



Asignatura: Application Development for Mobile Devices.
Tema: La clase Material Design.

I. Introducción.

Las características principales de **Material Design** son las siguientes.

- Es un lenguaje visual de diseño común, que posee sus propias normas para casi todos los detalles y se mantienen independientemente del tamaño de pantalla, donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal.
- Incluye una renovación de la tipografía Roboto para ésta que pueda adaptarse correctamente a todas las plataformas.
- Es amplio, gráfico e intuitivo. Los elementos fundamentales de diseño se basan en tipografía de imprenta, mallas, espacio, escala, color y uso de imágenes.
- Los tratamientos visuales se hacen más agradable a la vista. Las opciones adecuadas de color, imágenes de borde a borde, tipografía a gran escala y el espacio, crean una interfaz gráfica amplia para sumergir al usuario en la experiencia.
- Con el movimiento se respeta y refuerza al usuario como el motor principal.
- El diseño de material proporciona un conjunto de propiedades para personalizar el tema de diseño de materiales en color.



Figura 1. Opciones de diseño con MaterialDesign.

En el siguiente ejercicio se diseña un tema personalizado e implanta el control de navegación utilizando el RecyclerView. Se utilizan cinco atributos principales para personalizar el tema general.

II. Personalización del color con Material Design.

Material Design proporciona un conjunto de propiedades para personalizar el tema del Material Design Color. Se utilizan cinco atributos principales para personalizar el tema general.

colorPrimaryDark	Es el color principal más oscuro de la aplicación y se aplica principalmente al fondo de la barra de notificación.
colorPrimary	Es el color principal de la aplicación. Se aplicará como fondo barra de herramientas.
textColorPrimary	Es el color principal del texto. Se aplica a la barra de título.
windowBackground	Es el color de fondo predeterminado de la aplicación.
navigationBarColor	Es el que define el color de fondo de la barra de navegación del pie de página.

En la siguiente figura se indican las principales zonas donde se aplican los cambios:

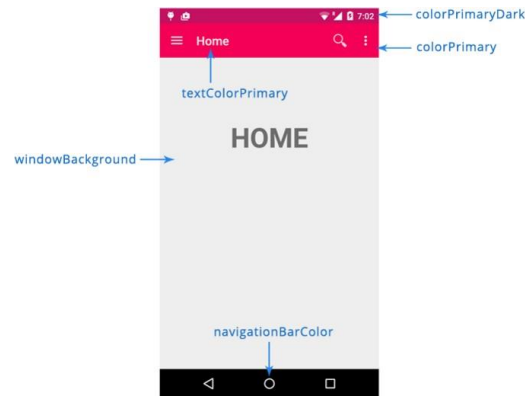


Figura 2. Las diferentes zonas de cambios con MaterialDesign.

III. Creación del tema Material Design.

1. En Android Studio, seleccionar File ->New Project y llenar todos los detalles necesarios para crear un nuevo proyecto. Cuando se solicite la selección de una actividad predeterminada, seleccionar Blank Activity y continuar.
2. Abrir el archivo res->values->strings.xml y a continuación añadir los valores de string:

```
<resources>
    <string name="app_name">Material Design</string>
    <string name="action_settings">Settings</string>
    <string name="action_search">Search</string>
    <string name="drawer_opn">Open</string>
    <string name="drawer_close">Close</string>

    <string name="nav_item_home">Home</string>
    <string name="nav_item_friends">Friends</string>
    <string name="nav_item_notifications">Messages</string>

    <!-- navigation drawer item labels -->
    <string-array name="nav_drawer_labels">
        <item>@string/nav_item_home</item>
        <item>@string/nav_item_friends</item>
        <item>@string/nav_item_notifications</item>
    </string-array>

    <string name="title_messages">Messages</string>
    <string name="title_friends">Friends</string>
    <string name="title_home">Home</string>
</resources>
```

3. Abrir el archivo res->values->colors.xml y añadir los siguientes valores del color. Si no se localiza el archivo colors.xml, crear un archivo con ese nombre en la ruta mencionada:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#F50057</color>
    <color name="colorPrimaryDark">#C51162</color>
    <color name="textColorPrimary">#FFFFFF</color>
    <color name="windowBackground">#FFFFFF</color>
    <color name="navigationBarColor">#000000</color>
    <color name="colorAccent">#FF80AB</color>
</resources>
```

4. Abrir el archivo res->values->dimens.xml y añadir las siguientes dimensiones:

```
<resources>
```



```

<!-- Default screen margins, per the Android Design guidelines. -->
<dimen name="activity_horizontal_margin">16dp</dimen>
<dimen name="activity_vertical_margin">16dp</dimen>
<dimen name="nav_drawer_width">260dp</dimen>
</resources>

```

5. Abrir el archivo `res->values->styles.xml` y agregar los siguientes estilos. Los estilos definidos en este archivo son los más comunes en todas las versiones de Android. En este caso se nombra el tema con `MyMaterialTheme`:

```

<resources>
<style name="MyMaterialTheme" parent="MyMaterialTheme.Base">
</style>
<style name="MyMaterialTheme.Base" parent="Theme.AppCompat.Light.DarkActionBar">
<item name="windowNoTitle">true</item>
<item name="windowActionBar">false</item>
<item name="colorPrimary">@color/colorPrimary</item>
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="colorAccent">@color/colorAccent</item>
</style>
</resources>

```

6. Debajo de la carpeta `res`, crear una carpeta `values-v21`. Dentro de la carpeta `values-21`, crear otro archivo `styles.xml` con los siguientes estilos. Estos estilos son particulares solamente para Android Lollipop:

```

<resources>
<style name="MyMaterialTheme" parent="MyMaterialTheme.Base">
<item name="android:windowContentTransitions">true</item>
<item name="android:windowAllowEnterTransitionOverlap">true</item>
<item name="android:windowAllowReturnTransitionOverlap">true</item>
<item
name="android:windowSharedElementEnterTransition">@android:transition/move</item>
<item
name="android:windowSharedElementExitTransition">@android:transition/move</item>
</style>
</resources>

```

7. Ahora, ya se encuentran listos los estilos básicos de Material Design. Para aplicar el tema, abrir el archivo `AndroidManifest.xml` y modificar el atributo `android:theme` de la etiqueta `<application>`:
- ```

android:theme="@style/MyMaterialTheme"

```

Por lo anterior, después de aplicar el tema, el contenido del `AndroidManifest.xml` debe ser similar al siguiente:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="info.androidhive.materialdesign" >

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/MyMaterialTheme" >
<activity
android:name=".activity.MainActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

```



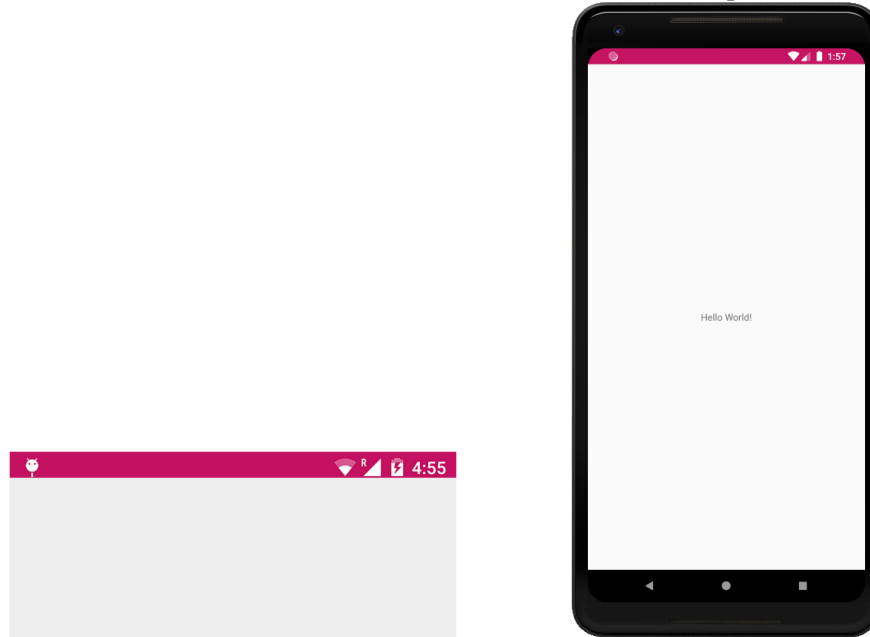
```

 </intent-filter>
 </activity>
</application>

</manifest>

```

Ahora, al ejecutar la aplicación, la barra de notificaciones muestra el cambio de color que se mencionó en los estilos.



**Figura 3.** El cambio de color con estilo.

## Añadiendo el Toolbar (ActionBar).

Crear una plantilla independiente para el Toolbar e incluirla en otra plantilla, en donde se desee que el Toolbar se muestre.

8. Crear el archivo `res->layout->toolbar.xml` y añadir el siguiente elemento `android.support.v7.widget.Toolbar`. Lo anterior crea una barra de herramientas con un tema y altura específicos:

```

<?xml version="1.0" encoding="utf-8"?>

<android.support.v7.widget.Toolbar
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:local="http://schemas.android.com/apk/res-auto"
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:minHeight="?attr/actionBarSize"
android:background="?attr/colorPrimary"
local:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
local:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

```

9. Abrir el archivo de la plantilla principal `activity_main.xml` y añadir la barra de herramientas utilizando la etiqueta

```

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

```

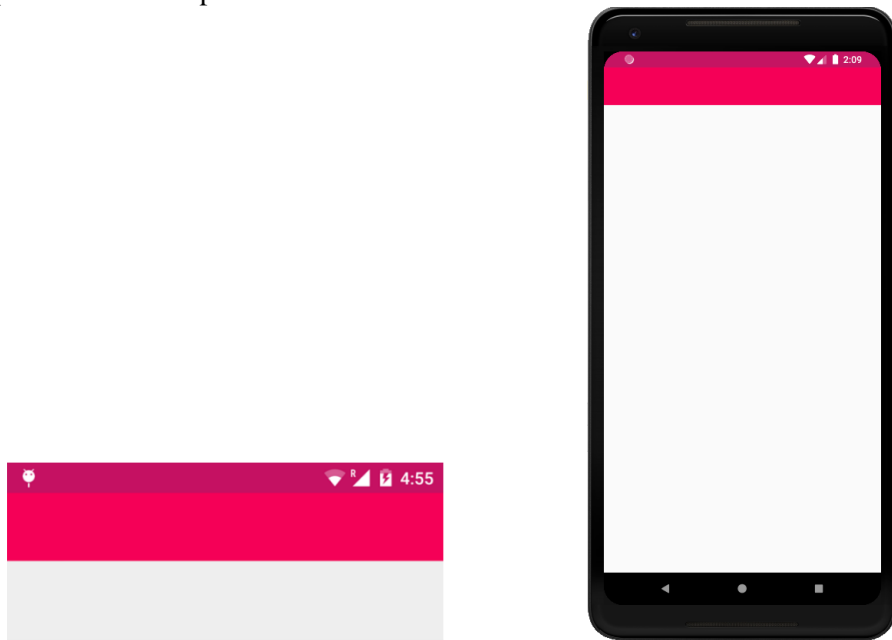


```
<LinearLayout
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:orientation="vertical">

 <include
 android:id="@+id/toolbar"
 layout="@layout/toolbar" />
</LinearLayout>


</RelativeLayout>
```

