Лабораторная №5
Создание диалогов.

В этом уроке вы:

- Научитесь создавать различные виды стандартных диалогов
- Научитесь создавать диалоги с произвольным содержимым
- Научитесь получать данные из диалогов, введенные пользователем
- Научитесь выводить диалоги двумя способами
- Познакомитесь с созданием главного меню окна и контекстного меню для компонента управления

1. Диалоги в Android

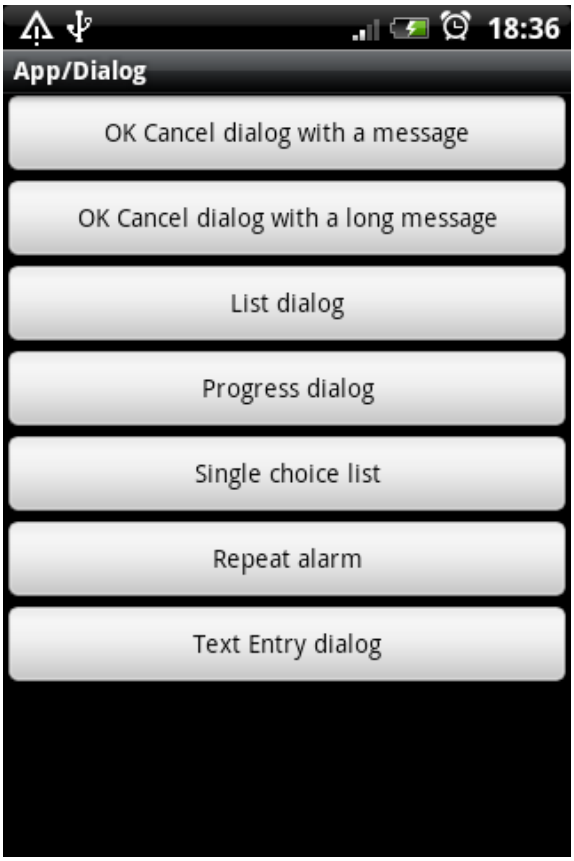
Диалог – это небольшое окно, всплывающее поверх остальных окон приложения. Диалог используется обычно для следующих целей:

- Сообщить пользователю какую-либо информацию, например, о недоступности сети, внутренней ошибке. Обычно, в этом случае на диалоге появляется единственная кнопка «ОК», закрывающая диалог.
- Запросить у пользователя подтверждение какого-либо действия, в этом случае на диалоге имеется 2-3 кнопки: «ОК», «Отмена», иногда «Справка».
- Запросить у пользователя какую-либо информацию, например, ввод логина и пароля.
- т.д.

Например, следующим образом меняется внешний вид приложения при появлении диалога:

В любом случае, вывод диалога на экран – действие, прерывающее жизненный цикл окна приложения, поверх которого он выводится – вызывается метод onPause() класса окна.

При этом это окно недоступно для пользователя, пока диалог не будет закрыт и фокус не будет возвращен вызывающему окну.



Активное окно

Окно встало на паузу, фокус ввода передался диалогу.

Внешний вид диалога в Android может быть абсолютно любым – не существует никакого минимального или максимального набора составляющих элементов окна диалога. В общем случае, можно в качестве содержимого диалога передать произвольный layout.

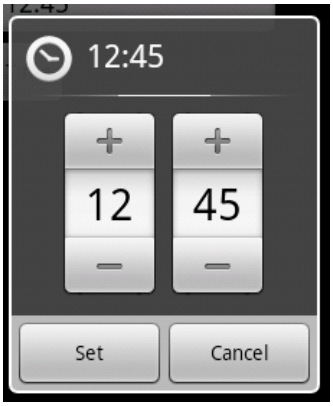
2. Класс Dialog

Все диалоги в Android являются наследниками класса Dialog. Непосредственно создавать объект класса Dialog нам не придется, мы всегда будем пользоваться его известными наследниками:

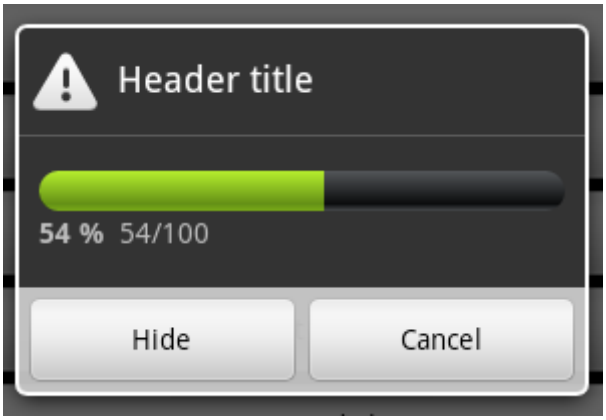
- **AlertDialog.** Назван диалогом «предупреждения», хотя это необязательно так. Это может быть любой диалог, содержащий любую информацию, какие-то кнопки. Возможно, AlertDialog предложит пользователю выбрать какой-то вариант из предложенных, например, день недели, на который установить событие.

