundColor;
    }

    public void setEditTextBackgroundColor(int editTextBackgroundColor) {
       borderlessEditTextView.setComponentColor(editTextBackgroundColor);
        this.editTextBackgroundColor = editTextBackgroundColor;
    }

    public HorizontalLinearLayout getBorderlessEditTextLayout() {
        return borderlessEditTextLayout;
    }

    private void setBorderlessEditTextLayout() {
        borderlessEditTextLayout =
           new HorizontalLinearLayout(getContext(), this,
                GenericLayoutParams.MATCH_PARENT,
                GenericLayoutParams.WRAP_CONTENT);
        setEditTextIconView();
        setBorderlessEditTextView();
    }

    public ViewComponent getEditTextUnderline() {
        return editTextUnderline;
    }

    private void setEditTextUnderline() {
        editTextUnderline =
                new ViewComponent(this, R.color.faintLine, 1);
    }

    public ImageViewComponent getEditTextIconView() {
```

```
        return editTextIconView;
    }

    private void setEditTextIconView() {
        editTextIconView =
                new ImageViewComponent(borderlessEditTextLayout,
                     editTextIcon);
        editTextIconView.setLayoutGravity(Gravity.CENTER_VERTICAL);
        editTextIconView.setImageObject(editTextIcon);
        editTextIconView.setImageSize(27);
    }

    public EditTextComponent getBorderlessEditTextView() {
        return borderlessEditTextView;
    }

    private void setBorderlessEditTextView() {
        borderlessEditTextView =
                new EditTextComponent(borderlessEditTextLayout,
                     editTextHint);
        borderlessEditTextView.setMargins(5, 0, 0, 0);
        setEditTextBackgroundColor(R.color.whiteColor);
    }
}
```

Development of components for reuse using Component library is the design philosophy of Component library.

GenericLayoutParams Class

This class controls the arrangement of components horizontally and vertically relative to their parent layouts and contents.

Private Fields
ViewGroup.LayoutParams **layoutParams**

Constants
public static final int **ZERO_SPACE**
public static final int **MATCH_PARENT**
public static final int **WRAP_CONTENT**

As their names imply, **ZERO_SPACE** gives zero as the width or height of layout, **MATCH_PARENT** gives the width or height of layout the exact width or height of its parent layout, while **WRAP_CONTENT** gives the width or height of layout the exact width or height that will suitably wrap its contents.

Public Constructors
GenericLayoutParams(ViewGroup viewGroup**, int** horizontalParam**, int** verticalParam**)**

GenericLayoutParams constructor takes ViewGroup or an object of any class that extends ViewGroup directly or indirectly as its first parameter. The second and third parameters are any of the constants of GenericLayoutParams class (ZERO_SPACE, MATCH_PARENT or WRAP_CONTENT).

Public Methods
void **setLayoutMargin(View** view**, float** left**, float** top**, float** right**, float** bottom**)**
void **setLayoutGravity(View** view**, int** gravity**)**
ViewGroup.LayoutParams **getLayoutParams()**

All the methods of **GenericLayoutParams** class alter layout dimensions (parameters) of all the Views (components) given to them except **getLayoutParams** method, which returns the layout parameters.