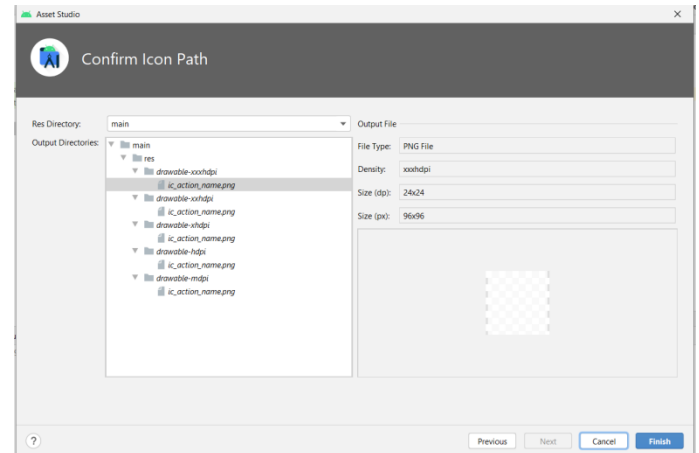
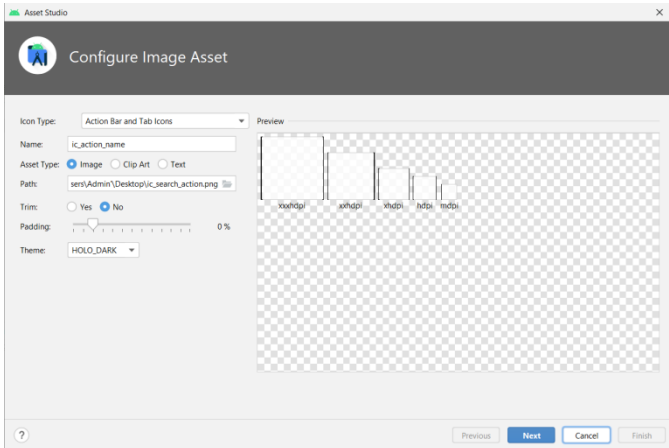
Ejecutar la aplicación para verificar la apariencia de la barra de herramientas:



**Figura 4.** La barra de herramientas.

Enseguida se añadirá un título a la barra de herramientas y se habilitarán las acciones de sus elementos.

10. Utilizar un ícono de búsqueda, por ejemplo , e importarlo al Android Studio como un Image Asset.
11. Para importar el Image Asset al Android Studio, clic derecho en **res->New->ImageAsset**. Se muestra una ventana de diálogo para importar el recurso. Localizar el icono de búsqueda que se descargó en el paso anterior, seleccionar **Action Bar and Tab Icons** en el **Asset Type** y asignarle el nombre de recurso con **ic\_action\_search\_** para continuar:



**Figura 5.** Configuración de Image Asset y confirmación de Icon Path.

12. Después de importar el ícono, abrir el archivo `res->menu->menu_main.xml` y añadir el elemento de menú, como se indica enseguida:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 tools:context=".MainActivity">

 <item
 android:id="@+id/action_search"
 android:title="@string/action_search"
 android:orderInCategory="100"
 android:icon="@drawable/ic_action_search"
 app:showAsAction="ifRoom" />

 <item
 android:id="@+id/action_settings"
 android:title="@string/action_settings"
 android:orderInCategory="100"
 app:showAsAction="never" />
</menu>
```

13. Ahora, abrir el archivo `MainActivity.java` y realizar los siguientes cambios:

- Heredar la actividad de `AppCompatActivity`.
- Activar la barra de herramientas, invocando al método `setSupportActionBar()` pasando el objeto de la barra de herramientas.
- Sobrescribir los métodos `onOptionsItemSelected()` y `onOptionsItemSelected()` para activar las acciones de los elementos de la barra de herramientas:

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

 private Toolbar mToolbar;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
```



```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

mToolbar = (Toolbar) findViewById(R.id.toolbar);

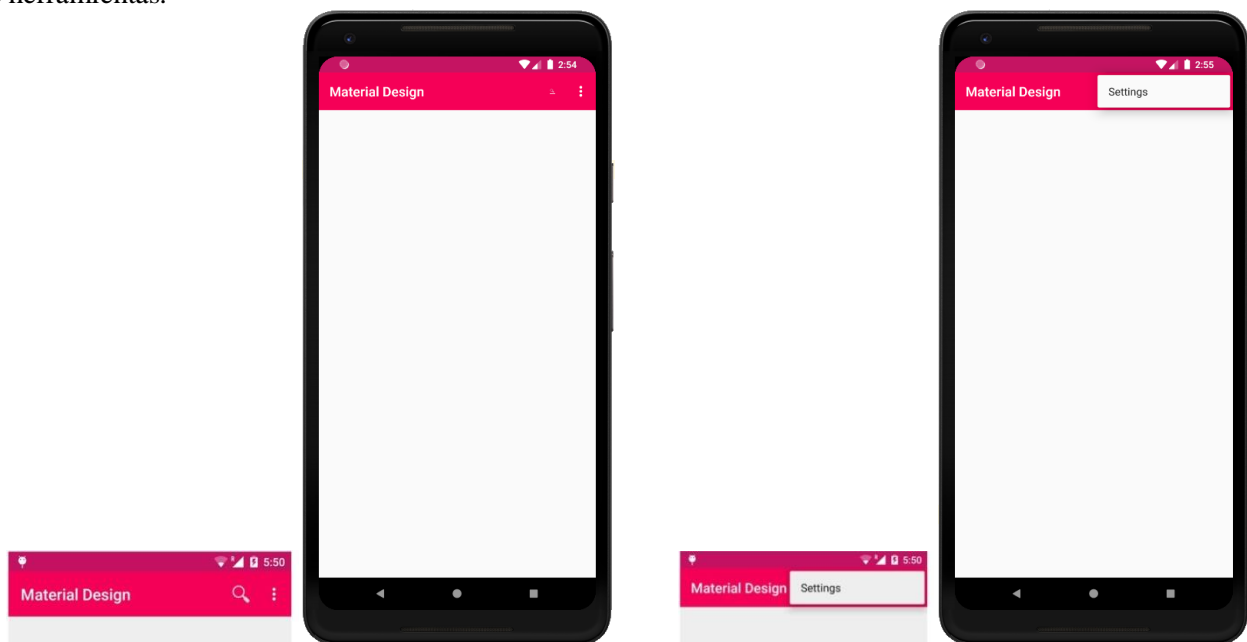
setSupportActionBar(mToolbar);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.menu_main, menu);
 return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
 // Handle action bar item clicks here. The action bar will
 // automatically handle clicks on the Home/Up button, so long
 // as you specify a parent activity in AndroidManifest.xml.
 int id = item.getItemId();

 //noinspection SimplifiableIfStatement
 if (id == R.id.action_settings) {
 return true;
 }

 return super.onOptionsItemSelected(item);
}
}
```