- **DatePickerDialog.** Стандартный диалог выбора даты, встроенный в ОС Android. Обычно, содержит барабан и сформированную дату.

- **TimePickerDialog.** Стандартный диалог выбора времени, встроенный в ОС Android. Обычно, содержит барабан и сформированное время.



- **ProgressDialog.** Диалог, содержащий полосу прогресса и некоторую информацию о текущем прогрессе выполнения какой-либо асинхронной длительной операции (скачивание файлов, вычисления, обращение к базе данных, т.д.)



Давайте научимся создавать и выводить каждый из них.

3. Класс AlertDialog.

Как уже говорилось, класс AlertDialog используется для вывода любых диалогов. Если диалог, который мы хотим вывести, явно не походит ни к одному из описанных выше типов (ProgressDialog, DatePickerDialog, т.д.), мы принимаем решение пользоваться классом AlertDialog.

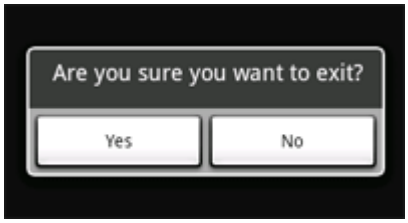
Для конструирования диалогов типа AlertDialog используется специальный класс AlertDialog.Builder. Методами этого класса мы создаем внешний вид диалога, задаем заголовок, подготавливаем внешний вид компонентов, т.д. Последующий вызов метода create() этого класса возвращает нам объект класса Dialog, готовый к выводу. Пример:

```
// Создем builder для конструирования AlertDialog'а. Аргумент конструктора -
// контекст окна, на котором будем выводить диалог.
AlertDialog.Builder builder = new AlertDialog.Builder(this);

// Конструируем диалог
builder.setMessage("Are you sure you want to exit?");
builder.setCancelable(false);

// Устанавливаем слушатель события "Нажали кнопку Yes"
builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
                MainActivity.this.finish();
        }
});

// Устанавливаем слушатель события "Нажали кнопку No"
builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
        }
});

// Создали объект AlertDialog, готовый к выводу на экран
AlertDialog alert = builder.create();
```

Давайте выведем диалог на экран. Для этого вызовем метод show() класса AlertDialog (который наследуется от класса Dialog).

```
alert.show();
```

Результат будет следующим:



Диалог может содержать три стандартные кнопки, именуемые PostiveButton, NegativeButton, NeutralButton. Не обязательно воспринимать эти названия как определяющие действия, выполняемые при нажатии этих кнопок. Это всего лишь избежание наименований button1, button2, button3.

Давайте познакомимся с некоторыми методами класса AlertDialog.Builder, которые позволят нам конструировать внешний вид диалога:

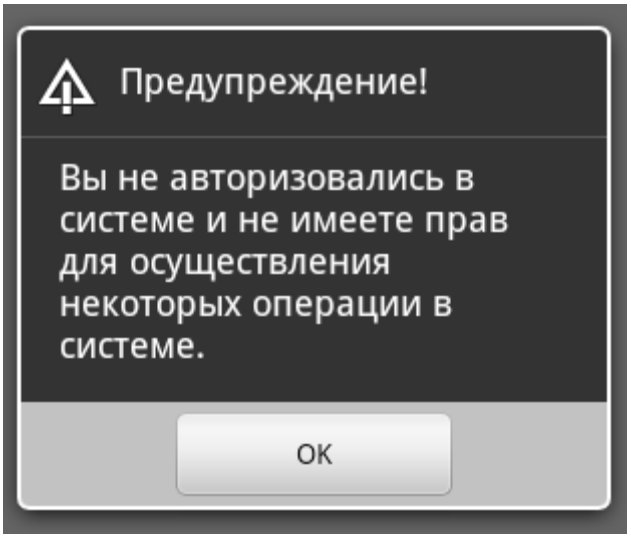
public AlertDialog.Builder setMessage (CharSequence message)	Задает выводимое в диалоге сообщение message.
public AlertDialog.Builder setMessage (int messageId)	Задает выводимое в диалоге сообщение, которое берет из ресурсов типа string с идентификатором messageId.
public AlertDialog.Builder setTitle (CharSequence title)	Задает заголовок диалога
public AlertDialog.Builder setTitle (int titleId)	Задает заголовок окна из ресурса string с идентификатором titleId
public AlertDialog.Builder setCancelable (boolean cancelable)	Если cancellable==false, то диалог нельзя закрыть нажатием кнопки Back.
public AlertDialog.Builder setPositiveButton (CharSequence text, DialogInterface.OnClickListener listener)	Создает кнопку с надписью text и вызывает метод onClick() слушателя listener.
public AlertDialog.Builder setNegativeButton (CharSequence text, DialogInterface.OnClickListener listener)	Создает кнопку с надписью text и вызывает метод onClick() слушателя listener.
public AlertDialog.Builder setNeutralButton (CharSequence text, DialogInterface.OnClickListener listener)	Создает кнопку с надписью text и вызывает метод onClick() слушателя listener.
public AlertDialog.Builder setIcon (int iconId)	Устанавливает иконку в диалоге из ресурса drawable с идентификатором iconId.

