Dialog（对话框）

1、简介

Dialog，对话框，这是一种在App中非常常见的交互类型，主要是用于给用户提供选择的。常见的对话框有询问确定或取消那一类的普通的对话框，也有如时间选择器和日期选择器一类的对话框。他们之间的使用尽管存在差异，但是流程基本一致，创建对话框、设置参数、设置相应的监听器和显示对话框

2、基本用法

2.1、标准的询问对话框

创建 AlertDialog.Builder 对象:

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);
```

设置对话框的相关参数:

设置对话框的对应参数如：标题文本、对话框信息、按钮的文本和按钮的监听器等等

```
builder.setIcon(R.mipmap.ic_launcher_round)    //对话框图标
        .setCancelable(true)  //是否可以点击外部关闭对话框
        .setMessage("This the Message")    //对话框信息
        .setNegativeButton("Cancel", new DialogInterface.OnClickListene
            @Override
            public void onClick(DialogInterface dialogInterface, int i)
                //否定按钮点击后的效果
            }
        })
        .setPositiveButton("OK", new DialogInterface.OnClickListener()
            @Override
            public void onClick(DialogInterface dialogInterface, int i)
              //确定按钮点击后的效果
            }
        })
        .setTitle("AlertDialog");    //对话框标题
Dialog dialog = builder.create();    //创建对话框
```

弹出对话框:

在需要弹出对话框的位置调用以下代码：

```
dialog.show();
```

关闭对话框：

在需要关闭对话框的位置调用以下代码

```
dialog.dismiss();
```

效果图：

2.2、选择对话框

创建 AlertDialog.Builder 对象:

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);
```

设置对话框的相关参数：

```
String[] items = new String[]{"OptionA", "OptionB", "OptionC"};   //初始
builder.setTitle("单选对话框");
builder.setItems(items, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        //选项被点击以后的操作
    }
});
//      由于Message和Item列表都是占用的内容区域，所以两个不能同时存在
//      builder.setMessage("请选择一下选项之一");
Dialog dialog = builder.create();
```

