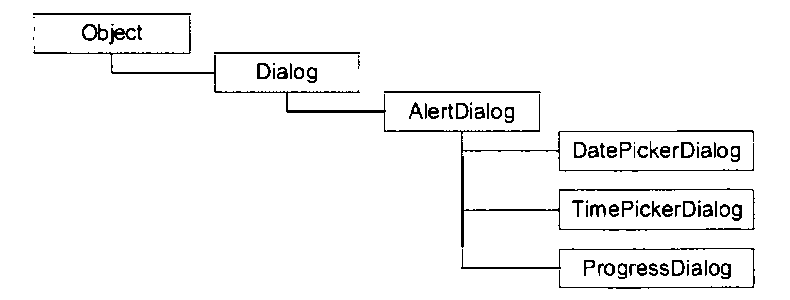
**Диалоговые окна**

**Диалог**— обычно маленькое окно, которое появляется перед текущей деятельностью. Основная деятельность при этом теряет фокус, и диалог принимает все пользовательское взаимодействие. Диалоги обычно используются для уведомлений и коротких действий, которые непосредственно касаются приложения, а также для индикации прогресса выполнения длительных задач.

Android поддерживает следующие типы диалоговых окон:

* **AlertDialog**— диалог с кнопками, списком, флажками или радиокнопками;
* **ProgressDialog** — диалог с индикатором прогресса;
* **DatePickerDialog** — диалог выбора даты;
* **TirnePickerDialog** — диалог выбора времени.

Иерархия классов диалоговых окон представлена на рисунке:



**Класс Dialog**является базовым для всех классов диалоговых окон. Поскольку***ProgressDialog, TirnePickerDialog и DatePickerDialog***— расширение **класса AlertDialog**, они также могут иметь командные кнопки.

* [AlertDialog](http://androidfanclub.ru/programming/alertdialog)
* [DatePickerDialog](http://androidfanclub.ru/programming/datepickerdialog)
* [ProgressDialog](http://androidfanclub.ru/programming/progressdialog)
* [TimePickerDialog](http://androidfanclub.ru/programming/timepickerdialog)
* [Создание диалоговых окон](http://androidfanclub.ru/programming/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D0%B4%D0%B8%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D1%8B%D1%85-%D0%BE%D0%BA%D0%BE%D0%BD)
* [Создание собственных диалогов](http://androidfanclub.ru/programming/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D1%81%D0%BE%D0%B1%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D1%85-%D0%B4%D0%B8%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2)

**AlertDialog**

**Диалог AlertDialog**— расширение ***класса Dialog***. Он используется при построении большинства диалоговых окон. В этих диалогах доступна для использования любая функциональность из нижеперечисленных:

* заголовок;
* текстовое сообщение;
* одна, две или три кнопки;
* список;
* флажки;
* радиокнопки.
* [AlertDialog с кнопками](http://androidfanclub.ru/programming/alertdialog-%D1%81-%D0%BA%D0%BD%D0%BE%D0%BF%D0%BA%D0%B0%D0%BC%D0%B8)
* [AlertDialog с радиокнопками](http://androidfanclub.ru/programming/alertdialog-%D1%81-%D1%80%D0%B0%D0%B4%D0%B8%D0%BE%D0%BA%D0%BD%D0%BE%D0%BF%D0%BA%D0%B0%D0%BC%D0%B8)
* [AlertDialog с флажками](http://androidfanclub.ru/programming/alertdialog-%D1%81-%D1%84%D0%BB%D0%B0%D0%B6%D0%BA%D0%B0%D0%BC%D0%B8)
* [AlertDialog со списком](http://androidfanclub.ru/programming/alertdialog-%D1%81%D0%BE-%D1%81%D0%BF%D0%B8%D1%81%D0%BA%D0%BE%D0%BC)

**AlertDialog с кнопками**

Для создания **AlertDialog с кнопками** используется группа методов set...Button() ***класса AlertDialog.Builder***:

* setPositiveButton ();
* setNegativeButton ();
* setNeutralButton ().

Для создания диалогового окна сначала надо создать объект **класса Builder**, передав в качестве параметра контекст приложения:

AlertDialog.Builder builder

new AlertDialog.Builder(getApplicationContext());

Затем, используя ***методы класса Builder***, задать для создаваемого диалога необходимые свойства, например текстовое сообщение в окне методом ***setMessage():***

builder.setMessage("Are you sure you want to exit?");

После задания свойств диалога определяют командные кнопки диалога и обработку событий на них. В **AlertDialog** можно добавить только по одной кнопке каждого типа: Positive, Negative и Neutral, т. е. максимально возможное количество кнопок в диалоге — три.