Давайте опробуем некоторые из этих методов:

```
// Создем builder для конструирования AlertDialog'a
AlertDialog.Builder builder = new AlertDialog.Builder(this);

// Конструируем диалог
builder.setMessage("Вы не авторизовались в системе и не имеете прав для
осуществления некоторых операции в системе.");

// Диалог закрывается по кнопке Back на устройстве
```

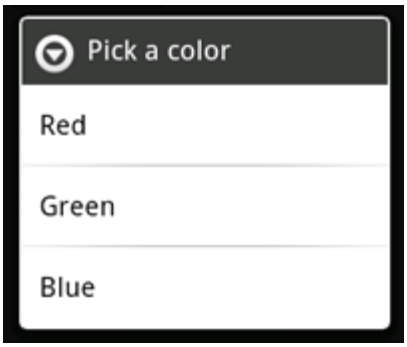
```
builder.setCancelable(true);

// Устанавливаем слушатель события "Нажали кнопку Yes"
builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
                MainActivity.this.finish();
        }
});

builder.setTitle("Предупреждение!");
builder.setIcon(android.R.drawable.ic_dialog_alert);

// Создали объект AlertDialog, готовый к выводу на экран
AlertDialog alert = builder.create();
alert.show();
```

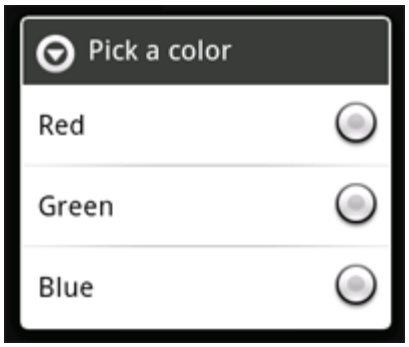
Результат выполнения:



4. AlertDialog с выбором одного варианта.

Можно создать диалог, в котором пользователю предлагается выбор одного из предложенных вариантов, например:





Код:

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

AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Pick a color");
builder.setSingleChoiceItems(items, -1, new DialogInterface.OnClickListener()
{
        public void onClick(DialogInterface dialog, int item) {
                Toast.makeText(getApplicationContext(), items[item],
Toast.LENGTH_SHORT).show();
}
});

AlertDialog alert = builder.create();
alert.show();
```

Метод public AlertDialog.Builder setSingleChoiceItems (CharSequence[] items, int checkedItem, DialogInterface.OnClickListener listener) добавляет в диалог компонент для выбора одного элемента из массива items. checkedItem – индекс элемента, выбранного по умолчанию (-1 для отсутсвия выбора по умолчанию), listener – слушатель события «выбор сделан», вызывающий у себя метод onClick(dialog, item), где item – индекс выбранного элемента.

5. AlertDialog с выбором нескольких вариантов.

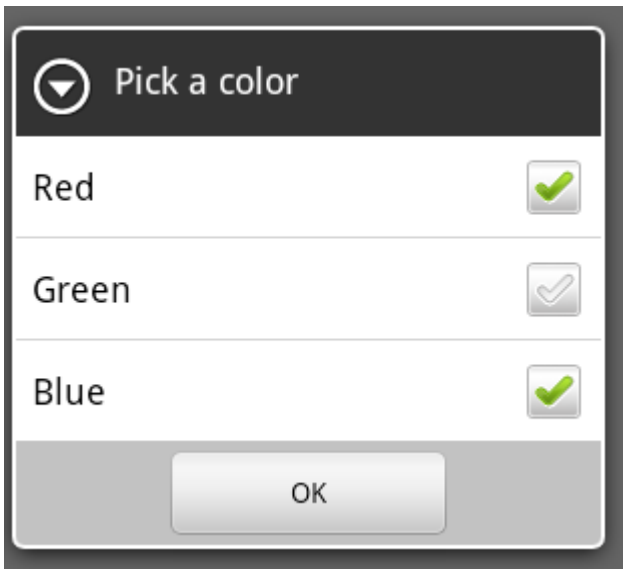
Можно создать диалог, в котором пользователю предлагается выбор нескольких из предложенных вариантов, например:

```
final CharSequence[] items = {"Red", "Green", "Blue"};
boolean[] defaultChoise = {true, false, true};
builder.setTitle("Pick a color");
builder.setMultiChoiceItems(items, defaultChoise, new
OnMultiChoiceClickListener() {

        public void onClick(DialogInterface dialog, int which, boolean
isChecked) {
                Log.d("TAG", items[which]+": "+isChecked);
        }
});