弹出对话框：

在需要弹出对话框的位置调用以下代码：

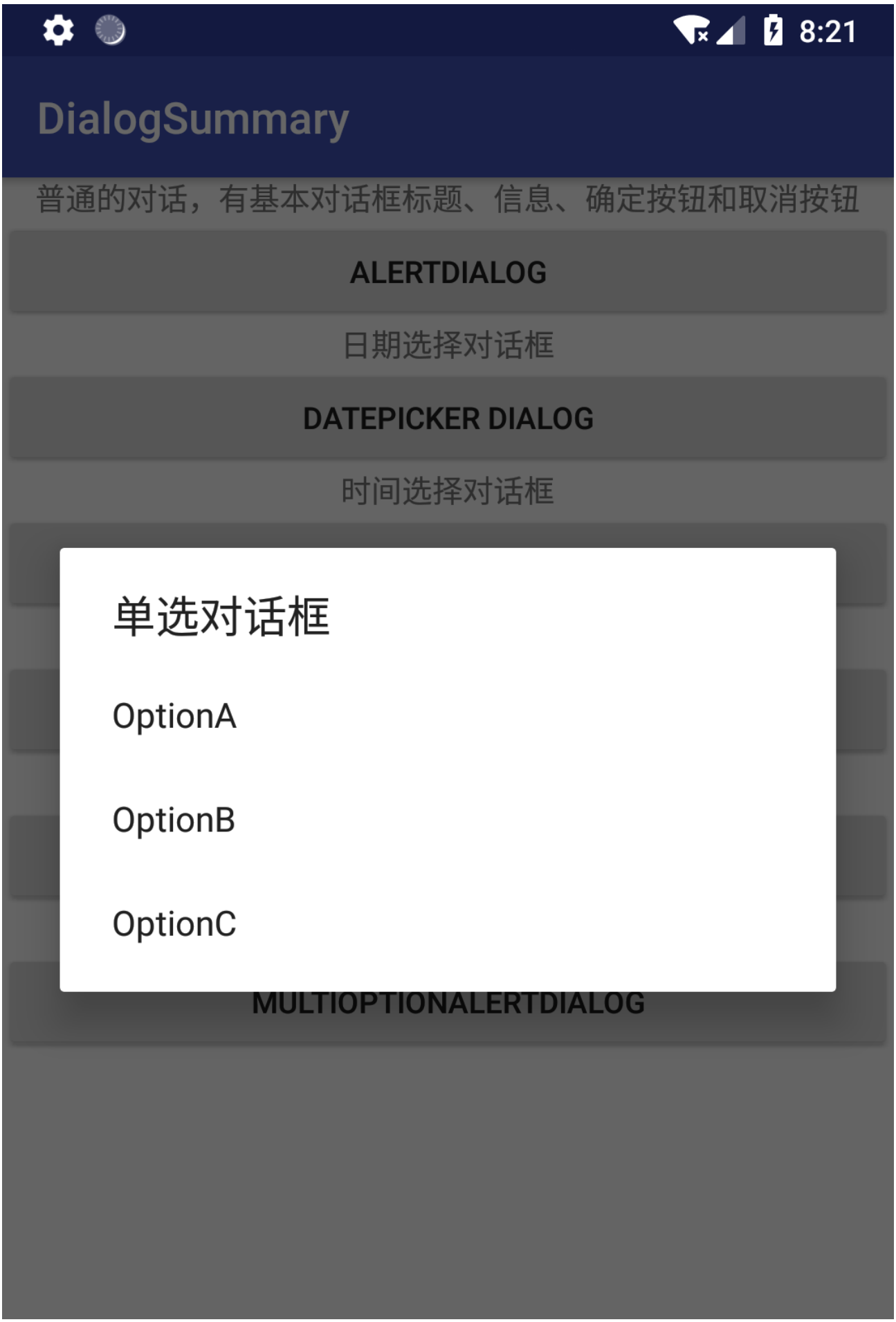
```
dialog.show();
```

关闭对话框：

在需要关闭对话框的位置调用以下代码

```
        dialog.dismiss();
```

效果图：



2.3、单选项对话框

注意：与2.2的单选对话框的不同之处在于，2.3这个单选是以 RatioButton 的形式存在并且在点击以后Dialog不会伴随消失

创建 AlertDialog.Builder 对象:

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);
```

设置对话框的相关参数:

```
builder.setTitle("单选对话框");
String[] items = new String[]{"OptionA", "OptionB", "OptionC"};    //初
//与2.2之间的区别
builder.setSingleChoiceItems(items, 0 /*默认被选择的选项*/, new DialogIn
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        //选项点击时的操作
    }
}).setNegativeButton("取消", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        //取消按钮点击时的操作
    }
}).setPositiveButton("确定", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        //确定按钮点击时的操作
    }
});
Dialog dialog = builder.create();
```

弹出对话框:

在需要弹出对话框的位置调用以下代码:

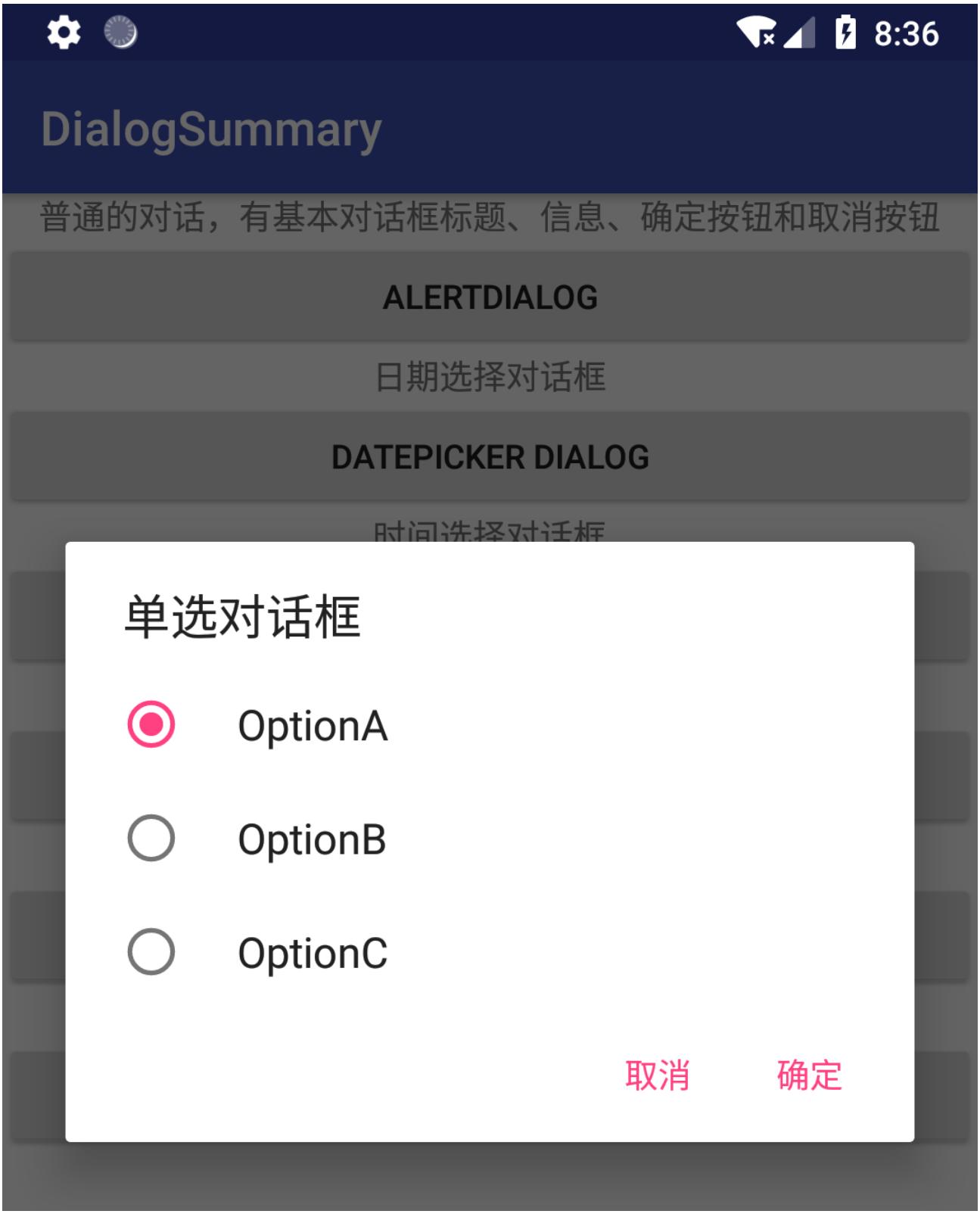
```
dialog.show();
```

关闭对话框：

在需要关闭对话框的位置调用以下代码

```
dialog.dismiss();
```

效果图：



2.4、多选项对话框

创建 AlertDialog.Builder 对象:

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);
```

设置对话框的相关参数:

```
builder.setTitle("单选对话框");
String[] items = new String[]{"OptionA", "OptionB", "OptionC"};   //初
//单选与多选之间的区别
boolean flags = new boolean[]{true, false, true};   //各选项的初始默认值
builder.setMultiChoiceItems(items, flags , new DialogInterface.OnMult
    @Override
    public void onClick(DialogInterface dialogInterface, int i, boole
          //选项被选上时的操作
    }
}).setNegativeButton("取消", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
          //取消键被按下时的操作
    }
}).setPositiveButton("确定", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
          //确定键被按下时的操作
    }
});
//      由于Message和Item列表都是占用的内容区域，所以两个不能同时存在
//        builder.setMessage("请选择一下选项之一");
Dialog dialog = builder.create();
```