Для каждой кнопки используется один из методов **set.Button()**, которые принимают в качестве параметров надпись для кнопки и **интерфейс Dialoglnterface**. *OnClickListener*, который определяет действие, когда пользователь нажимает кнопку. Например, для создания диалога с кнопками *Yes* и *No* код будет выглядеть приблизительно так:

builder.setPositiveButton("Yes", new Dialoglnterface.OnClickListener()

public void onClick(Dialoglnterface dialog, int id)

// закрываем текущую деятельность

AlertDialogButtonActivity.this.finish();

}) ;

builder.setNegativeButton("No", new Dialoginterface.OnClickListener()

public void onClick(Dialoginterface dialog, int id)

// закрываем диалог и возвращаемся к текущей деятельности dialog.cancel();

Чтобы пользователь не мог закрыть диалог клавишей *<Back>* на клавиатуре мобильного устройства, вызывается **метод setcancelable ():**

builder.setCancelable(false);

И наконец, отображаем диалог:

AlertDialog alert builder.create();

Создайте новый проект и в диалоге *Create New Project* введите следующие значения:

* Project name — AlertDialogButtonApp;
* Application name — AlertDialogButton Sample;
* Package name — corn. samples. alertdialogbutton;

Откройте файл разметки и создайте структуру разметки с единственной кнопкой для вызова диалога:

Листинг. Файл разметки main.xml

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

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<Button

android:id="@+id/button"

android:text="Call AlertDialog with Buttons"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"/>

</LinearLayout>

В ***классе AlertDialogButtonActivity*** реализуйте последовательность создания диалога, приведенную ранее. Полный код класса представлен в листинге:

Листинг. Файл AlertDialogButtonActivity.java

package com.samples.alertdialogbutton;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

public class AlertDialogButtonActivity extends Activity {

private final int IDD\_EXIT = 0;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final Button callButton = (Button)findViewById(R.id.button);

// добавляем слушатель события для кнопки вызова диалога

callButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

showDialog(IDD\_EXIT);

}

});

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case IDD\_EXIT:

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("Are you sure you want to exit?");

builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

AlertDialogButtonActivity.this.finish();

}

});

builder.setNegativeButton("No", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dialog.cancel();

}

});

builder.setCancelable(false);

return builder.create();

default:

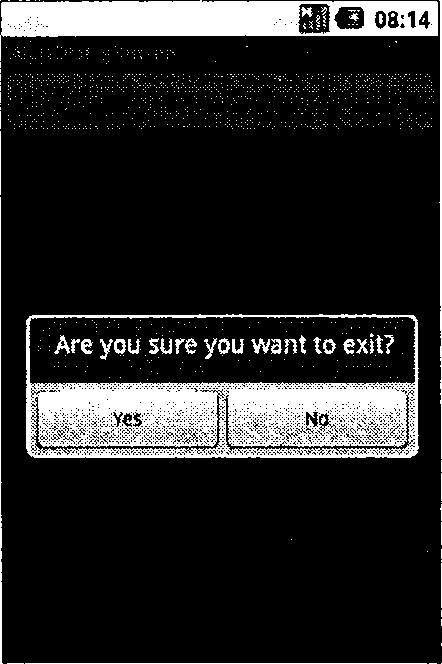
return null;

}

}

}

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться***окно AlertDialog*** с кнопками Yes и No. При закрытии диалога кнопкой Yes приложение закончит работу.



*AlertDialog с кнопками Yes и No*

**AlertDialog с радиокнопками**

Для создания **диалогового окна с радиокнопками** применяется **метод setSingleChoiceitems()**. Если диалоговое окно создается внутри ***onCreateDialog()***, система Android управляет состоянием списка с радиокнопками. Пока текущая деятельность активна, диалоговое окно при последующих вызовах запоминает ранее выбранные пункты.

Создание **AlertDialog с радиокнопками** похоже на создание диалога со списком, только вместо **метода setitems()** вызывается **метод setSingleChoiceitems():**

CharSequence[] colors {"Red", "Green", "Blue"};

builder.setSingleChoiceitems(colors, О,

new Dialoginterface.OnClickListener()

{ public void onClick(Dialoginterface dialog, int item)

Toast.makeText(getApplicationContext(), "Color: + mColors[item],

Toast.LENGTH\_SHORT) .show();

Первый параметр в **методе setSingleChoiceitems()**— массив значений для радиокнопок,второй параметр— целочисленное значение индекса радиокнопки, которая будет включена по умолчанию при вызове диалога. Если требуется по умолчанию установить все радиокнопки в выключенное состояние, необходимо установить значение второго параметра в 1.

При нажатии радиокнопки диалог закрываться не будет. В отличие от диалога со списком, который закрывается после выбора элемента списка, диалог с радиокнопками требует добавления командных кнопок для управления диалогом. Диалог закрывается только командами кнопками.

**Для примера приложения с вызовом диалогового окна с радиокнопками создайте новый проект и в диалоге *Create New Project* введите следующие значения:**

* Project name — AlertDialogRadioButtonsApp;
* Application name — AlertDialogRadioButtons Sample;
* Package name — com.samples.alertdialogradiobuttons;
* Create Activity— AlertDialogRadioButtonsActivity.

В **классе AlertDialogRadioButtonsActivity**реализуйте последовательность создания диалога с радиокнопками, как в показанном ранее примере. Полный код класса представлен в листинге:

Файл AlertDialogRadioButtonsActivity.java

package com.samples.alertdialogradiobuttons;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.Toast;

public class AlertDialogRadioButtonsActivity extends Activity {

private final static int IDD\_COLOR = 0;

private final CharSequence[] mColors = {"Red", "Green", "Blue"};

private int mResult;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final Button callButton = (Button)findViewById(R.id.button);

// добавляем слушатель события для кнопки вызова диалога

callButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

showDialog(IDD\_COLOR);

}

});

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case IDD\_COLOR:

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Pick a color");

builder.setSingleChoiceItems(mColors, 0, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int item) {