Packages of need in Component library include:

- layoutcomponent
- viewcomponent
- utility

com.chibuzo.component.layoutcomponent package contains the layout component classes in this library which include:

- DrawerLayoutComponent
- FrameLayoutComponent
- HorizontalLayoutComponent
- LinearLayoutComponent
- RecyclerViewComponent
- RelativeLayoutComponent
- VerticalLinearLayout
- ViewPagerComponent

com.chibuzo.component.viewcomponent package contains the view component classes in this library which include:

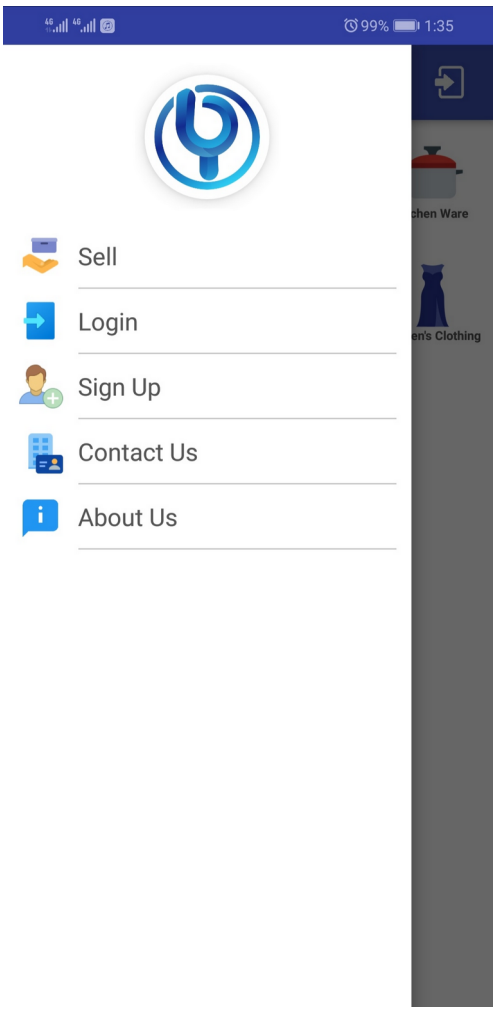
- BorderlessEditText
- ButtonComponent
- EditTextComponent
- FormFieldComponent
- IconLabelButton
- IconOnlyButton
- IconTextMenuComponent
- ImageViewComponent
- ImageViewParent
- ImageViewScreen
- ProgressBarComponent
- RoundElevatedPicture
- ScrollViewComponent
- SlideMenuComponent
- TextViewComponent
- ViewComponent

utility package contains the utility class which is:

- AU (short for Activity Utility)

Classes from the com.chibuzo.component.layoutcomponent package:

DrawerLayoutComponent Class



This is a layout component that allows components to be pulled out from left edge, right edge or left and right edges of the window.

DrawerLayoutComponent inherits from **androidx.drawerlayout.DrawerLayout**

Private Fields:
GenericLayoutParams genericLayoutParams

Constructors:
DrawerLayoutComponent(Context context**)**

Public Methods:
void **setComponentColor(int** color**)**
void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**

FrameLayoutComponent Class

This is a layout component designed to arrange its children one on top another facing the screen (in z-axis).

FrameLayoutComponent inherits from **android.widget.FrameLayout**

Private Fields:
float layoutWeight**;**
GenericLayoutParams genericLayoutParams**;**

Constructors:
FrameLayoutComponent(Context context**, int** horizontalParams**, int** verticalParams**)**
FrameLayoutComponent(Context context**, ViewGroup** viewGroup**, int** horizontalParam**, int** verticalParam**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
void **setComponentColor(int** color**)**
void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
void **setElevation(float** elevation**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(float** left**, float** top**, float** right**, float** bottom**)**
void **setMargins(float** left**, float** top**, float** right**, float** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**

HorizontalLinearLayout Class

This is a layout component designed to arrange its children horizontally from left to right.

HorizontalLinearLayout inherits from
com.chibuzo.component.layoutcomponent.LinearLayoutComponent

Constructors:
HorizontalLinearLayout(Context context**, int** horizontalParam**, int** verticalParam**)**
HorizontalLinearLayout(Context context**, ViewGroup** viewGroup**, int** horizontalParam**, int** verticalParam**)**

Public Methods:
See LinearLayoutComponent methods; they are inherited by HorizontalLinearLayout.

LinearLayoutComponent Class

This is an abstract linear layout component class extended by **HorizontalLayoutComponent** class and **VerticalLayoutComponent** class.

LinearLayoutComponent inherits from **android.widget.LinearLayout**

Private Fields:
float layoutWeight
GenericLayoutParams genericLayoutParams

Constructors:

LinearLayoutComponent(Context context**, ViewGroup** viewGroup**, int** horizontalParam**, int** verticalParam**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
void **setComponentColor(int** color**)**
void **setLayoutDimension(float** layoutWidth**, float** layoutHeight**)**
void **setLayoutWidth(float** layoutWidth**)**
void **setLayoutHeight(float** layoutHeight**)**
void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
void **setElevation(float** elevation**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(float** left**, float** top**, float** right**, float** bottom**)**
void **setMargins(float** left**, float** top**, float** right**, float** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**

RecyclerViewComponent Class

This layout class is used to display large sets of data in the user interface with small memory footprint.