弹出对话框：

在需要弹出对话框的位置调用以下代码：

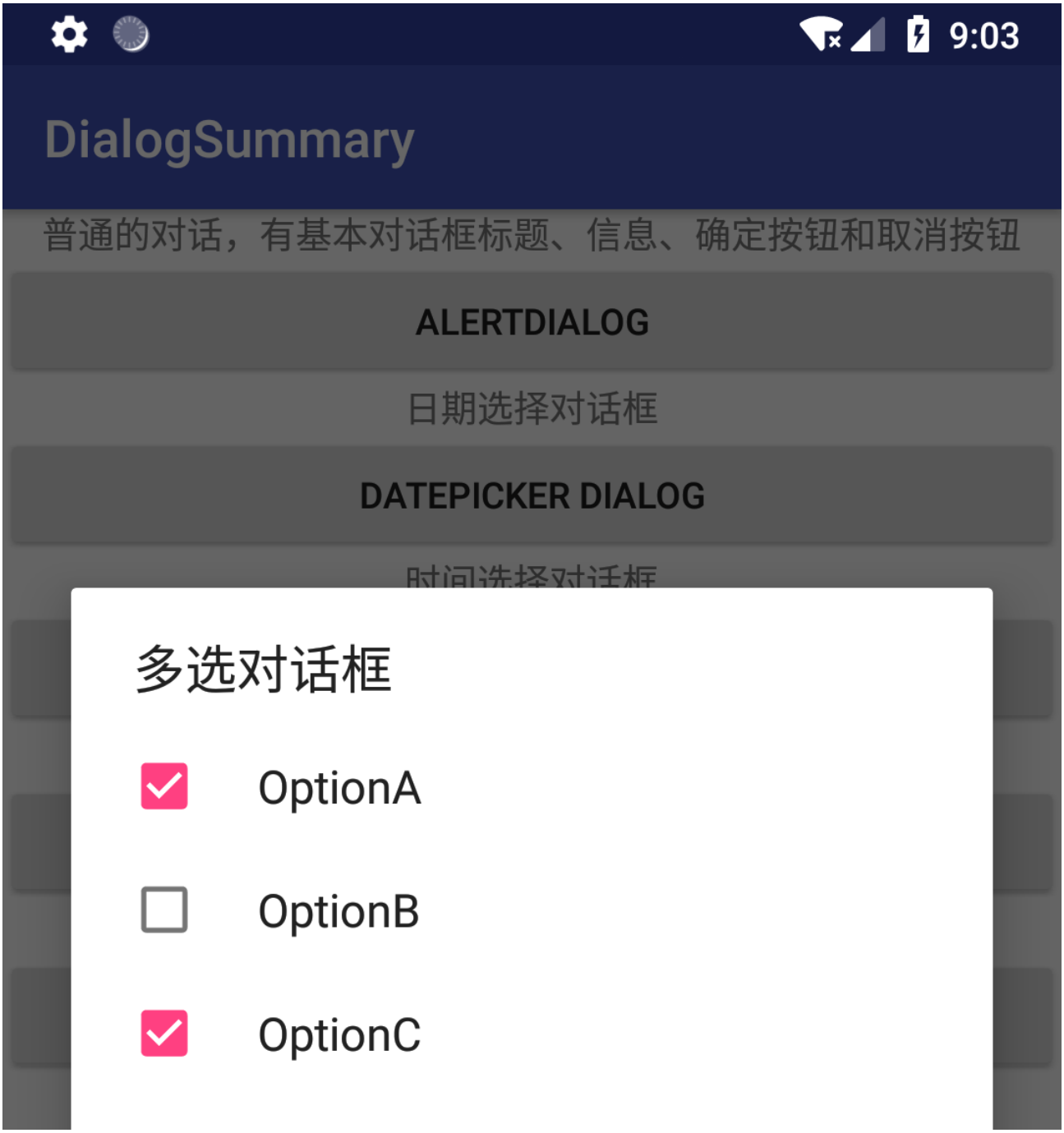
```
dialog.show();
```

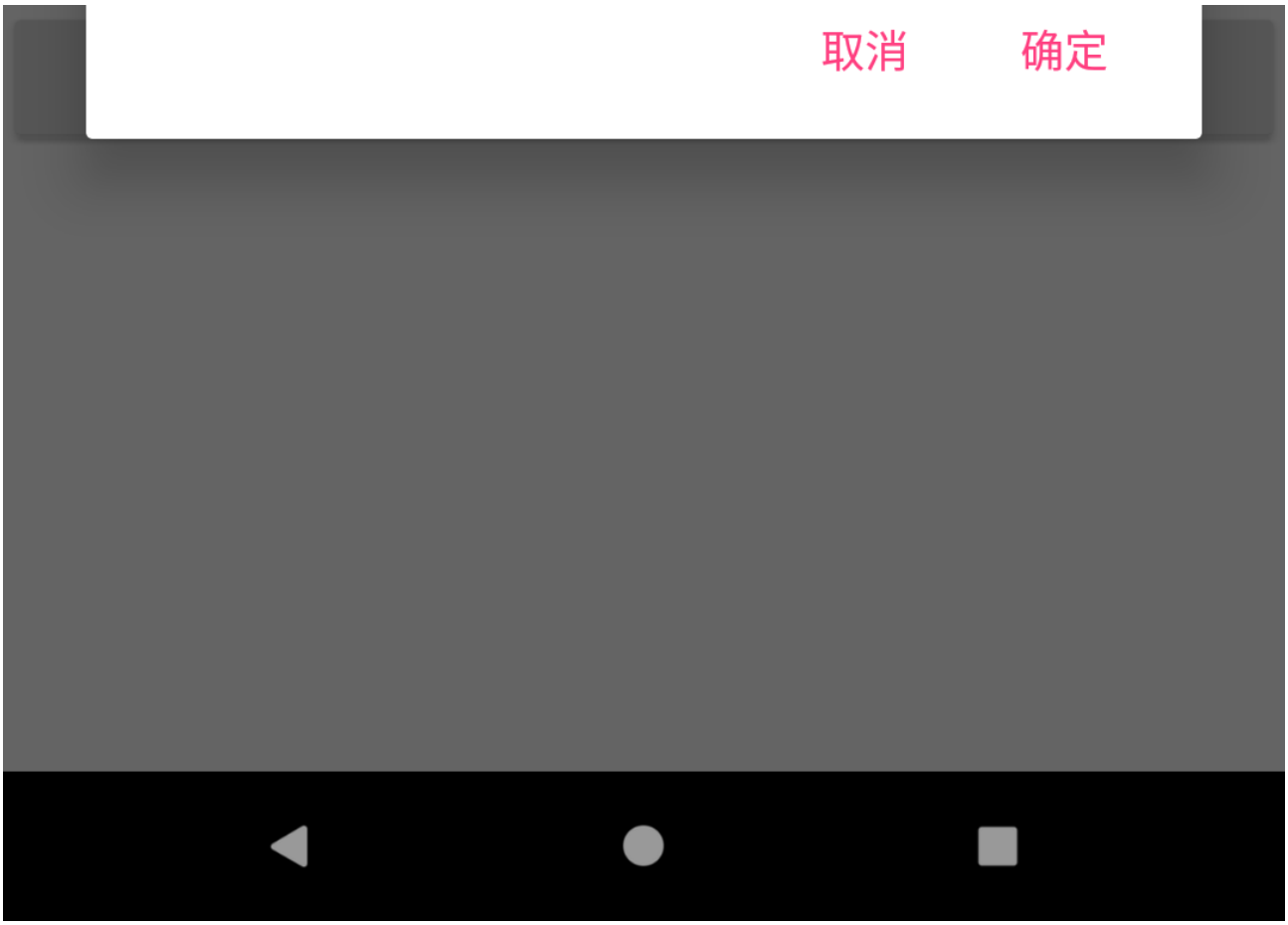
关闭对话框：

在需要关闭对话框的位置调用以下代码

```
dialog.dismiss();
```

效果图：





2.5、时间选择器

创建 TimePickerDialog 对象:

构造函数中参数分别为：上下文对象，时间选择监听器，初始时，初始分，是否以24小时显示

```
TimePickerDialog dialog = new TimePickerDialog(this, new TimePickerDi
    @Override
    public void onTimeSet(TimePicker timePicker, int i, int i1) {
        //按下确定后对所选时间进行相应操作
    }
}, Calendar.getInstance().get(Calendar.HOUR_OF_DAY), Calendar.getInst
```

弹出对话框:

在需要弹出对话框的位置调用以下代码:

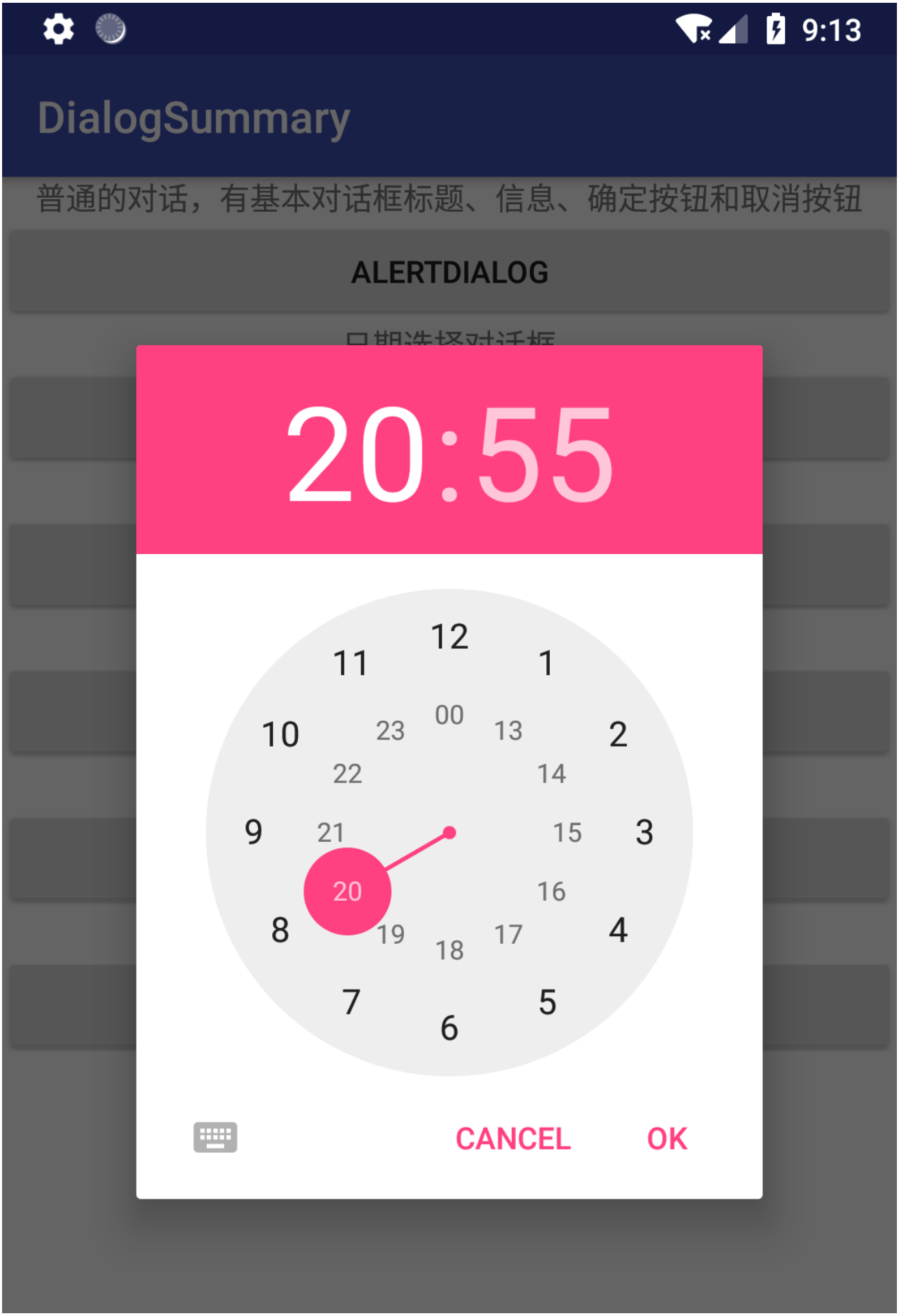
```
dialog.show();
```

关闭对话框:

在需要关闭对话框的位置调用以下代码

```
        dialog.dismiss();
```

效果图：



2.6、日期选择器

创建 DatePickerDialog 对象:

构造函数中参数分别为：上下文对象，日期选择监听器，初始年，初始月，初始日

```
DatePickerDialog datePickerDialog = new DatePickerDialog(this, new Da
    @Override
    public void onDateSet(DatePicker datePicker, int i, int i1, int i
        //按下确定后对所选日期进行相应操作
    }
}, Calendar.getInstance().get(Calendar.YEAR), Calendar.getInstance().
```

弹出对话框:

在需要弹出对话框的位置调用以下代码:

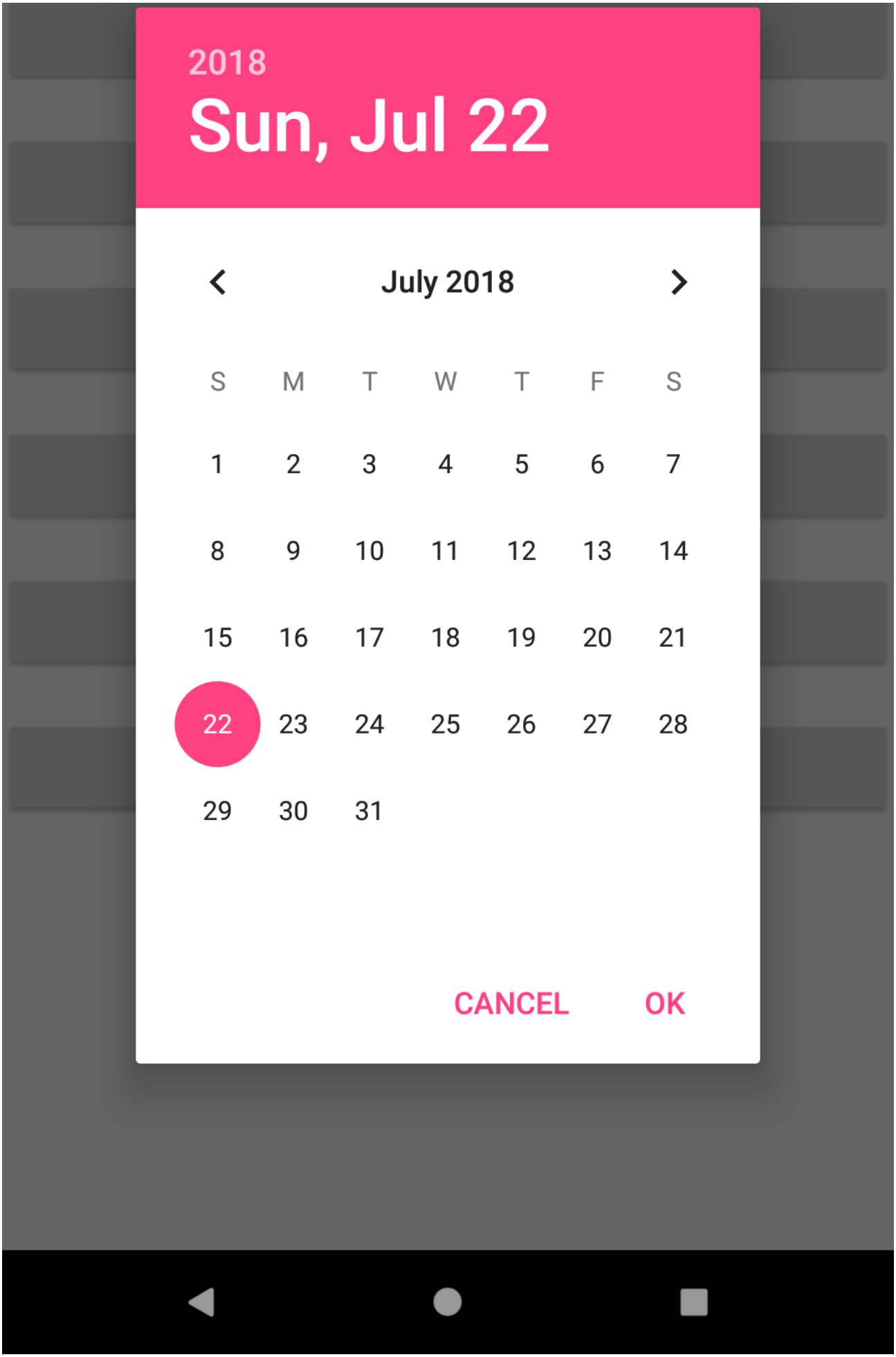
```
dialog.show();
```

关闭对话框:

在需要关闭对话框的位置调用以下代码

```
dialog.dismiss();
```

效果图:


2018
Sun, Jul 22
July 2018
S M T W T F S
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
CANCEL
OK

总结

Dialog 对话框，作为一个原生的控件，Android给它定制很多种样式，使用步骤主要围绕着创建Dialog、设置相关参数、显示、处理交互结果（各种Listener）几个大的步骤，比较简单。但是简单的东西限制就会多，总有定制好的Dialog样式无法满足需求的时候，这时候我们就需要自定义了，自定义Dialog可以让Dialog这个控件更加具有生命力，对此我后面再作专项总结。