//Toast.makeText(getApplicationContext(), "Color: " + mColors[item], Toast.LENGTH\_SHORT).show();

mResult = item;

}

});

builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

Toast.makeText(getApplicationContext(), "Color: " + mColors[mResult], Toast.LENGTH\_SHORT).show();

}

});

builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dialog.cancel();

Toast.makeText(getApplicationContext(), "Dialog cancel", Toast.LENGTH\_SHORT).show();

}

});

builder.setCancelable(false);

return builder.create();

default:

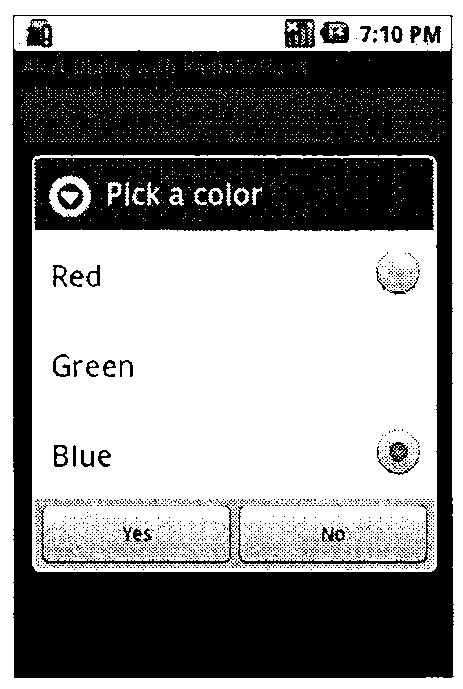
return null;

}

}

}

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться**окно AlertDialog** с тремя радиокнопками для выбора цвета. При выборе одной из кнопок появится всплывающее уведомление, показывающее выбранный цвет.



*AlertDialog с радиокнопками*

**AlertDialog с флажками**

Для создания диалогового окна с радиокнопками применяется **метод setSingleChoiceitems().** Если диалоговое окно создается внутри **onCreateDialog()**, система Android управляет состоянием списка. Пока текущая деятельность активна, диалоговое окно при последующих вызовах запоминает ранее выбранные пункты.

Создание диалогового окна с флажками очень похоже на создание диалога с радиокнопкам и, только вместо **метода setSingleChoiceitems()** вызывается метод **setMultiChoiceitems():**

CharSequence[] colors {"Red", "Green", "Blue"};

final boolean[] mCheckeditems {true, false, false};

builder.setMultiChoiceitems(colors, checkeditems,

new Dialoginterface.OnMultiChoiceClickListener() @Override

pubic void onClick (Dialoginterface dialog, int which, boolean isChecked)

mCheckeditems[which] isChecked;

Первый параметр в **методе setMultiChoiceitems()** — массив значений для списка с флажками, второй параметр — булевый массив состояний флажков списка по умолчанию при вызове диалога.

*Так же, как и для диалога с радиоютопками для диалога с флажками добавляют командные кнопки для ега закрытия.*

**Для примера приложения с вызовом диалогового окна с флажками создайте новый проект и в диалоге Create New Project введите следующие значения:**

* Project name — AlertDialogCheckBoxesApp;
* Application name— AlertDialogCheckBoxes Sample;
* Package name— сот. samples. alertdialogcheckboxes;
* Create Activity — AlertDialogCheckBoxesActivity.

Полный код класса, реализующего деятельность, представлен в листинге:

Файл Alert.DialogCheckBoxesActivity.java

package com.samples.alertdialogcheckboxes;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.Toast;

public class AlertDialogCheckBoxesActivity extends Activity {

private final static int IDD\_COLOR = 0;

final CharSequence[] mColors = {"Red", "Green", "Blue"};

final boolean[] mCheckedItems = {true, false, false};

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final Button callButton = (Button)findViewById(R.id.button);

// добавляем слушатель события для кнопки вызова диалога

callButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

showDialog(IDD\_COLOR);

}

});

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case IDD\_COLOR:

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Pick a color");

builder.setMultiChoiceItems(mColors, mCheckedItems, new DialogInterface.OnMultiChoiceClickListener() {

@Override

public void onClick(DialogInterface dialog, int which, boolean isChecked) {

mCheckedItems[which] = isChecked;

}

});

builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

StringBuilder state = new StringBuilder();

for (int i = 0; i < mColors.length; i++) {

state.append("Color: " + mColors[i] + ", state: ");

if (mCheckedItems[i])

state.append("checked\n");

else

state.append("unchecked\n");

}

Toast.makeText(getApplicationContext(),

state.toString(), Toast.LENGTH\_LONG).show();

}

});

builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dialog.cancel();

Toast.makeText(getApplicationContext(), "Dialog cancel", Toast.LENGTH\_SHORT).show();

}

});

builder.setCancelable(false);

return builder.create();

default:

return null;

}

}

}

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться окно AlertDialog со списком из трех флажков для выбора цвета. При закрытии диалогового окна кнопкой Yes появится всплывающее уведомление, показывающее состояние флажков выбора цвета.