RecyclerViewComponent inherits from **androidx.recyclerview.widget.RecyclerView**

Private Fields:
float layoutWeight
GenericLayoutParams genericLayoutParams

Constructors:
RecyclerViewComponent(ViewGroup viewGroup**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
void **setComponentColor(int** color**)**
void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
void **setElevation(float** elevation**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(int** left**, int** top**, int** right**, int** bottom**)**
void **setMargins(int** left**, int** top**, int** right**, int** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**

VerticalLinearLayout Class

This is a layout component designed to arrange its children vertically from top to bottom.

VerticalLinearLayout inherits from
com.chibuzo.component.layoutcomponent.LinearLayoutComponent

Constructors:
VerticalLinearLayout(Context context**, int** horizontalParam**, int** verticalParam**)**
VerticalLinearLayout(Context context**, ViewGroup** viewGroup**, int** horizontalParam**, int** verticalParam**)**

ViewPagerComponent Class

This is a layout component that allows user to flip left and right through pages of data.

ViewPagerComponent inherits from **androidx.viewpager.widget.ViewPager**

Private Fields:
float layoutWeight
GenericLayoutParams genericLayoutParams

Constructors:
ViewPagerComponent(Context context**, ViewGroup** viewGroup**, int** horizontalParam**, int** verticalParam**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
void **setElevation(float** elevation**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(float** left**, float** top**, float** right**, float** bottom**)**
void **setMargins(int** left**, int** top**, int** right**, int** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**

Classes from the com.chibuzo.component.viewcomponent package:

BorderlessEditText Class

This is a custom EditText that has icon, with no borders.

BorderlessEditText inherits from
com.chibuzo.component.layoutcomponent.VerticalLinearLayout

Private Fields:
int editTextIcon
String editTextHint
int editTextBackgroundColor
ViewComponent editTextUnderline
ImageViewComponent editTextIconView
EditTextComponent borderlessEditTextView

HorizontalLinearLayout borderlessEditTextLayout

Constructors:
BorderlessEditText(ViewGroup viewGroup**, int** editTextIcon**, String** editTextHint**)**

Public Methods:
String **getEditTextHint()**
void **setEditTextHint(String** editTextHint**)**
int **getEditTextIcon()**
void **setEditTextIcon(int** editTextIcon**)**
int **getEditTextBackgroundColor()**
void **setEditTextBackgroundColor(int** editTextBackgroundColor**)**
HorizontalLinearLayout **getBorderlessEditTextLayout()**
ViewComponent **getEditTextUnderline()**
ImageViewComponent **getEditTextIconView()**
EditTextComponent **getBorderlessEditTextView()**

Private Methods:
void **setBorderlessEditTextLayout()**
void **setEditTextUnderline()**
void **setEditTextIconView()**
void **setBorderlessEditTextView()**

ButtonComponent Class

A user interface component a user can tap to or click on to perform an action.

ButtonComponent inherits from
com.chibuzo.component.layoutcomponent.VerticalLinearLayout

Private Fields:
int labelSize
String buttonLabel
TextViewComponent textViewComponent

Constructors:
ButtonComponent(ViewGroup viewGroup**, String** buttonLabel**)**
ButtonComponent(ViewGroup viewGroup**, String** buttonLabel**, int** labelSize**)**

Public Methods:
TextViewComponent **getButtonLabel()**
void **setButtonLabel()**

EditTextComponent Class

This is a user interface component for entering and modifying text.

EditTextComponent inherits from **androidx.appcompat.widget.AppCompatEditText**

Private Fields:
float layoutWeight
int editorInputType

GenericLayoutParams genericLayoutParams

Public Constants:
static final int **INPUT_TYPE_TEXT**
static final int **INPUT_TYPE_NUMBER**
static final int **INPUT_TYPE_DATE_TIME**
static final int **INPUT_TYPE_PHONE**
static final int **INPUT_TYPE_TEXT_PASSWORD**
static final int **INPUT_TYPE_NUMBER_PASSWORD**

Constructors:
EditTextComponent(ViewGroup viewGroup, **String** hint**)**
EditTextComponent(ViewGroup viewGroup**, String** hint**, int** editorInputType**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
void **setComponentColor(int** color**)**
void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(float** left**, float** top**, float** right**, float** bottom**)**
void **setMargins(float** left**, float** top**, float** right**, float** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**
int **getEditorInputType()**
void **setEditorInputType(int** editorInputType**)**

FormFieldComponent Class

This is a form field or form input component that has an EditText component and label.