AlertDialog alert = builder.create();
alert.show();
```

Метод public AlertDialog.Builder setMultiChoiceItems (CharSequence[] items, boolean[] checkedItems, DialogInterface.OnClickListener listener) добавляет в диалог компонент для выбора нескольких элементов из массива items.

defaultChoise – массив длиной, равной длине массива items, при этом если defaultChoise[i]==true, то вариант items[i] по умолчанию выбран.

Для каждого выбранного варианта вызывается метод onClick() слушателя событий от диалога listener. При этом which – индекс выбранного элемента. Например, следующим образом отработает вышеприведённый код:



LogCat:

08-15 21:19:13.030: D/TAG(8034): Red: true

08-15 21:19:13.570: D/TAG(8034): Blue: true

Каждый раз при выборе элемента будет вызываться метод onClick() слушателя. Поэтому каждый раз мы увидим новую строку в LogCat.

6. Установка произвольного содержимого диалога.

Мы научились конструировать диалог из стандартных компонентов – заголовка, текстового содержимого, кнопок для принятия решения.

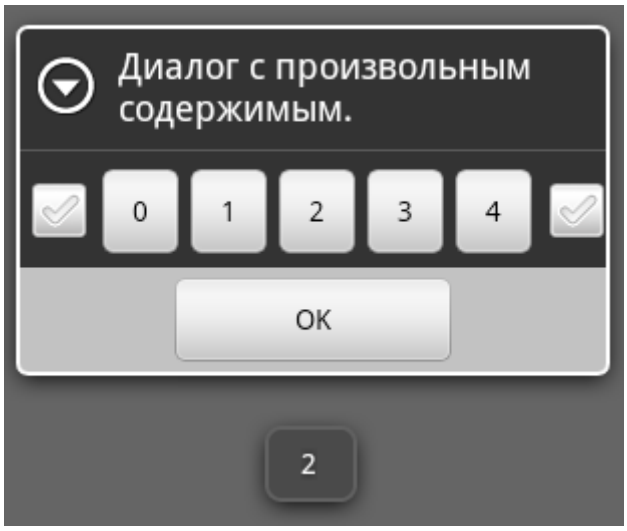
Как же замечательно, что и заголовком, и содержимым диалога могут быть произвольные компоненты View. Итак, еще два метода AlertDialog.Buider:

public AlertDialog.Builder setCustomTitle (View customTitleView)	Устанавливает вид заголовка в произвольное View. Таким образом, можно отказаться от стандартного заголовка, содержащего иконку и текст.
public AlertDialog.Builder setView (View view)	Устанавливает содержание диалога в произвольное View. Таким образом, можно использовать любой layout как содержимое диалога.

Давайте рассмотрим пример:

```
// Создем builder для конструирования AlertDialog'a
AlertDialog.Builder builder = new AlertDialog.Builder(this);

// Конструируем диалог
//builder.setMessage("Вы не авторизовались в системе и не имеете прав для
осуществления некоторых операции в системе.");
builder.setCancelable(true);

// Устанавливаем слушатель события "Нажали кнопку Yes"
builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
                MainActivity.this.finish();
        }
});

builder.setTitle("Диалог с произвольным содержимым.");

// Формируем View для вставки в содержимое диалога
LinearLayout l = new LinearLayout(this);
l.addView(new CheckBox(this));
for(int i=0;i<5;i++) {
        final Button b = new Button(this);
        b.setText(Integer.toString(i));
        b.setOnClickListener(new OnClickListener() {

                public void onClick(View v) {
                        Toast.makeText(b.getContext(), b.getText(),
Toast.LENGTH_SHORT).show();
                }

        });

l.addView(b);
}
l.addView(new CheckBox(this));
```

```
// Устанавливаем View в содержимое диалога
builder.setView(l);

// Создали объект AlertDialog, готовый к выводу на экран
AlertDialog alert = builder.create();
alert.show();
```



В этом примере мы создали LinearLayout и заполнили его различными View, некоторым даже назначили поведение при нажатии. Таким образом, мы видим, что можно запрограммировать layout и поведение содержимого View, а затем при помощи метода setView() установить его в диалог.

Но чаще мы не будем таким образом создавать View для отображения его в содержимом диалога. Гораздо интереснее нам будет создать отдельный файл дизайна layout, и из него сформировать View для отображения.

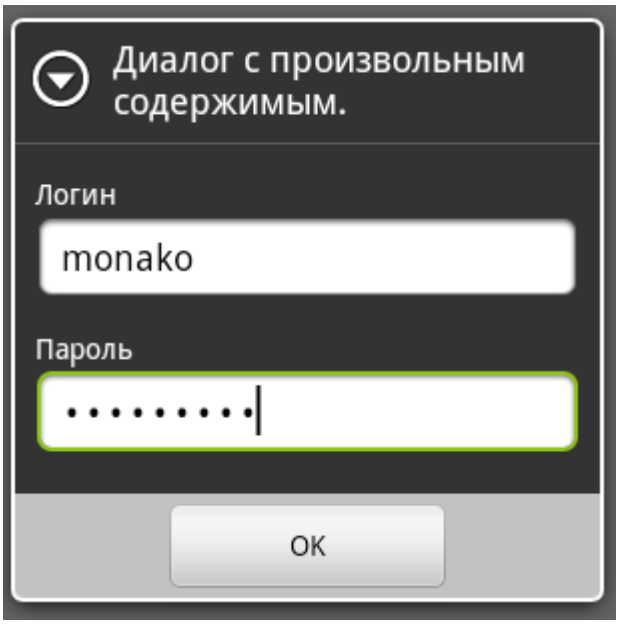
Предположим мы создали файл mydialog.xml, описывающий внешний вид содержимого диалога. Наша задача – сформировать из этого файла объект класса View. Для этого воспользуемся следующими строками:

```
View view = getLayoutInflater().inflate(R.layout.mydialog, null);
builder.setView(view);
```

Метод Activity getLayoutInflater() вызывает объект специального класса LayoutInflater, предназначенного для создания объекта класса View по его XML-описанию. Вы много раз воспользуетесь этим классом в дальнейшем.

Метод inflate(int layoutId) создает View по описанию в layout-ресурсе с идентификатором lalyoutId.

Результат выполнения этого кода:



Как же в этом случаем при нажатии кнопки ОК считать содержимое полей и выполнить с ними какие-то действия? Довольно просто и понятно:

```
// Создем builder для конструирования AlertDialog'a
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Диалог с произвольным содержимым.");

// Формируем View для вставки в содержимое диалога
final View view = getLayoutInflater().inflate(R.layout.mydialog, null);
builder.setView(view);
builder.setPositiveButton("OK", new OnClickListener() {

        public void onClick(DialogInterface dialog, int which) {
                EditText edLogin = (EditText) view.findViewById(R.id.edLogin);
                EditText edPassword = (EditText)
view.findViewById(R.id.edPassword);

                String login = edLogin.getText().toString();
                String password = edPassword.getText().toString();

                // делаем что-то с login и password
        }

});

// Создали объект AlertDialog, готовый к выводу на экран
AlertDialog alert = builder.create();
alert.show();
```

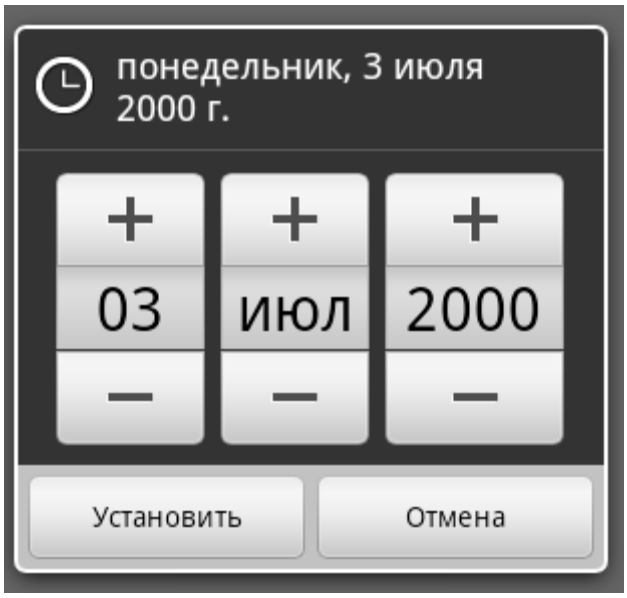
Аналогично, можно установить произвольный View в заголовок диалога при помощи метода setCustomTitle(View view).

7. DatePickerDialog – диалог выбора даты

Рассмотрим способ вывода диалога для выбора даты:

```
DatePickerDialog dpd = new DatePickerDialog(this, new OnDateSetListener() {

                public void onDateSet(DatePicker view, int year, int
monthOfYear, int dayOfMonth) {
                // этот код отработает после нажатия кнопки «Установить»
                Log.d("TAG", dayOfMonth+"."+monthOfYear+"."+year);
                }

}, 2000, 06, 3); // выбор по умолчанию - 3.06.2000

dpd.show();
```

Результат:



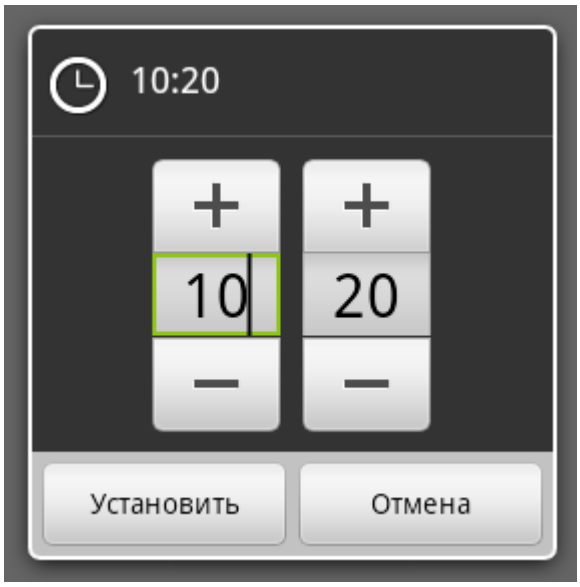
Как видно, мы используем конструктор класса DatePickerDialog, вторым параметром которого является listener, у которого вызывается метод onDateSet() после нажатия кнопки «Установить». Параметры этого метода – тройка целых чисел, определяющих выбранную дату.

8. TimePickerDialog – диалог выбора времени

Рассмотрим способ вывода диалога для выбора времени:

```
TimePickerDialog tpd = new TimePickerDialog(this, new OnTimeSetListener() {

        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                                Log.d("TAG", hourOfDay+":"+minute);
                        }
        }, 10, 20, true);


tpd.show();
```

Результат:



Как видно, мы используем конструктор класса TimerPickerDialog, вторым параметром которого является listener, у которого вызывается метод onTimeSet() после нажатия кнопки «Установить». Параметры этого метода – пара целых чисел, определяющих выбранное время. Последний параметр логического типа определяет, будет ли использован 24-хчасовой формат времени.

9. Отображение диалога.

Мы уже познакомились с одним из способов создания и отображения диалога – метод show() класса Dialog. Этот метод хорошо работает. Но если вы перевернете устройство – диалог исчезнет, так как произойдет пересоздание объекта окна. Чтобы диалог не исчезал – добавьте атрибут android:configChanges=”orientation” тегу <activity> в файле AndroidManifest.xml. Можно на этом ограничиться рассказом о способах вызова диалога, но добавление этого тега не рекомендовано Android. Есть рекомендованные “Android way”-способы создания диалогов и их вывода на экран.

Чтобы вызывать диалог поверх Activiy, вызовите метод Activity showDialog(int id). Этот метод вызовет специальный метод класса Activiy - onCreateDialog(int id), в который передаст целое число, являющееся идентификатором диалога. Пример:

```
private static final int DIALOGID_TIMEPICKER = 10;
private static final int DIALOGID_ERROR = 11;


public void bGetTimeButtonClick(View v) {
        showDialog(DIALOGID_TIMEPICKER);
}

public void readFile(File f) {
        try {
                FileInputStream fis = new FileInputStream(f);
                // .....
        } catch (Exception e) {
                showDialog(DIALOGID_ERROR);
        }
}

// Этот метод будет вызван после showDialog(id), если диалог с
идентификатором id еще не был создан
@Override
protected Dialog onCreateDialog(int id) {
        Dialog dialog;

        switch (id) {
        case DIALOGID_TIMEPICKER:
                // если вызов был showDialog(DIALOGID_TIMEPICKER);
                dialog = createTimePickerDialog();
                break;
        case DIALOGID_ERROR:
// если вызов был showDialog(DIALOGID_ERROR);
                dialog = createErrorDialog();
                break;
        default:
                dialog = super.onCreateDialog(id);
        }

        return dialog;
}

// Этот метод будет вызван всегда

@Override
protected void onPrepareDialog(int id, Dialog dialog) {
        switch (id) {
        case DIALOGID_TIMEPICKER:
        break;
```

```
        case DIALOGID_ERROR:
                ((AlertDialog)dialog).setMessage(msg);
                break;
        default:
                super.onCreateDialog(id);
        }

}

private Dialog createErrorDialog() {
        AlertDialog.Builder b = new AlertDialog.Builder(this);
        b.setTitle("Error");
        b.setMessage(msg);
        return b.create();
}

private Dialog createTimePickerDialog() {
        Calendar c = Calendar.getInstance();
        TimePickerDialog tpd = new TimePickerDialog(this,
                        new OnTimeSetListener() {

                                public void onTimeSet(TimePicker view, int
hourOfDay,
                                                int minute) {
                                        Log.d("TAG", hourOfDay + ":" + minute);
                                }
                        }, c.get(Calendar.HOUR_OF_DAY), Calendar.MINUTE, true);

        return tpd;
}

...
```

После вызова showDialog(id) вызывается метод onCreateDialog(id), где объект класса Dialog создается и возвращается системе с определенным идентификатором id. После этого сразу же вызывается метод onPrepareDialog(id, dialog), в котором мы обновляем содержание диалога. При последующих вызовах showDialog(id) если диалог с идентификатором id уже создавался, метод onCreateDialog(id) вызываться не будет, а сразу же будет вызван onPrepareDialog(id, dialog), в который передастся прошлая версия диалога и где она может быть обновлена.

Этот механизм рекомендован Android, он регистрирует диалог в специальном списке окна и следит за его отображением после пересоздании окна. Поэтому при смене ориентации экрана диалог не исчезнет, для этого не нужно будет добавлять дополнительных атрибутов в тег <activity> в AndroidManifest.xml.

Чтобы уничтожить диалог с идентификатором id, удалив его из зарегистрированных, вызовите метод removeDialog(id). Теперь при следующем вызове showDialog(id) вновь отработает метод onCreateDialog(id).

ЗАМЕЧАНИЕ АВТОРА: в моей практике я перестал использовать этот механизм для отображения диалогов и начал применять метод show() класса Dialog. Я всегда добавляю атрибут android:configChanges=”orientation” тегу <activity> в AndroidManifest.xml. Это гораздо удобнее и не вызывает проблем.

10. «Best Practices» по отображению диалогов.

Хочу оговориться, что к этим техникам по организации вывода диалогов я пришел через некоторое время и считаю их наиболее удобными для использования в больших приложениях. Я всегда отталкиваюсь от соображений логичности, простоты, гибкости кода и иногда пренебрегаю рекомендованными методологиями в программировании, тем не менее, эти техники работают на ура =).

Давайте подумаем, как бы нам хотелось вызвать диалог с сообщением об ошибке.

Мне бы понравилось так:

```
showErrorDialog("В приложении произошла ошибка!").
```

Также хочу заметить, что многие диалоги в приложении будут повторяться на разных окнах.

Например, если наше приложение работает с сетью и, зайдя в очередное окно, начиная обращаться к серверу, обнаруживает отсутствие соединения, оно должно вывести один и тот же диалог с сообщением об ошибке.

Или другой пример: у нас есть операции, совершение которых требует ввода пароля. В этом случае каждый раз мы будем выводить диалог с запросом пароля, обрабатывать его и принимать решение о последующих действиях.

Итак, я рекомендую иметь специальный класс Dialogs.java, где будут собраны статические методы по созданию диалогов различного вида.

```
public class Dialogs {

        // Диалог с сообщением об ошибке.
        public static Dialog createErrorDialog(Context c, String title, String msg)
{
                AlertDialog.Builder builder = new AlertDialog.Builder(c);
                builder.setTitle(title);
                builder.setMessage(msg);
                return builder.create();
        }

}
```

Вызов такого метода я бы организовал так:

```
public void showErrorDialog(String title, String msg) {
        Dialogs.createErrorDialog(this, title, msg).show();
}
```

Предположим, мы выводим диалог с целью получить от пользователя какую-то информацию и на основе ее принять решение о последующих действиях.

В таком случае прибегнем к следующей методологии: создадим слушатель события «получено значение» и будем заставлять отрабатывать его единственный метод, передавая ему информацию, которую ввел пользователь.

Для этого создадим интерфейс ValueSetListener:

```
public interface ValueSetListener<T> {

        public void onValueSet(T value);

}
```

Давайте добавим в класс Dialogs метод, который вернет нам объект диалога для ввода пароля:

```
public class Dialogs {

…….

        public static Dialog createInputPasswordDialog(Context c, String
title,
                        String prompt, final ValueSetListener<String> listener)
{
                AlertDialog.Builder builder = new AlertDialog.Builder(c);
                builder.setTitle(title);
                builder.setMessage(prompt);

                // Добавляем поле ввода пароля
                final EditText edPassword = new EditText(c);

        edPassword.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD);
                builder.setView(edPassword);

                // Добавляем кнопку "OK" - по нажатию слушателю вернется
содержимое edPassword
                builder.setPositiveButton("OK", new OnClickListener() {

                        public void onClick(DialogInterface dialog, int which)
{
                                notifyListenerValueSet(listener,
edPassword.getText()
                                                .toString());
                        }
                });

                // Добавляем кнопку "Cancel" - по нажатию слушателю вернется
null
                builder.setPositiveButton("Cancel", new OnClickListener() {

                        public void onClick(DialogInterface dialog, int which)
{
                                notifyListenerValueSet(listener, null);
                        }
                });

                return builder.create();
        }

        // Вызывает у слушателя метод onValueSet() с полученным в диалоге
значением value
        private static <T> void notifyListenerValueSet(
                        ValueSetListener<T> listener, T value) {
                if (listener != null)
                        listener.onValueSet(value);
        }

}
```

Вызов и обработка результата из класса MainActivity:

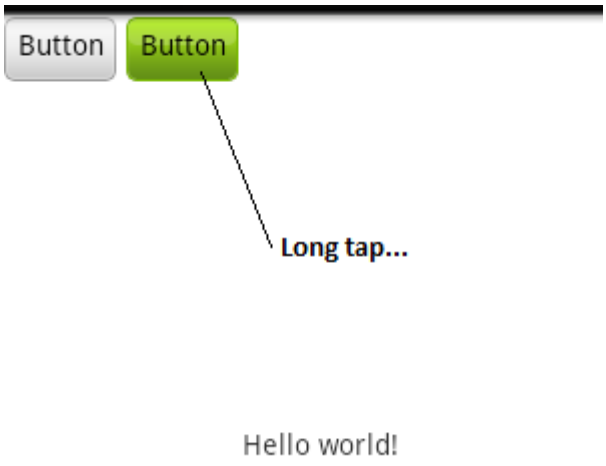
```
public void showInputPasswordDialog() {
        Dialogs.createInputPasswordDialog(this, "Пароль", "Введите пароль",
new ValueSetListener<String>() {

                public void onValueSet(String password) {
                        if(password==null)
                                Toast.makeText(MainActivity.this, "Вы нажали
Cancel", Toast.LENGTH_SHORT).show();
                        else if(isPasswordCorrect(password))
                                // ... действия, если пароль верный
                        else
                                // ... действия, если пароль неверный
                }