*AlertDialog с флажками*

**AlertDialog со списком**

Чтобы создавать **AlertDialog со списком** выбираемых пунктов, необходимо использовать**метод setitems()**, параметрами которого является массив данных для отображения в списке диалога и ***интерфейс Dialoginterface***. **OnClickListener**, который определяет действие, когда пользователь выбирает элемент списка, например:

CharSequence[] colors {"Red", "Green", "Blue"};

builder.setitems(colors, new Dialoginterface.OnClickListener()

pubic void onClick (Dialoginterface dialog, int item)

//обрабатываем выбор элемента списка *Toast.makeText(getApplicationContext(),*

"Color: + mColors[item], Toast.LENGTH\_SHORT).show();

Для примера приложения с вызовом диалогового окна со списком создайте новый проект и в диалоге *Create New Project*введите следующие значения:

* Project name — AlertDialogListApp;
* Application name — AlertDialog with List Sample;
* Package name — com. samples. alertdialoglist;
* Create Activity — AlertDialogListActivity.

В **классе AlertDialogListActivity** реализуйте последовательность создания диалога со списком, приведенную ранее. Полный код класса представлен в листинге:

Файл разметки AlertDialogListActivity.Java

package com.samples.alertdialoglist;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.Toast;

public class AlertDialogListActivity extends Activity {

private final int IDD\_COLOR = 0;

final CharSequence[] mColors = {"Red", "Green", "Blue"};

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final Button callButton = (Button)findViewById(R.id.button);

// добавляем слушатель события для кнопки вызова диалога

callButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

showDialog(IDD\_COLOR);

}

});

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case IDD\_COLOR:

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Pick a color");

builder.setItems(mColors,

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int item) {

Toast.makeText(getApplicationContext(), "Color: " +

mColors[item], Toast.LENGTH\_SHORT).show();

dialog.cancel();

}

});

builder.setCancelable(false);

return builder.create();

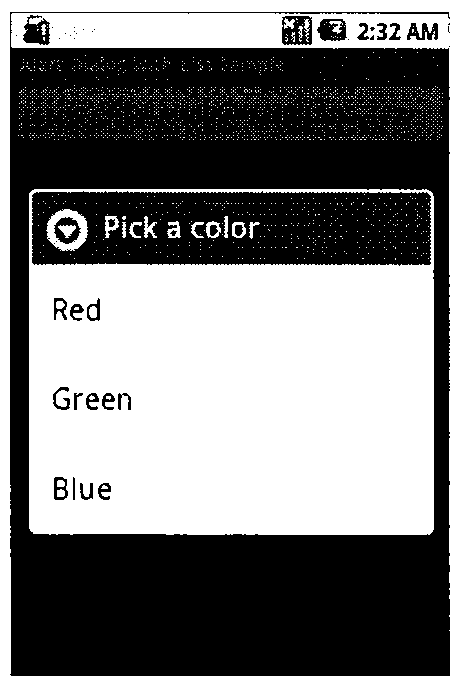
default:

return null;

}

}

}



*AlertDialog со списком*

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться окно AlertDialog со списком из трех пунктов для выбора цвета. При выборе одного из пунктов меню появится всплывающее уведомление, показывающее выбранный цвет.

**DatePickerDialog**

**DatePickerDialog** предназначен для выбора даты пользователем. Обычно перед созданием диалога выбора даты надо получить текущий год, месяц и день из системы. Это можно сделать, создав **экземпляр класса Calendar**и записав дату через **метод get() класса Calendar**в переменные, созданные в нашем классе:

Calendar с Calendar.getlnstance();

mYear c.get(Calendar.YEAR);

mMonth c.get(Calendar.MONTH);

mDay c.get(Calendar.DAY\_OF\_MONTH);

Затем в ***onCreateDialog()*** надо создать ***объект DatePickerDialog*** вызовом конструктора класса:

DatePickerDialog dialog new DatePickerDialog(

this, rnDateSetListener, mYear, mMonth, mday);

Первый параметр, передаваемый в конструктор, — это контекст текущей деятельности,второй — имя метода обратного вызова для обработки события установки даты, остальные параметры содержат дату, которую отобразит запускаемый диалог.

Имя метода обратного вызова для обработки, события установки даты — это реализация вложенного **интерфейса DatePickerDialog**. **OnDateSetListener**, которая, например, может модифицировать передаваемые переменные, хранящие дату:

private DatePickerDialog.OnDateSetListener mDateSetListener

new DatePickerDialog.OnDateSetListener()

{public void onDateSet(DatePicker view,int year,int monthOfYear,int dayOfMonth)

{ mYear year; mMonth monthOfYear; mDay = dayOfMonth;

Вызов диалога производится обычным способом — **через метод showDialog(int dialogid).**

В качестве примера приложения с вызовом диалогового окна установки даты создайте новый проект и в диалоге Create New Project введите следующие значения:

* Project name — DatePickerDialogApp;
* Application name — DatePickerDialog Sample;
* Package name — сот. samples. datepickerdialog;
* Create Activity — DatePickerDialogActivity.