FormFieldComponent inherits from
com.chibuzo.component.layoutcomponent.VerticalLinearLayout

Private Fields:
int labelSize
String hint
String formLabel
int editorInputType
EditTextComponent editTextComponent
TextViewComponent textViewComponent

Constructors:
FormFieldComponent(ViewGroup viewGroup**, String** formLabel**, String** hint**)**
FormFieldComponent(ViewGroup viewGroup**, String** formLabel**, String** hint**, int** editorInputType**)**
FormFieldComponent(ViewGroup viewGroup**, String** formLabel**, int** labelSize**, String** hint**, int** editorInputType**)**

Public Methods:
EditTextComponent **getFormInput()**

void **setFormInput()**
TextViewComponent **getFormLabel()**
void **setFormLabel()**

IconLabelButton Class

This is a Button component that has both icon and text as label.

IconLabelButton inherits from
com.chibuzo.component.layoutcomponent.HorizontalLinearLayout

Private Fields:
int labelSize
Context context
TextViewComponent buttonLabel
ImageViewComponent buttonIcon

Constructors:
IconLabelButton(ViewGroup viewGroup**, int** drawableResource**, String** buttonLabel**)**
IconLabelButton(ViewGroup viewGroup**, Drawable** drawable**, String** buttonLabel**)**
IconLabelButton(ViewGroup viewGroup**, int** drawableResource**, String** buttonLabel**, int** labelSize**)**
IconLabelButton(ViewGroup viewGroup**, Drawable** drawable**, String** buttonLabel**, int** labelSize**)**

Public Methods:
TextViewComponent **getButtonLabel()**
void **setButtonLabel(String** buttonLabel**)**
void **setButtonLabel(TextViewComponent** buttonLabel**)**
ImageViewComponent **getButtonIcon()**
void **setButtonIcon(int** drawableResource**)**
void **setButtonIcon(Drawable** drawable**)**

IconOnlyButton Class

This is a Button component that has only icon in place of label.

IconOnlyButton inherits from
com.chibuzo.component.layoutcomponent.VerticalLinearLayout

Private Fields:
ImageViewComponent buttonIcon

Contructors:
IconOnlyButton(ViewGroup viewGroup**, int** drawableResource**)**
IconOnlyButton(ViewGroup viewGroup**, Drawable** drawable**)**

Public Methods:
ImageViewComponent **getButtonIcon()**
void **setButtonIcon(Drawable** drawable**)**
void **setButtonIcon(int** drawableResource**)**

IconTextMenuComponent Class

This is a menu item component that has both icon and label.

IconTextMenuComponent inherits from
com.chibuzo.component.layoutcomponent.VerticalLinearLayout

Private Fields:
Object menuIcon
String menuLabel
float menuIconSize
float menuLabelSize
ViewComponent separatorView
TextViewComponent menuLabelView
ImageViewComponent menuIconView
HorizontalLinearLayout parentContainerLayout

Contructors:
IconTextMenuComponent(ViewGroup viewGroup**, Object** menuIcon**, String** menuLabel**)**
IconTextMenuComponent(ViewGroup viewGroup**, Object** menuIcon**, String** menuLabel**, float** menuIconSize**)**
IconTextMenuComponent(ViewGroup viewGroup**, Object** menuIcon**, String** menuLabel**, float** menuIconSize**, float** menuLabelSize**)**

Public Methods:
Object **getMenuIcon()**
void **setMenuIcon(Object** menuIcon**)**
String **getMenuLabel()**
void **setMenuLabel(String** menuLabel**)**
float **getMenuLabelSize()**
void **setMenuLabelSize(int** menuLabelSize**)**
float **getMenuIconSize()**
void **setMenuIconSize(float** menuIconSize**)**
HorizontalLinearLayout **getParentContainerLayout()**
ImageViewComponent **getMenuIconView()**
TextViewComponent **getMenuLabelView()**
ViewComponent **getSeparatorView()**

Private Methods:
void **setParentContainerLayout()**
void **setMenuIconView()**
void **setMenuLabelView()**
void **setSeparatorView()**

ImageViewComponent Class

This component is used to display image resources.