Después de aplicar los cambios anteriores, ejecutar la aplicación y verificar que se muestran los dos íconos de acción de la barra de herramientas.



**Figura 6.** Los iconos de acción de la barra de herramientas.

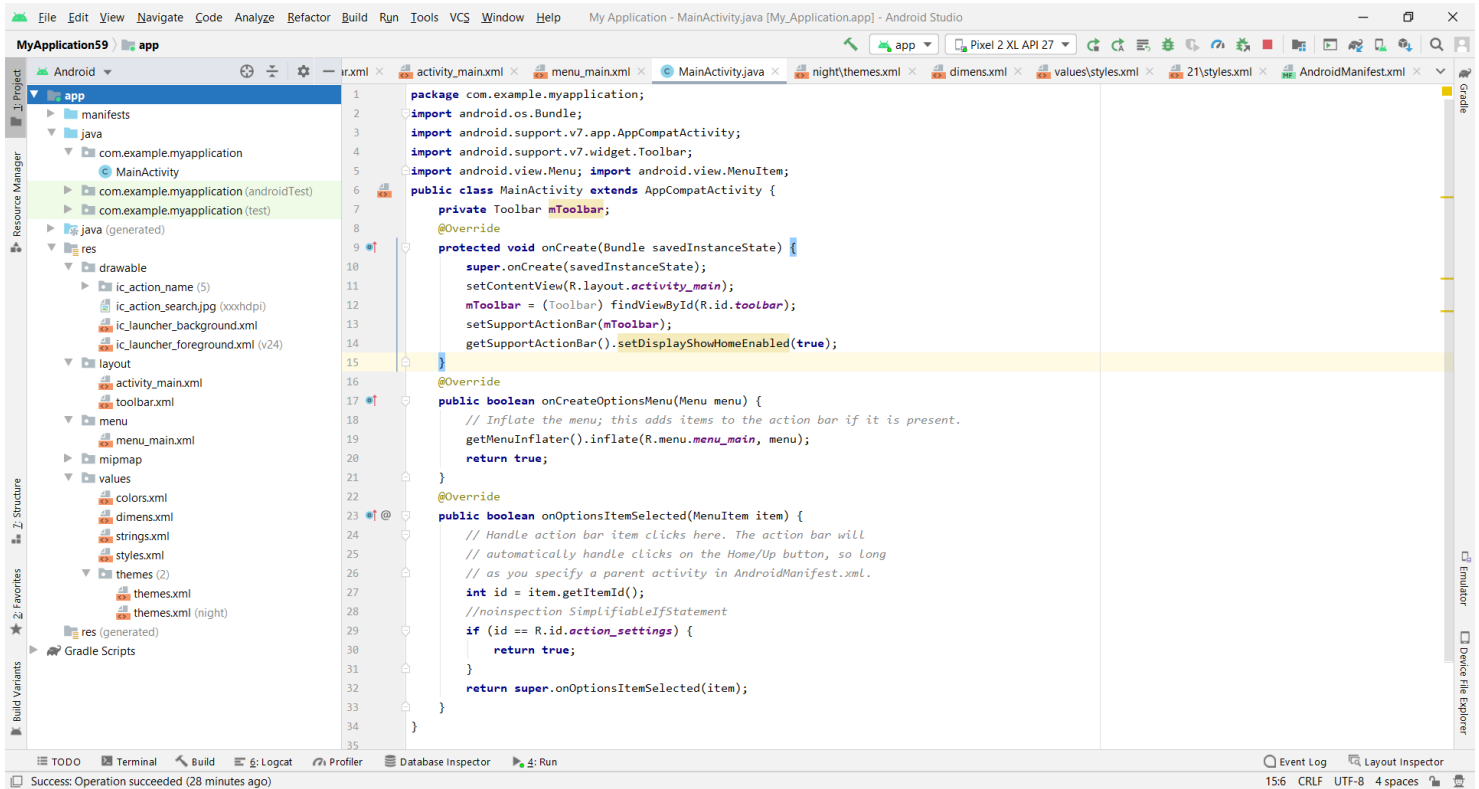


Figura 7. Estructura del proyecto hasta el paso 13.