Откройте файл разметки и создайте структуру разметки с текстовым полем и кнопкой вызова подобно листингу:

**Файл разметки main.xml**

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

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

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:id="@+id/text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="28px"

android:textStyle="bold"

android:text=""/>

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Change the date"/>

</LinearLayout>

Полный код класса деятельности, реализующий работу с DatePickerDialog, приведен в листинге:

**Файл DatePickerDialogActivity.Java**

package com.samples.datepickerdialog;

import java.util.Calendar;

import android.app.Activity;

import android.app.DatePickerDialog;

import android.app.Dialog;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.DatePicker;

import android.widget.TextView;

public class DatePickerDialogActivity extends Activity {

private TextView mDateDisplay;

private Button mPickDate;

private int mYear;

private int mMonth;

private int mDay;

static final int DATE\_DIALOG\_ID = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

// capture our View elements

mDateDisplay = (TextView) findViewById(R.id.text);

mPickDate = (Button) findViewById(R.id.button);

// add a click listener to the button

mPickDate.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

showDialog(DATE\_DIALOG\_ID);

}

});

// get the current date

final Calendar c = Calendar.getInstance();

mYear = c.get(Calendar.YEAR);

mMonth = c.get(Calendar.MONTH);

mDay = c.get(Calendar.DAY\_OF\_MONTH);

// display the current date

updateDisplay();

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case DATE\_DIALOG\_ID:

return new DatePickerDialog(this,

mDateSetListener,

mYear, mMonth, mDay);

}

return null;

}

// updates the date we display in the TextView

private void updateDisplay() {

mDateDisplay.setText(

new StringBuilder()

// Month is 0 based so add 1

.append(mMonth + 1).append("-")

.append(mDay).append("-")

.append(mYear).append(" "));

}

// the callback received when the user "sets" the date in the dialog

private DatePickerDialog.OnDateSetListener mDateSetListener =

new DatePickerDialog.OnDateSetListener() {

public void onDateSet(DatePicker view, int year,

int monthOfYear, int dayOfMonth) {

mYear = year;

mMonth = monthOfYear;

mDay = dayOfMonth;

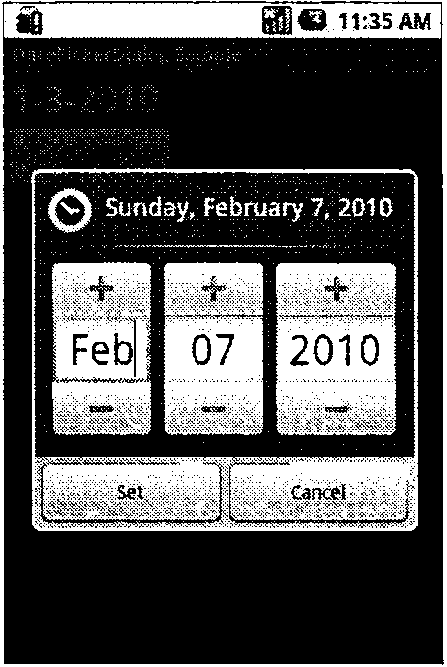
updateDisplay();

}

};

}

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться**окно DatePickerDialog**, в котором пользователь может установить дату. При закрытии диалога возвращаемый результат запишется в текстовое поле в верхней части экрана приложения.



*Диалог DatePickerDialog*

**ProgressDialog**

**ProgressDialog** — подкласс AlertDialog, который представляет собой диалоговое окно с индикатором прогресса. В диалог также можно добавить управляющие кнопки, например для отмены выполняемой задачи.

Для создания диалога с индикатором прогресса **необходимо инициализировать объект ProgressDialog** вызовом ***конструктора класса ProgressDialog*** (*Context*), передав в качестве параметра контекст текущей деятельности:

ProgressDialog progressDialog new ProgressDialog(Activity\_Name.this);

Далее надо установить требуемые свойства диалогового окна, например:

progressDialog.setProgressStyle(ProgressDialog.STYLE\_HORIZONTAL);

progressDialog.setMessage("Loading. progressDialog.setCancelable(false);

Обычно выполнение длительных задач происходит в другом потоке, т. е. в нашем приложении необходимо создать второй поток и сообщать о прогрессе его выполнения назад, в основной поток деятельности через **объект Handler**:

Handler handler new Handler()

public void handleMessage(Message msg)

progressDialog.setProgress(total);

В качестве примера приложения, в котором будет вызываться диалоговое окно с индикатором прогресса, **создайте в Eclipse новый проект** и в диалоге Create New Project введите следующие значения:

* Project name — ProgressDialogApp;
* Application name — ProgressDialog Sample;
* Package name — com. samples. progressdialog;
* Create Activity — ProgressDialogActivity.

В **классе ProgressDialogActivity**реализуйте последовательность создания и задания свойств для диалога, приведенную ранее. Код для запуска и работы второго потока реализован в классе ***SecondThread***, который создается и запускается в методе обратного вызова **onCreateDialog()**класса деятельности.

Полный код **класса ProgressDialogActivity**, реализующего деятельность для приложения, представлен в листинге:

Файл ProgressDialogActivity Java

package com.samples.progressdialog;

import android.app.Activity;

import android.app.Dialog;

import android.app.ProgressDialog;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.Toast;

public class ProgressDialogActivity extends Activity {

static final int IDD\_PROGRESS = 0;

ProgressThread mProgressThread;

ProgressDialog mProgressDialog;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final Button callButton = (Button)findViewById(R.id.button);

callButton.setOnClickListener(new OnClickListener(){

public void onClick(View v) {

showDialog(IDD\_PROGRESS);

}

});

}

protected Dialog onCreateDialog(int id) {

switch(id) {

case IDD\_PROGRESS:

mProgressDialog = new ProgressDialog(ProgressDialogActivity.this);

mProgressDialog.setProgressStyle(ProgressDialog.STYLE\_HORIZONTAL);