ImageViewComponent inherits from
com.chibuzo.component.viewcomponent.ImageViewParent

Private Fields:

Bitmap bitmap

Constructors:
ImageViewComponent(ViewGroup viewGroup**, Object** imageObject**)**
ImageViewComponent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**)**
ImageViewComponent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** cornerRadius**)**
ImageViewComponent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** horizontalParam**, int** verticalParam**)**
ImageViewComponent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** cornerRadius**, int** horizontalParam**, int** verticalParam**)**

Public Methods:
String **loadDeviceStorageImage(int** requestCode**, int** resultCode**, Intent** intent**)**
void **setImagePlaceholder(int** placeholder**)**
void **setImageSize(float** allSides**)**
void **setImageSize(float** width**, float** height**)**
void **setRoundCornerPlaceholder(int** placeholder**)**

Private Methods:
String **processCurrentImage(Uri** uri**)**

ImageViewParent Class

This is the parent class of ImageView component classes in this library that are used to display image resources.

ImageViewParent inherits from **androidx.appcompat.widget.AppCompatImageView**

Private Fields:
float layoutWeight

Protected Fields:
Object object
int placeholder
int cornerRadius
GenericLayoutParams genericLayoutParams

Contructors:
ImageViewParent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**)**
ImageViewParent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** cornerRadius**)**
ImageViewParent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** horizontalParam**, int** verticalParam**)**
ImageViewParent(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** cornerRadius**, int** horizontalParam**, int** verticalParam**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
float **getActualImageWidth(Object** imageObject**)**
float **getActualImageHeight(Object** ImageObject**)**
void **setComponentColor(int** color**)**

void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
void **setWidthByDevice(int** rightMargin**)**
void **setHeightByDevice(int** height**)**
void **setImageWidth(float** imageWidth**)**
void **setImageHeight(float** imageHeight**)**
void **setRoundCornerImage(int** imageResource**)**
void **setRoundCornerImage(Uri** imageUri**)**
void **setImageObject(Object** object**)**
void **setCircularCenterImage(Integer** integer**)**
void **setCircularCenterImage(Object** imageObject**)**
void **setCircularCenterImage(String** string**)**
void **setCircularCenterImage(Drawable** drawable**)**
void **setCircularCenterImage(File** file**)**
void **setCircularCenterImage(Uri** uri**)**
void **setCircularCenterImage(byte[]** byteArray**)**
void **setCircularCenterImage(Bitmap** bitmap**)**
void **setRoundCornerCenterImage(Integer** integer**)**
void **setRoundCornerCenterImage(Object** imageObject**)**
void **setRoundCornerCenterImage(String** string**)**
void **setRoundCornerCenterImage(Drawable** drawable**)**
void **setRoundCornerCenterImage(File** file**)**
void **setRoundCornerCenterImage(Uri** uri**)**
void **setRoundCornerCenterImage(byte[]** byteArray**)**
void **setRoundCornerCenterImage(Bitmap** bitmap**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(float** left**, float** top**, float** right**, float** bottom**)**
void **setMargins(float** left**, float** top**, float** right**, float** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**

ImageViewScreen Class

This component is used to display image resources, while taking screen dimensions into consideration.

ImageViewScreen inherits from **com.chibuzo.component.viewcomponent.ImageViewParent**

Private Fields:
Bitmap bitmap
int densityPixel
int deviceDisplayWidth
int deviceDisplayHeight

Constructors:
ImageViewScreen(ViewGroup viewGroup**, Object** imageObject**)**
ImageViewScreen(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**)**
ImageViewScreen(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** cornerRadius**)**
ImageViewScreen(ViewGroup viewGroup**, Object** imageObject**, int** placeholder**, int** cornerRadius**, int** densityPixel**)**

Public Methods:
String **loadDeviceStorageImage(int** requestCode**, int** resultCode**, Intent** intent**)**
void **setImagePlaceholder(int** placeholder**)**
void **setRoundCornerPlaceholder(int** placeholder**)**

Private Methods:
String **processCurrentImage(Uri** uri**)**
void **setImageSize(int** placeholder**)**

RoundElevatedPicture Class

This is an ImageView component that crops image and displays it on a circular elevated surface.

RoundElevatedPicture inherits from
com.chibuzo.component.layoutcomponent.FrameLayoutComponent

Private Fields:
float imageSize
int paletteColor
int paletteElevation
Object imageObject
ImageViewComponent roundedPictureView
VerticalLinearLayout roundedPicturePalette

Constructors:
RoundElevatedPicture(ViewGroup viewGroup**, Object** imageObject**, float** imageSize**)**

Public Methods:
float **getImageSize()**
void **setImageSize(float** imageSize**)**
int **getPaletteColor()**
void **setPaletteColor(int** paletteColor**)**
Object **getImageObject()**
void **setImageObject(Object** imageObject**)**
int **getPaletteElevation()**
void **setPaletteElevation(int** paletteElevation**)**
void **setPaletteMargin(float** allSides**)**
void **setPalettePadding(float** allSides**)**

VerticalLinearLayout **getRoundedPicturePalette()**
ImageViewComponent **getRoundedPictureView()**

Private Methods:
void **setRoundedPicturePalette()**
void **setRoundedPictureView()**

ScrollViewComponent Class

This layout component adds scrolling ability to contents that are larger than the size of the containing layout component such as LinearLayout, FrameLayout, RelativeLayout, e.t.c.