        }).show();
}
```

11. Работа с меню.

Практически все приложения на Android используют выпадающие меню. Выглядят они следующим образом:

Такие меню обычно появляются в нижней части экрана по нажатию кнопки Menu на устройстве.

В Android предусмотрен и другой вид меню – контекстное меню. Оно появляется в результате долговременного нажатия на каком-то компоненте управления, к которому оно привязано.



Для работы с меню используется класс Menu. Можно создать меню методами этого класса, можно создать меню из XML-описания.

Давайте создадим файл mainmenu.xml в папке res/menu проекта. Содержимое файла имеет следующую структуру:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

Корневым элементом структуры является тег <menu>, он содержит несколько тегов <item>, описывающих элементы меню. Каждый элемент имеет следующие свойства:

- android:id – идентификатор пункта меню для доступа к нему из кода
- android:icon – ссылка на иконку
- android:title – название пункта меню

Таким образом, мы познакомились с еще одним типом ресурсов – menu. В файле R.java для каждого ресурса этого типа с именем menu_name создается константа R.menu.menu_name.

12. Создание главного меню окна.

Главное меню вызывается при нажатии кнопки Menu на устройстве. При этом отрабатывает специальный метод класса Activity – onCreateOptionsMenu(). Вот стандартный его обработчик:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
}
```

Чтобы извлечь объект меню из его XML-описания, используется объект класса MenuInflater. Мы получаем его из метода getMenuInflater() окна и применяем метод inflate(int menuId, Menu menu), который формирует объект Menu из ресурса “@menu/menuId” и встраивает его в уже подготовленный объект menu. Можно доформировать его дополнительными пунктами следующим образом:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
        menu.add("Доп.пункт1");
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
}
```

Метод должен возвращать true, если меню в итоге будет выведено. Если метод вернет false, меню не будет выведено на экран.

Итак, меню мы вывели. Как отловить выбор пункта меню? Автоматически вызывается метод Activity – onOptionsItemSelected(MenuItem item), где item – выбранный пункт меню.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_1:
                    // Выбран пункт меню с id=@id/menu_1
                    return true;
                case R.id.menu_2:
                        // Выбран пункт меню с id=@id/menu_2
                    return true;
                default:
                    return super.onOptionsItemSelected(item);
            }
        }
```

13. Создание контекстного меню.

Контекстное меню появляется при долговременном нажатии на элемент управления, к которому он привязан. Чтобы привязать меню к элементу View v, необходимо вызвать метод `registerForContextMenu(v)`. Тогда перед вызовом меню отработает метод класса Activty - `onCreateContextMenu()`. Давайте посмотрим на его реализацию:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        int menuId;

        if(v==button1)
                menuId = R.menu.menu_for_button1;
        if(v==button2)
                menuId = R.menu.menu_for_button2;
        if(v==button3)
                menuId = R.menu.menu_for_button3;

        MenuInflater inflater = getMenuInflater();
        inflater.inflatemenuId, menu);
}
```

Аргументы метода:

- menu – созданная заготовка меню, которую можно доформировать.
- v – View, к которому привязано меню
- menuInfo – дополнительная информация о меню.

В зависимости от v мы можем определить структуру меню, которая будет выводится на экран. Для этого определим константу, соответствующую ресурсу типа menu с нужным идентификатором.

14. Домашнее задание

Давайте познакомимся с тем как добавлять View в Layout из кода. Давайте добавим кнопку на LinearLayout:

```
Button b = new Button(this);
b.setText("Button!");
LinearLayout l = new LinearLayout(this);
l.addView(b); // кнопка добавлена на LinearLayout!
```

Давайте так же познакомимся с тем как добавить View обработчик события «Нажатие» из кода:

```
b.setOnClickListener(new OnClickListener() {

        public void onClick(View v) {
                // Действия при нажатии. V – ссылка на View, на которое нажали
        }
});
```

Создайте приложение, выводящее на экран 10 кнопок с надписями «Кнопка 1», «Кнопка 2»….

Долговременное нажатие (long-tap как говорят на сленге) по каждой кнопке приведет к вызову контекстного меню с пунктами:

- Редактирование – вызовет диалог, в котором запрашивается название кнопки и меняется по кнопке OK. Диалог содержит кнопки OK, Cancel.
- Удаление – кнопка удаляется (Метод removeView(View v) класса LinearLayout).

Нажатие по кнопке Menu вызовет главное меню окна с пунктами:

- Выбрать цвет экрана – появляется диалог с выбором цвета: красный, белый, черный, зеленый. Выбор одного из пунктов цвет LinearLayout меняется на выбранный (метод setBackgroundColor() класса View, например:

```
l.setBackgroundColor(getResources().getColor(R.color.green));
```

- Выход – появляется диалог «Вы действительно хотите выйти из приложения?» с кнопками «Да», «Нет», «Справка». Нажатие кнопки «Да» приводит к уничтожению окна экрана, а следовательно, выходу из приложения (метод finish() класса Activity). Нажатие по кнопке «Справка» выводит AlertDialog с разъяснениями о том, что данные не будут сохранены при выходе из приложения.
- О программе – выводится AlertDialog с любым нестандартным заголовком и информацией об этом приложении.

И ВОПРОС КО ВСЕМ:

Предположим, я хочу выводить одинаковое главное меню на всех окнах моего приложения (это чаще всего мне и будет нужно). Неужели мне придется во всех Activities перегружать методы onCreateOptionsMenu() и onOptionsItemSelected() ???