//progressDialog.setProgressStyle(ProgressDialog.STYLE\_SPINNER);

mProgressDialog.setMessage("Loading...");

mProgressThread = new ProgressThread(handler);

mProgressThread.start();

return mProgressDialog;

default:

return null;

}

}

// Define the Handler that receives messages from the thread and update the progress

final Handler handler = new Handler() {

public void handleMessage(Message msg) {

int total = msg.getData().getInt("total");

mProgressDialog.setProgress(total);

if (total >= 100){

dismissDialog(IDD\_PROGRESS);

mProgressThread.setState(ProgressThread.STATE\_DONE);

Toast.makeText(getApplicationContext(), "Task is finished", Toast.LENGTH\_SHORT).show();

}

}

};

}

**Код класса SecondThread для работы с потоком:**

Файл SecondThread.java

package com.samples.progressdialog;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.util.Log;

public class ProgressThread extends Thread {

Handler mHandler;

final static int STATE\_DONE = 0;

final static int STATE\_RUNNING = 1;

int mState;

int mTotal;

ProgressThread(Handler h) {

mHandler = h;

}

public void run() {

mState = STATE\_RUNNING;

mTotal = 0;

while (mState == STATE\_RUNNING) {

try {

Thread.sleep(100);

} catch (InterruptedException e) {

Log.e("ERROR", "Thread Interrupted");

}

Message msg = mHandler.obtainMessage();

Bundle b = new Bundle();

b.putInt("total", mTotal);

msg.setData(b);

mHandler.sendMessage(msg);

mTotal++;

}

}

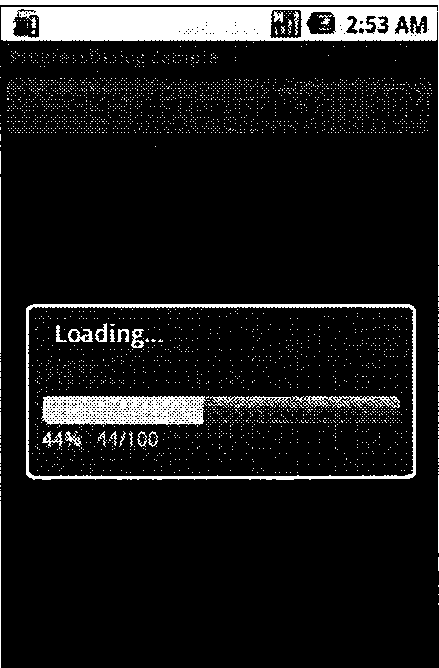
public void setState(int state) {

mState = state;

}

}

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться диалоговое окно с индикатором прогресса и надписями, отображающими степень завершения задачи:



*ProgressDialog со стилем STYLE\_HORIZONTAL*

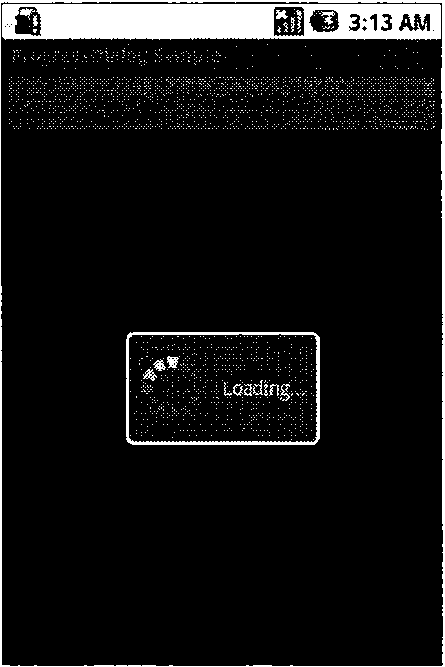
**Метод setProgressStyle()** устанавливает стиль диалога. Стили задаются константами, определенными в ***классе ProgressDialog:***

* style\_horizontal;
* style\_spinner.

Попробуйте поменять стиль шкалы индикатора прогресса в приложении. Измените в листинге строку с вызовом метода **setProgressStyle ():**

progressDialog.setProgressStyle(ProgressDialog.STYLE\_SPINNER);

Измените код в классе и запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться ***окно ProgressDialog***, но уже не с линейным, а с круговым индикатором прогресса:



*ProgressDialog со стилем STYLE\_SPINNER*

**TimePickerDialog**

**TimePickerDialog** предназначен для выбора даты пользователем. В целом процедура создания диалога выбора времени аналогична созданию **DatePickerDialog** с небольшими отличиями в деталях.

Если необходимо прочитать текущее системное время, также используется ***класс Calendar***, например:

Calendar с Calendar.getlnstance();

mHour c.get(Calendar.HOUR\_OF\_DAY);

mMinute c.get(Calendar.MINUTE);

Затем в **onCreateDialog ()**надо создать **объект TimePickerDialog** вызовом конструктора класса:

TimePickerDialog dialog new TimePickerDialog (

this, timeSetListener, mHour, mMinute, false) ;

В последнем параметре конструктора указывается формат отображения времени 12часовой (АМ/РМ) — false или 24часовой — true.

