Bottom line message

Snackbar snackbar = Snackbar.make(findViewById(R.id.constrain), "…", Snackbar.LENGTH\_SHORT);

snackbar .show() - display line at the bottom alert bar, add design to the grades, constrains is an id of main block of application

snackbar.setAction("button’s name", this); - add button with listener, which you have to implement OnClickListener and override its method

Alert with yes, no and neutral buttons

public class DialigAlert extends DialogFragment {  
 @NonNull  
 @Override  
 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity()); // create builder  
 builder.setMessage("main text"); // text

builder.setPositiveButton("yes", new DialogInterface.OnClickListener() { // if yes  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
   
 }  
 });  
 builder.setNegativeButton("no", new DialogInterface.OnClickListener() { // if no  
 @Override  
 public void onClick(DialogInterface dialog, int which) {

}  
 });  
 builder.setNeutralButton("nasrat", new DialogInterface.OnClickListener() { // if pohui  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
 });  
 return builder.create();  
 }  
}

In main:

new DialigAlert().show(getSupportFragmentManager(), "tag"); - start created class

alert with personal answers or list view

public class AlertWithManyItems extends DialigAlert {  
 @NonNull  
 @Override  
 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity()); // create builder  
 builder.setTitle("do action?"); // text  
  
 final String[] str = getResources().getStringArray(R.array.*arr*); - get strings  
 builder.setItems(str, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {

}  
 });  
 return builder.create();  
 }  
}

In main:

new AlertWithManyItems().show(getSupportFragmentManager(), "tag");

Check boxes in alerts

public class CheckBoxAlert extends DialigAlert {  
  
 @NonNull  
 @Override  
 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
 final String[] str = getResources().getStringArray(R.array.*arr*);  
 final List<String> list = new ArrayList(); - for collecting elements after each press   
  
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity()); // create builder  
 builder.setTitle("do action?"); // text  
  
 builder.setMultiChoiceItems(str, null, new DialogInterface.OnMultiChoiceClickListener() { - second parameter is default variables, method, which handle each click   
 @Override  
 public void onClick(DialogInterface dialog, int which, boolean isChecked) {  
 if(isChecked){  
 list.add(str[which]);  
 }else{  
 list.remove(str[which]);  
 }  
 }  
 });  
 return builder.create();  
 }  
}

In main:

new CheckBoxAlert().show(getSupportFragmentManager(), "tag");

Radio button alert

public class RadioButtonAlert extends DialigAlert {  
  
 String string;  
  
 @NonNull  
 @Override  
 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
 final String[] str = getResources().getStringArray(R.array.*arr*);  
 string = "";  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity()); // create builder  
 builder.setTitle("do action?"); // text  
  
 builder.setSingleChoiceItems(str, -1, new DialogInterface.OnClickListener() { - check each press, second parameter - default  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 string = str[which];  
 }  
 });  
 return builder.create();  
 }  
}

In main:

new RadioButtonAlert().show(getSupportFragmentManager(), "tag");

Custom alert

public class CustomAlert extends DialigAlert {  
 @NonNull  
 @Override  
 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
 LayoutInflater inflater = getActivity().getLayoutInflater();  
  
 builder.setView(inflater.inflate(R.layout.*custom\_dialog*, null));

return builder.create();  
 }  
}

In main:

new CustomAlert().show(getSupportFragmentManager(), "tag");

Communicate main class with dialog one

In dialog class crate interface with needed methods, then create variable of this interface and set in on attach method by hardcoding parameter activity to needed interface. Then use methods of this variable where needed in dialog class. Then implement this interface in main class and override this methods ow you need