ScrollViewComponent inherits from **android.widget.ScrollView**

Private Fields:
float layoutWeight
GenericLayoutParams genericLayoutParams

Constructors:
ScrollViewComponent(ViewGroup viewGroup**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(float** left**, float** top**, float** right**, float** bottom**)**
void **setMargins(float** left**, float** top**, float** right**, float** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**

SlideMenuComponent Class

This component is a custom DrawerLayoutComponent that has toolbar layout and left slide menu layout.

SlideMenuComponent inherits from
com.chibuzo.component.layoutcomponent.DrawerLayoutComponent

Private Fields:
HorizontalLinearLayout toolbarLayout
VerticalLinearLayout slideMenuLayout
VerticalLinearLayout parentContainerLayout

Constructors:
SlideMenuComponent(Context context**)**

Public Methods:
VerticalLinearLayout **getParentContainerLayout()**
void **setParentContainerLayout(VerticalLinearLayout** parentContainerLayout**)**
void **setParentContainerLayout()**
HorizontalLinearLayout **getToolbarLayout()**
void **setToolbarLayout(HorizontalLinearLayout** toolbarLayout**)**

void **setToolbarLayout()**
VerticalLinearLayout **getSlideMenuLayout()**
void **setSlideMenuLayout(VerticalLinearLayout** slideMenuLayout**)**
void **setSlideMenuLayout()**

TextViewComponent Class

This is a user interface component that displays text to the user.

TextViewComponent inherits from **androidx.appcompat.widget.AppCompatTextView**

Private Fields:
int alignment
int textViewColor
float layoutWeight
GenericLayoutParams genericLayoutParams

Public Constants:
static final int **BOLD_TEXT**
static final int **NORMAL_TEXT**
static final int **TEXT_ALIGN_LEFT**
static final int **TEXT_ALIGN_RIGHT**
static final int **TEXT_ALIGN_CENTER**

Constructors:
TextViewComponent(ViewGroup viewGroup**, String** text**, float** textSize**)**
TextViewComponent(ViewGroup viewGroup**, String** text**, float** textSize**, int** textStyle**)**
TextViewComponent(ViewGroup viewGroup**, String** text**, float** textSize**, int** textStyle**, int** alignment**)**

Public Methods:
int **getTextViewColor()**
void **setTextViewColor(int** textViewColor**)**
void **setLayoutGravity(int** gravity**)**
void **setComponentColor(int** color**)**
void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setPadding(float** left**, float** top**, float** right**, float** bottom**)**
void **setMargins(float** left**, float** top**, float** right**, float** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**
int **getAlignment()**
void **setTextStyle(int** textStyle**)**
void **setAlignment(int** alignment**)**

ViewComponent Class

ViewComponent inherits from **android.view.View** which is the base class for widgets, which are used to create interactive UI components (buttons, text fields, text input, e.t.c.).

Private Fields:
float layoutWeight
int componentColor
float componentWidth
float componentHeight
GenericLayoutParams genericLayoutParams

Constructors:
ViewComponent(ViewGroup viewGroup**, int** componentColor**, float** componentHeight**)**

Public Methods:
void **setLayoutGravity(int** gravity**)**
int **getComponentColor()**
void **setComponentColor(int** componentColor**)**
float **getComponentWidth()**
void **setComponentWidth(float** componentWidth**)**
float **getComponentHeight()**
void **setComponentHeight(float** componentHeight**)**
void **setBackground(int** background**)**
void **setDrawable(Drawable** drawable**)**
float **getLayoutWeight()**
void **setLayoutWeight(float** layoutWeight**)**
void **setMargins(float** left**, float** top**, float** right**, float** bottom**)**
GenericLayoutParams **getGenericLayoutParams()**
void **setGenericLayoutParams(GenericLayoutParams** genericLayoutParams**)**