Имя метода обратного вызова для обработки события установки даты — это реализация вложенного **интерфейса TimePickerDialog**. **OnTimeSetListener**, например:

private TimePickerDialog.OnTimeSetListener mTimeSetListener new TimePickerDialog. OnTimeSetListener () public void onTimeSet(TimePicker view, int hourOfDay, int minute) { mHour hourOfDay; mMinute minute;

В качестве примера приложения с вызовом диалога установки времени создайте новый проект и в *диалоге Create New Project*введите следующие значения:

* Project name — TimePickerDialogActivi tyApp;
* Application name — TimePickerDialogActivity Sample;
* Package name — com. samples. timepickerdialog;
* Create Activity — TimePickerDialogActivity.

Структура файла разметки аналогична листингу  для DatePickerDialog. Код класса деятельности TimePickerDialogActivity приведен в листинге ниже:

**Файл TimePickerDialogActivity.java**

package com.samples.timepickerdialog;

import java.util.Calendar;

import android.app.Activity;

import android.app.Dialog;

import android.app.TimePickerDialog;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.TimePicker;

public class TimePickerDialogActivity extends Activity {

private TextView mTimeDisplay;

private Button mPickTime;

private int mHour;

private int mMinute;

static final int TIME\_DIALOG\_ID = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

mTimeDisplay = (TextView)findViewById(R.id.text);

mPickTime = (Button)findViewById(R.id.button);

mPickTime.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

showDialog(TIME\_DIALOG\_ID);

}

});

// читаем текущее время

final Calendar c = Calendar.getInstance();

mHour = c.get(Calendar.HOUR\_OF\_DAY);

mMinute = c.get(Calendar.MINUTE);

// обновляем вывод времени на экран

updateDisplay();

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case TIME\_DIALOG\_ID:

return new TimePickerDialog(this, mTimeSetListener, mHour, mMinute, false);

}

return null;

}

// обновление вывода времени на экран в поле TextView

private void updateDisplay() {

mTimeDisplay.setText(

new StringBuilder()

.append(pad(mHour)).append(":")

.append(pad(mMinute)));

}

private TimePickerDialog.OnTimeSetListener mTimeSetListener =

new TimePickerDialog.OnTimeSetListener() {

public void onTimeSet(TimePicker view, int hourOfDay, int minute) {

mHour = hourOfDay;

mMinute = minute;

updateDisplay();

}

};

private static String pad(int c) {

if (c >= 10)

return String.valueOf(c);

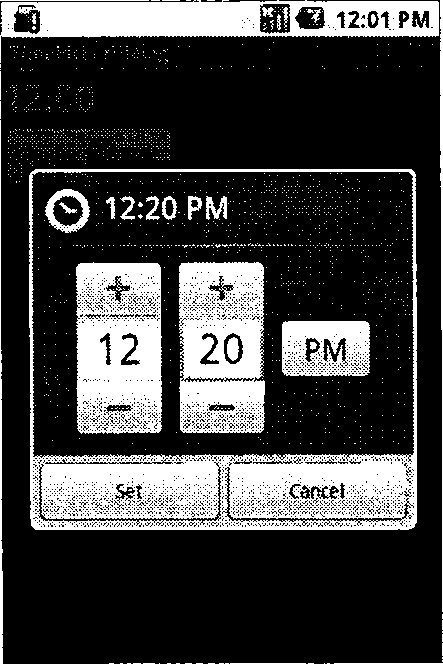
else

return "0" + String.valueOf(c);

}

}

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться окно TirnePickerDialog, в котором пользователь может установить время. При закрытии диалога возвращаемый результат запишется в текстовое поле в верхней части экрана приложения:



*Диалог TirnePickerDialog*

# Создание диалоговых окон

Диалоговое окно всегда создается и отображается как часть находящейся в фокусе деятельности. Диалоги обычно создают внутри метода обратного вызова **onCreateDialog(),**который необходимо реализовать в вашей деятельности. При этом система Androidавтоматически управляет состоянием диалога (или нескольких диалогов) и прикрепляет их к текущей деятельности, фактически делая ее владельцем каждого диалога.

Можно создавать диалог без***onCreateDialog()***, например в обработчике нажатия кнопки вызова диалога, но тогда он не будет присоединен к текущей деятельности. Чтобы прикрепить его к деятельности, необходимо вызвать***метод setOwnerActivitу()***, передав ему в качестве параметра текущую деятельность.

Для отображения диалогового окна необходимо вызвать **метод showDialog()** и передать ему в качестве параметра идентификатор диалога (константа, которую надо объявить в коде программы), который вы хотите отобразить. Например:

private static final int IDD\_EXIT О;

showDialog(IDD\_EXIT);

Когда диалог вызывается впервые, **система Android вызывает onCreateDialog()**из текущей деятельности. В **onCreateDialog()** передается тот же самый идентификатор, который передавался в **showDialog()**. После того как вы создали диалог, вы возвращаете объект в конце метода.

Если в деятельности должны вызываться несколько различных диалоговых окон, сначала необходимо определить целочисленный идентификатор для каждого диалога, например:

private static final int IDD ALERT О;

private static final int IDD\_EXIT

Эти идентификаторы потом можно использовать в вызове метода **showDialog()** и в обработчике события **onCreateDialog()** в операторе **switch:**

protected Dialog onCreateDialog(int id) Dialog dialog;

switch(id) { case IDD\_ALERT:

break;

case IDD EXIT:

default:

dialog null;

return dialog;

Внутри **оператора switch**описывается сама процедура создания диалоговых окон, которая различна для каждого типа диалоговых окон и будет описана в следующих разделах этой главы.

