To multidevice **appearance** application hold shift and press one frame, than another and in right click mouse choose chains, then in right tool window click on blue lines, they will change in way to change size with width, change margin with width or constantly hold

<uses-permission android:name="android.permission.READ\_CONTACTS"></uses-permission> - set permissions

<android:permission="android.permission.READ\_Contacts"> - set permissions in activity

You can create interface in frame, then implement it in main activity and override methods. Then override method **onAttach** in frame and create there example of that interface. If you use functions in this variable they will process in main activity

Write strings in special file

TextView view = findViewById(R.id.second); - get variable

Log.d("title", "message"); - show logs

**Toast.makeText(getActivity(),"**string**",Toast.LENGTH\_SHORT).show();** - alert on the bottom of the screen

**Actions**

To create action on button you have to create new activity **new->create activity->empty activity** set function name in tool window (onclick) and create this action in class, where button is placed, then write in function:

public void main2(View view){ - create function and get this parameter  
 Intent intent = new Intent(this, Main2Activity.class); - create intent, second variable is name of action class + Action   
 startActivity(intent); - set new window   
}

Exchange info between actions

EditText text = findViewById(R.id.ft); - get variable by id  
String str = text.getText().toString(); - get text  
  
Intent intent = new Intent(this, Main2Activity.class); - get intent  
intent.putExtra(“var1”, str); - record as map  
startActivity(intent); - start activity

Second class

Intent intent = getIntent(); - get current intent   
  
String str = intent.getStringExtra(“var1”); - get str by key  
TextView view = findViewById(R.id.second); - get text variable by id (second)  
view.setText(str); - set text

Create shortcut to the selected class

In file AndroidManifest in needed action add parameter android:parentActivityName=”.name of class”, this will add an arrow on the top bar, press it and you will go to the action of the pasted class

Fragments

Create new xml in layout and new class. Then write in class

public class top extends Fragment {  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 return inflater.inflate(R.layout…., container, false); - in first parameter add needed xml file from layout  
 }  
}

Insert through XML

then in the main xml, where you want to paste those fragments write

<LinearLayout  
 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”  
 android:orientation=”vertical”  
 >  
<fragment  
 android:layout\_width=”match\_parent”  
 android:layout\_height=”100px”  
 android:id=”@+id/bot”  
 android:layout\_weight=”1”  
 android:name=”com.example.boris\_age.myapplication.bot”></fragment>  
  
</LinearLayout>

Insert layout in runtime

In main XML add layout and set id (for example conteiner), than create layout as usual (on the top). Than in main class add first layout as base one

FragmentManager fragmentManager = getSupportFragmentManager();

if (findViewById(R.id.conteiner) != null) { - if it is container is free  
 if (savedInstanceState != null) { - must enter for it not to be destroyed   
 return;  
 }

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();   
 bot b = new bot(); - example of needed class  
 fragmentTransaction.add(R.id.conteiner, b, null); - set class  
 fragmentTransaction.commit(); - push it  
}

Then replace fragments

Create new layout as usual, than in current layout class add:

View view = inflater.inflate(R.layout.bot, container, false); - get current page  
button = view.findViewById(R.id.button); - find needed button  
  
button.setOnClickListener(new View.OnClickListener(){ - replace on click  
 @Override  
 public void onClick(View v) {  
 MainActivity.fragmentManager.beginTransaction().replace(R.id.conteiner, new top(),null).commit(); - if you want to come back by pressing “back” – add .addToBackStack(null) before commit or:

getActivity().getSupportFragmentManager().beginTransaction().replace(R.id.lay, new create(), null).addToBackStack(null).commit();

}  
});  
return view;

Connect main activity with fragment

At fragment create inside interface, where create needed methods, then implement this class into main activity and override methods as you need. Create example of interface and in fragment override method onAttach, where you set example of interface:

Exampe\_of\_interface exampe\_of\_interface;

@Override  
public void onAttach(Context context) {  
 super.onAttach(context);  
 Activity activity = (Activity)context;  
 exampe\_of\_interface = (Exampe\_of\_interface)activity;  
}

Then create action listener in onCreate method and there you need to run needed methods from that variable (in example is activity)

Connect fragment activity with fragment

Do the same as on the top, but in the method, overridden in main create or replace fragment with parametrs:

bot b = new bot(); - create example of fragment  
Bundle bundle = new Bundle(); - create case of values  
bundle.putString("mess", str); - set values with keys  
b.setArguments(bundle); - add to your fragment this values  
  
FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction().add(R.id.container, b, null); - create fragment in fragment place in main with id container, and set fragment b  
fragmentTransaction.addToBackStack(null); - just write  
fragmentTransaction.commit(); - add fragment

then in fragment get elements in main method (onCreate):

Bundle bundle = getArguments();

And do what you want

**Preferences**

They are being saved even if application is closed

Create main activity and class which will process information. Import **android.content.SharedPreferences;**

SharedPreferences sharedPreferences = context.getSharedPreferences("name", Context.MODE\_PRIVATE); - create preference, to get context use:

getApplicationContext()

sharedPreferences.edit().putBoolean("key", value).commit(); - set variable (int, Boolean, String, …)

sharedPreferences.getBoolean("name", default); - get valuable or default one