## Añadiendo el Navigation Drawer.

En lugar de utilizar ListView para los elementos del menú, se utiliza RecyclerView en el Material Design. El RecyclerView es un componente flexible que permite colocar una ventana dentro de un conjunto mayor de datos.

14. En la carpeta java del proyecto, crear tres paquetes: **activity**, **adapter** y **model** y mover el archivo MainActivity.java al paquete de actividades. Lo anterior mantiene organizado al proyecto.

15. Abrir el archivo build.gradle localizado bajo el módulo app y agregar las dependencias siguientes. Después de agregar las dependencias, seleccionar **Build->Rebuild Project** para descargar las bibliotecas necesarias:

```
dependencies {
 compile fileTree(dir: 'libs', include: ['*.jar'])
 compile 'com.android.support:appcompat-v7:22.2.0'
 compile 'com.android.support:recyclerview-v7:22.2.+'
}
```

16. Debajo del paquete **model**, crear la clase NavDrawerItem.java con el código siguiente. Este modelo de clase es una clase POJO que define cada fila en el menú del NavigationDrawer. Revisar la inclusión de paquete del proyecto al inicio del código, por ejemplo **com.example.escom ...** etcétera:

```
public class NavDrawerItem {
 private boolean showNotify;
 private String title;

 public NavDrawerItem() {

 }

 public NavDrawerItem(boolean showNotify, String title) {
```





```

 this.showNotify = showNotify;
 this.title = title;
 }

 public boolean isShowNotify() {
 return showNotify;
 }

 public void setShowNotify(boolean showNotify) {
 this.showNotify = showNotify;
 }

 public String getTitle() {
 return title;
 }

 public void setTitle(String title) {
 this.title = title;
 }
}

```

17. Debajo de **res->layout**, crear un archivo de plantilla **nav\_drawer\_row.xml** y añadir el código siguiente. La plantilla incluye cada fila en el menú del **NavigationDrawer**. Si se desea personalizar el elemento del menú del **NavigationDrawer**, los cambios se deben realizar en este archivo. En este caso, solamente se posee un **TextView**:


```

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

 <TextView
 android:id="@+id/title"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:paddingLeft="30dp"
 android:paddingTop="10dp"
 android:paddingBottom="10dp"
 android:textSize="15dp"
 android:textStyle="bold" />

</RelativeLayout>

```

18. Descargar un icono del perfil, por ejemplo  y pegarlo en la carpeta **drawable**. Este paso es opcional, pero este ícono se utiliza en el encabezado del **NavigationDrawer**.

19. Crear otro archivo de plantilla **fragment\_navigation\_drawer.xml** y añadir el código siguiente. Esta plantilla incluye al componente completo del **NavigationDrawer**; además, contiene una sección del encabezado para mostrar la información del usuario y un **RecyclerView** para mostrar la lista.

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:background="@android:color/white">

```



```
<RelativeLayout
 android:id="@+id/nav_header_container"
 android:layout_width="match_parent"
 android:layout_height="140dp"
 android:layout_alignParentTop="true"
 android:background="@color/colorPrimary">

 <ImageView
 android:layout_width="70dp"
 android:layout_height="70dp"
 android:src="@drawable/ic_profile"
 android:scaleType="fitCenter"
 android:layout_centerInParent="true" />

</RelativeLayout>

<android.support.v7.widget.RecyclerView
 android:id="@+id/drawerList"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_below="@id/nav_header_container"
 android:layout_marginTop="15dp" />

</RelativeLayout>
```

20. Cuando se personaliza el RecyclerView, se necesita una clase adaptadora que incluya la típica plantilla xml. Para ello, debajo del paquete adaptador, crear la clase NavigationDrawerAdapter.java y añadir el código siguiente. Esta clase adaptadora infla (expande) el archivo nav\_drawer\_row.xml e incluye al menú del drawer RecyclerView.

```
import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.Collections;
import java.util.List;

public class NavigationDrawerAdapter extends
RecyclerView.Adapter<NavigationDrawerAdapter.MyViewHolder> {
 List<NavDrawerItem> data = Collections.emptyList();
 private LayoutInflater inflater;
 private Context context;

 public NavigationDrawerAdapter(Context context, List<NavDrawerItem> data) {
 this.context = context;
 inflater = LayoutInflater.from(context);
 this.data = data;
 }

 public void delete(int position) {
 data.remove(position);
 }
}
```



```
 notifyItemRemoved(position);
 }

 @Override
 public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
 View view = inflater.inflate(R.layout.nav_drawer_row, parent, false);
 MyViewHolder holder = new MyViewHolder(view);
 return holder;
 }

 @Override
 public void onBindViewHolder(MyViewHolder holder, int position) {
 NavDrawerItem current = data.get(position);
 holder.title.setText(current.getTitle());
 }

 @Override
 public int getItemCount() {
 return data.size();
 }
}

class MyViewHolder extends RecyclerView.ViewHolder {
 TextView title;

 public MyViewHolder(View itemView) {
 super(itemView);
 title = (TextView) itemView.findViewById(R.id.title);
 }
}
```

21. En el paquete **activity**, crear el archivo para un fragmento **FragmentDrawer.java**. Para crear un nuevo fragmento, clic derecho en **activity->New->Fragment->Fragment(Blank)** y asignar el nombre a la clase del fragmento:

```
import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.GestureDetector;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;
import java.util.List;

import info.androidhive.materialdesign.R;
import info.androidhive.materialdesign.adapter.NavigationDrawerAdapter;
import info.androidhive.materialdesign.model.NavDrawerItem;

public class FragmentDrawer extends Fragment {

 private static String TAG = FragmentDrawer.class.getSimpleName();
```



```
private RecyclerView recyclerView;
private ActionBarDrawerToggle mDrawerToggle;
private DrawerLayout mDrawerLayout;
private NavigationDrawerAdapter adapter;
private View containerView;
private static String[] titles = null;
private FragmentDrawerListener drawerListener;

public FragmentDrawer() {

}

public void setDrawerListener(FragmentDrawerListener listener) {
 this.drawerListener = listener;
}

public static List<NavDrawerItem> getData() {
 List<NavDrawerItem> data = new ArrayList<>();

 // preparing navigation drawer items
 for (int i = 0; i < titles.length; i++) {
 NavDrawerItem navItem = new NavDrawerItem();
 navItem.setTitle(titles[i]);
 data.add(navItem);
 }
 return data;
}

@Override
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

 // drawer labels
 titles = getActivity().getResources().getStringArray(R.array.nav_drawer_labels);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState) {
 // Inflating view layout
 View layout = inflater.inflate(R.layout.fragment_navigation_drawer, container,
 false);
 recyclerView = (RecyclerView) layout.findViewById(R.id.drawerList);

 adapter = new NavigationDrawerAdapter(getActivity(), getData());
 recyclerView.setAdapter(adapter);
 recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
 recyclerView.setOnItemClickListener(new RecyclerViewTouchListener(getActivity(),
 recyclerView, new ClickListener() {
 @Override
 public void onClick(View view, int position) {
 drawerListener.onDrawerItemSelected(view, position);
 mDrawerLayout.closeDrawer(containerView);
 }
 }
 }
```



```
 @Override
 public void onLongClick(View view, int position) {

 }
 });

 return layout;
}

public void setUp(int fragmentId, DrawerLayout drawerLayout, final Toolbar toolbar) {
 containerView = getActivity().findViewById(fragmentId);
 mDrawerLayout = drawerLayout;
 mDrawerToggle = new ActionBarDrawerToggle(getActivity(), drawerLayout, toolbar,
R.string.drawer_open, R.string.drawer_close) {
 @Override
 public void onDrawerOpened(View drawerView) {
 super.onDrawerOpened(drawerView);
 getActivity().invalidateOptionsMenu();
 }

 @Override
 public void onDrawerClosed(View drawerView) {
 super.onDrawerClosed(drawerView);
 getActivity().invalidateOptionsMenu();
 }

 @Override
 public void onDrawerSlide(View drawerView, float slideOffset) {
 super.onDrawerSlide(drawerView, slideOffset);
 toolbar.setAlpha(1 - slideOffset / 2);
 }
 };

 mDrawerLayout.setDrawerListener(mDrawerToggle);
 mDrawerLayout.post(new Runnable() {
 @Override
 public void run() {
 mDrawerToggle.syncState();
 }
 });
}

public static interface ClickListener {
 public void onClick(View view, int position);

 public void onLongClick(View view, int position);
}

static class RecyclerViewTouchListener implements RecyclerView.OnItemTouchListener {

 private GestureDetector gestureDetector;
 private ClickListener clickListener;

 public RecyclerViewTouchListener(Context context, final RecyclerView recyclerView,
final ClickListener clickListener) {
 this.clickListener = clickListener;
 }
}
```



```

 gestureDetector = new GestureDetector(context,
 GestureDetector.SimpleOnGestureListener() {
 @Override
 public boolean onSingleTapUp(MotionEvent e) {
 return true;
 }

 @Override
 public void onLongPress(MotionEvent e) {
 View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
 if (child != null && clickListener != null) {

 clickListener.onLongClick(child,
 recyclerView.getChildPosition(child));
 }
 }
 });

 @Override
 public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {

 View child = rv.findChildViewUnder(e.getX(), e.getY());
 if (child != null && clickListener != null && gestureDetector.onTouchEvent(e))
 {
 clickListener.onClick(child, rv.getChildPosition(child));
 }
 return false;
 }

 @Override
 public void onTouchEvent(RecyclerView rv, MotionEvent e) {
 }

 @Override
 public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {
 }

 }

 public interface FragmentDrawerListener {
 public void onDrawerItemSelected(View view, int position);
 }
}

```

22. Por último, abrir el archivo de la plantilla principal `activity_main.xml` y modificar el código como se indica enseguida. En esta plantilla se está agregando `android.support.v4.widget.DrawerLayout` para mostrar el menú del `NavigationDrawer`. También, se debe asignar la ruta correcta de `FragmentDrawer` en el elemento `<fragment>`. **NOTA: Asignar con cuidado esta ruta:**

```

<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/drawer_layout"

```



```
 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
 android:id="@+id/container_toolbar"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical">

 <include
 android:id="@+id/toolbar"
 layout="@layout/toolbar" />
 </LinearLayout>

 <FrameLayout
 android:id="@+id/container_body"
 android:layout_width="fill_parent"
 android:layout_height="0dp"
 android:layout_weight="1" />

 </LinearLayout>

 <fragment
 android:id="@+id/fragment_navigation_drawer"
 android:name="com.example.escom.materialdesign.activity.FragmentDrawer"
 android:layout_width="@dimen/nav_drawer_width"
 android:layout_height="match_parent"
 android:layout_gravity="start"
 app:layout="@layout/fragment_navigation_drawer"
 tools:layout="@layout/fragment_navigation_drawer" />

</android.support.v4.widget.DrawerLayout>
```

Ahora, se tienen todos los archivos, de las plantillas y clases, en su lugar correcto. Enseguida, se realizan los cambios necesarios en el archivo **MainActivity** para el funcionamiento del **NavigationDrawer**.

23. Abrir el archivo **MainActivity.java** y realizar los siguientes cambios:

- Implantar la actividad desde **FragmentDrawer.FragmentDrawerListener** y añadir el método sobrescrito **onDrawerItemSelected()**.
- Crear una instancia de **FragmentDrawer** y asignar los escuchas a los elementos de selección:

```
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
```



```
public class MainActivity extends AppCompatActivity implements
FragmentManager.FragmentManagerListener {

 private Toolbar mToolbar;
 private FragmentDrawer drawerFragment;

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

 mToolbar = (Toolbar) findViewById(R.id.toolbar);

 setSupportActionBar(mToolbar);
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);

 drawerFragment = (FragmentDrawer)
getSupportFragmentManager().findFragmentById(R.id.fragment_navigation_drawer);

 drawerFragment.setUp(R.id.fragment_navigation_drawer, (DrawerLayout)
findViewById(R.id.drawer_layout), mToolbar);
 drawerFragment.setDrawerListener(this);
 }

 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.menu_main, menu);
 return true;
 }

 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 // Handle action bar item clicks here. The action bar will
 // automatically handle clicks on the Home/Up button, so long
 // as you specify a parent activity in AndroidManifest.xml.
 int id = item.getItemId();

 //noinspection SimplifiableIfStatement
 if (id == R.id.action_settings) {
 return true;
 }

 return super.onOptionsItemSelected(item);
 }

 @Override
 public void onDrawerItemSelected(View view, int position) {
 }
}
```

Ahora, si se ejecuta la aplicación, se muestra el NavigationDrawer con un encabezado y una lista con algunos





elementos:

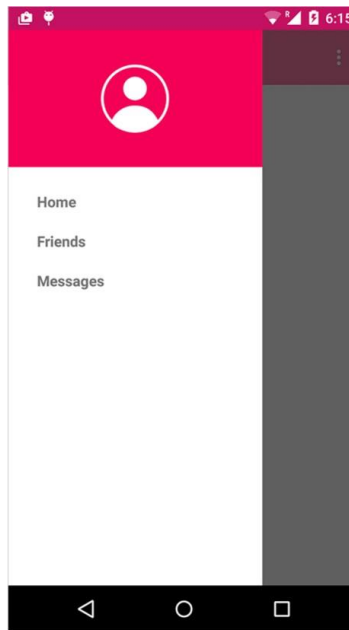


Figura 8. El NavigationDrawer.

## Implantación de la Selección de Elementos del NavigationDrawer.

Aunque el panel de navegación está funcionando, se puede notar que no se activa la selección de elementos de la lista. Esto se debe a que todavía se debe aplicar el clic de escucha a los elementos de RecyclerView.

Debido a que se tienen tres elementos de menú en el panel de navegación (Home, Friends y Messages), se deben crear tres clases de segmentos independientes para elemento del menú.

24. En `res->layout`, crear un archivo para una plantilla, nombrarla `fragment_home.xml` y añadir el código siguiente:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context="info.androidhive.materialdesign.activity.HomeFragment">
```

```
<TextView
 android:id="@+id/label"
 android:layout_alignParentTop="true"
 android:layout_marginTop="100dp"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:gravity="center_horizontal"
 android:textSize="45dp"
 android:text="HOME"
 android:textStyle="bold"/>
```

```
<TextView
 android:layout_below="@id/label"
 android:layout_centerInParent="true"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:textSize="12dp"
 android:layout_marginTop="10dp"
```



```
 android:gravity="center_horizontal"
 android:text="Edit fragment_home.xml to change the appearance" />
```

```
</RelativeLayout>
```

25. En el paquete **activity**, crear una clase de fragmento, nombrarla **HomeFragment.java** y añadir el código siguiente:

```
import android.app.Activity;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class HomeFragment extends Fragment {

 public HomeFragment() {
 // Required empty public constructor
 }

 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 }

 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState) {

 View rootView = inflater.inflate(R.layout.fragment_home, container, false);

 // Inflate the layout for this fragment
 return rootView;
 }

 @Override
 public void onAttach(Activity activity) {
 super.onAttach(activity);
 }

 @Override
 public void onDetach() {
 super.onDetach();
 }
}
```

26. Crear otras dos clases de fragmentos para **FriendsFragment.java** y **MessagesFragment.java** y sus respectivos archivos de plantilla **fragment\_friends.xml** y **fragment\_messages.xml**, y añadir el código como se hizo en los dos pasos anteriores.