Перед отображением диалогового окна Android вызывает дополнительный метод обратного вызова **onPrepareDialog (int, Dialog)**. Если требуется перед каждым вызовом диалогового окна изменять его свойства (например, текстовое сообщение или количество кнопок), это можно реализовать внутри этого метода. В этот метод передают идентификатор диалога и сам**объект Dialog**, который был создан в **методе обратного вызова onCreateDialog ().**

**Создание собственных диалогов**

Если есть необходимость в создании оригинального дизайна для диалога, можно создать собственную разметку для диалогового окна. После создания разметки необходимо передать корневой объект разметки в код создания диалога.

Чтобы получить корневой объект разметки, **используется класс Layoutinflater**. Этот класс нужен для преобразования XMLразметки в соответствующие объекты View в коде программы.

Сначала необходимо **инициализировать объект Layoutinflater** вызовом **метода getLayoutinfiater()**, затем получить корневое представление ***методом inflate (int, ViewGroup)***, где первый параметр— идентификатор ресурса схемы размещения, второй — идентификатор корневого представления разметки:

Layoutinflater inflater getLayoutlnflater();

View layout inflater.inflate(R.layout.custom\_layout,

(ViewGroup)findViewByid(R.id.toast\_layout));

Получив корневое представление, можно **методом findViewByid()**инициализировать все дочерние представления в разметке и задать для них информационное наполнение. Например, если в разметке определены ***виджеты TextView и ImageView***, код может выглядеть так:

TextView text (TextView)layout.findViewById(R.id.text);

text.setText("Are you sure you want to exit?");

ImageView image (ImageView)layout. findViewById(R.id. image);

image.setlmageResource(R.drawable.android3d);

Далее создается объект AlertDialog. Builder и методом setView () для него устанавливается полученная ранее разметка:

AlertDialog.Builder builder new AlertDialog.Builder(this);

builder. setView(layout);

Остальная инициализация свойств диалога и работа с ним в коде программы аналогична работе **со стандартным AlertDialog.**

В качестве примера приложения с собственным диалоговым окном создайте новый проект и в диалоге *Create New Project* введите следующие значения:

* Project name — CustomDialogApp;
* Application name — CustomDialog Sample;
* Package name — сот. samples. customdialog;
* Create Activity — CustomDialogActivity.

Создайте также отдельный файл разметки для диалогового окна, как показано в листинге:

**Файл разметки custom\_layout.xml**

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

android:id="@+id/toast\_layout"

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:padding="10dp"

android:background="#DAAA">

<ImageView android:id="@+id/image"

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:layout\_marginRight="10dp"/>

<TextView android:id="@+id/text"

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:textColor="#FFF"/>

</LinearLayout>

В **классе CustomDialogActivity** реализуйте создание и вызов диалогового окна с собственной разметкой, как было показано ранее в этом разделе. Полный ***код класса CustomDialogActivity*** показан в листинге:

**Файл CustomDialogActivlty.java**

package com.samples.customdialog;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.TextView;

public class CustomDialogActivity extends Activity {

private final static int IDD\_CUSTOM = 0;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final Button callButton = (Button)findViewById(R.id.button);

// добавляем слушатель события для кнопки вызова диалога

callButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

showDialog(IDD\_CUSTOM);

}

});

}

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case IDD\_CUSTOM:

LayoutInflater inflater = getLayoutInflater();

View layout = inflater.inflate(

R.layout.custom\_layout, (ViewGroup)findViewById(R.id.toast\_layout));

TextView text = (TextView)layout.findViewById(R.id.text);

text.setText("Are you sure you want to exit?");

ImageView image = (ImageView) layout.findViewById(R.id.image);

image.setImageResource(R.drawable.android3d);

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setView(layout);

builder.setMessage("This is a custom dialog!");

builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

CustomDialogActivity.this.finish();

}

});

builder.setNegativeButton("No", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dialog.cancel();

}

});

builder.setCancelable(false);

return builder.create();

default:

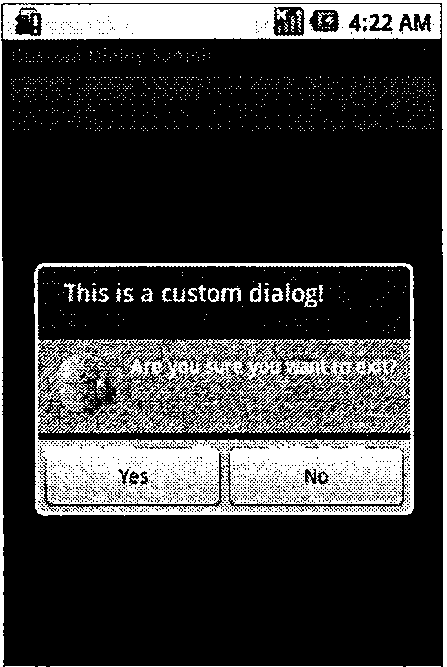
return null;

}

}

}

Запустите проект на выполнение. При нажатии кнопки вызова диалога должно появиться окно AlertDialog с кнопками Yes и No и с созданным нами нестандартным информационным наполнением. При закрытии диалога кнопкой Yes приложение закончит работу.



*Диалог с собственным дизайном*