27. Ahora, abrir el archivo **MainActivity.java** y realizar los cambios siguientes. En el siguiente código:

- El método **displayView()** muestra la vista del fragmento dependiendo de la selección del elemento en el menú de navegación. Este método debe invocarse en el método **onDrawerItemSelected()** para mostrar la vista correspondiente cuando se selecciona un elemento del menú de navegación:



```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity implements
FragmentManager.FragmentDrawerListener {

 private static String TAG = MainActivity.class.getSimpleName();

 private Toolbar mToolbar;
 private FragmentDrawer drawerFragment;

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

 mToolbar = (Toolbar) findViewById(R.id.toolbar);

 setSupportActionBar(mToolbar);
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);

 drawerFragment = (FragmentDrawer)
getSupportFragmentManager().findFragmentById(R.id.fragment_navigation_drawer);
 drawerFragment.setUp(R.id.fragment_navigation_drawer, (DrawerLayout)
findViewById(R.id.drawer_layout), mToolbar);
 drawerFragment.setDrawerListener(this);

 // display the first navigation drawer view on app launch
 displayView(0);
 }

 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.menu_main, menu);
 return true;
 }

 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
 // Handle action bar item clicks here. The action bar will
 // automatically handle clicks on the Home/Up button, so long
 // as you specify a parent activity in AndroidManifest.xml.
 int id = item.getItemId();

 //noinspection SimplifiableIfStatement
```



```
 if (id == R.id.action_settings) {
 return true;
 }

 if(id == R.id.action_search){
 Toast.makeText(getApplicationContext(), "Search action is selected!",
Toast.LENGTH_SHORT).show();
 return true;
 }

 return super.onOptionsItemSelected(item);
 }

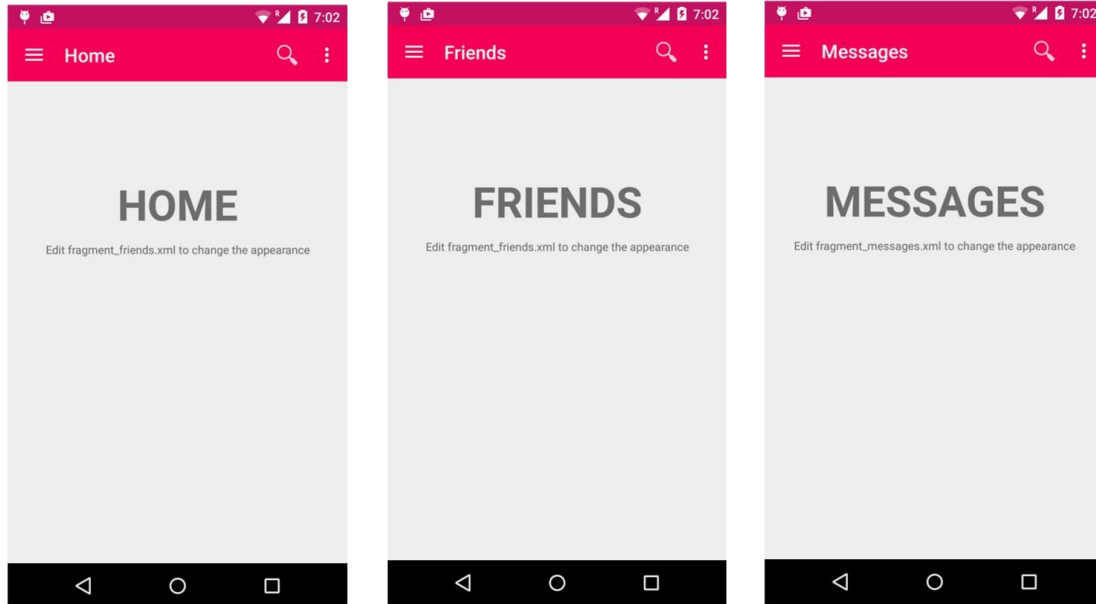
 @Override
 public void onDrawerItemSelected(View view, int position) {
 displayView(position);
 }

 private void displayView(int position) {
 Fragment fragment = null;
 String title = getString(R.string.app_name);
 switch (position) {
 case 0:
 fragment = new HomeFragment();
 title = getString(R.string.title_home);
 break;
 case 1:
 fragment = new FriendsFragment();
 title = getString(R.string.title_friends);
 break;
 case 2:
 fragment = new MessagesFragment();
 title = getString(R.string.title_messages);
 break;
 default:
 break;
 }

 if (fragment != null) {
 FragmentManager fragmentManager = getSupportFragmentManager();
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
 fragmentTransaction.replace(R.id.container_body, fragment);
 fragmentTransaction.commit();

 // set the toolbar title
 getSupportActionBar().setTitle(title);
 }
 }
}
```

Entonces, si ahora se ejecuta la aplicación, se muestra que sí funciona la selección del menú del panel de navegación y debajo de la barra de herramientas se muestran los correspondientes elementos:



**Figura 9.** Muestra de cada fragmento desde el menú del panel de navegación.

**Ejercicio 1:** Cambiar el texto de todos los componentes del idioma inglés al idioma español.

**Ejercicio 2:** Probar la ejecución del código anexo K.java y después agregarlo a la ejecución de la aplicación móvil.

**NOTA:** Generar un reporte de la aplicación final. Agregar la carpeta comprimida del proyecto en un archivo con la sintaxis AlumnoMaterialDesignGrupo.zip y enviarla al sitio indicado por el profesor.

## ANEXO.

El código K.java.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class K extends JPanel{
 JFrame jf;
 float x, y, dir;
 int level = 1;
 public K(){
 jf = new JFrame("K");
 addMouseListener(new MouseAdapter(){
 public void mousePressed(MouseEvent evt){
 level++;
 repaint();
 }
 });
 jf.add(this);
 jf.setVisible(true);
 }
 public void paintComponent(Graphics g){
 int length = getWidth()/2;
 y = (float)(getHeight()/2 + Math.sin(Math.toRadians(30))*length/2);
 x = this.getWidth()/4;
 g.setColor(Color.white);
 g.fillRect(0, 0, getWidth(), this.getHeight());
 }
}
```



```
g.setColor(Color.black);
dir = 0;
dK(g, length, level);
dir -= 120;
drawKoch(g, length, level);
dir -= 120;
drawKoch(g, length, level);
}
public void dK(Graphics g, double len, int n){
 if (n==0){
 double dirRad = Math.toRadians(dir);
 double xInc = len * Math.cos(dirRad); // x increment
 double yInc = len * Math.sin(dirRad); // y increment
 float x1 = x + (float)xInc;
 float y1 = y + (float)yInc;
 g.drawLine((int)x, (int)y, (int)x1, (int)y1);
 x = x1;
 y = y1;
 }else{
 dK(g, len/=3, --n);
 dir += 60;
 dK(g, len, n);
 dir -= 120;
 dK(g, len, n);
 dir += 60;
 dK(g, len, n);
 }
}
}
public static void main(String [] args){
 new K();
}
}
```