Лабораторная работа № 3 по теме:
«Работа с уведомлениями и диалогами»

Теоретическая часть

При работе пользователей с приложением могут возникнуть различные ситуации, о которых необходимо уведомить пользователя. Некоторые из них требуют, чтобы пользователь отреагировал на них, другие нет ― они выполняют чисто информативную функцию.

Уведомление ― это сообщение, которое Android отображает за пределами пользовательского интерфейса приложения. Они могут передавать напоминания, сообщения от других пользователей или другую своевременную информацию из конкретного приложения.

Существуют два типа уведомлений:

- Toast Notification — всплывающие уведомления, для кратких сообщений, не требующих реакции пользователя;
- Status Bar Notification — уведомления в строке состояния. Эти уведомления предназначены для постоянных напоминаний, отображающихся в виде значка в строке состояния и требующих реакции пользователя.

Всплывающие уведомления

Всплывающее уведомление является сообщением, которое появляется на поверхности окна приложения. Оно заполняет необходимое ему количество пространства, требуемого для сообщения, при этом текущее окно активити приложения остается видимым для пользователя.

Всплывающее уведомление обычно применяется для коротких текстовых сообщений. Они автоматически исчезают по истечении определённого времени.

Для создания всплывающего уведомления сначала необходимо создать экземпляр класса Toast с помощью метода makeText (). Он возвращает

правильно инициализированный объект Toast. Отобразить всплывающее уведомление можно с помощью show () (листинг 1).

Листинг 1. Создание и отображение всплывающего уведомления.

```
Context context = getApplicationContext();
CharSequence text = "New toast!";
int duration = Toast.LENGTH_LONG;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

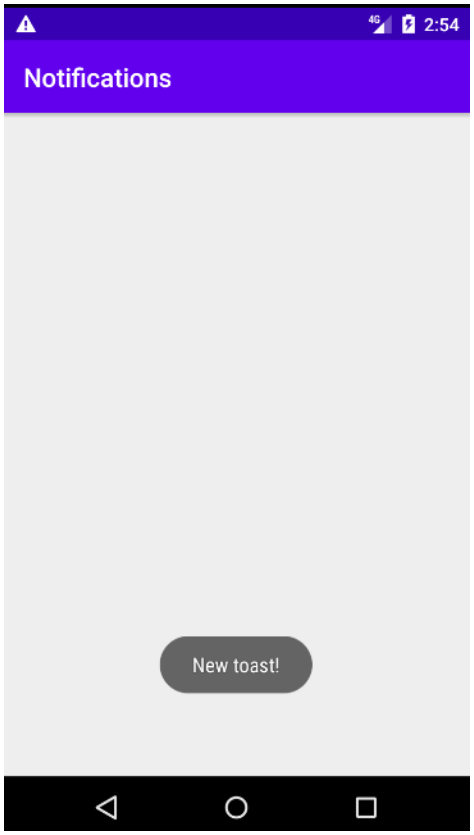

Рисунок 1. Всплывающее уведомление.

Метод makeText () принимает три параметра:

- контекст приложения;
- текстовое сообщение или ссылка на строковый ресурс с сообщением;
- продолжительность отображения уведомления.

Контекст (Context) — это базовый абстрактный класс, реализация которого обеспечивается системой Android. Это интерфейс к глобальной информации о среде приложения. Он позволяет получить доступ к ресурсам и классам, специфичным для приложения, а также выполнять операции на

уровне приложения, такие, как запуск активити, отправка широковещательных сообщений, получение намерений и прочее. Для получения контекста приложения можно воспользоваться методом getApplicationContext ().

Продолжительность отображения уведомления задаётся с помощью двух констант (значение по умолчанию — Toast.LENGTH_SHORT):

- Toast.LENGTH_SHORT — короткое уведомление (2 секунды);
- Toast.LENGTH_LONG — длительное уведомление (3.5 секунды).

Стандартное всплывающее уведомление появляется в нижней части экрана по центру по горизонтали. Изменить место появления уведомления можно с помощью метода setGravity (int, int, int). Этот метод принимает три параметра:

- стандартная константа для размещения объекта в пределах потенциально большего контейнера, определенная в классе Gravity (например, Gravity.CENTER, Gravity.TOP и др.);
- смещение по оси X;
- смещение по оси Y.

Если нужно сместить позицию вправо относительно положения, заданного стандартной константой, то следует увеличить значение второго параметра. Чтобы сдвинуть позицию вниз, следует увеличить значение последнего параметра.

Например, ниже приведён пример кода, смещающего уведомление в центр экрана:

```
toast.setGravity(Gravity.CENTER, 0, 0);
```

Если простого текстового сообщения недостаточно, а требуется, например, отобразить также картинку, можно создать собственный дизайн компоновки уведомления.

Создать компоновку уведомления можно в xml-ресурсе или динамически в коде приложения, создав корневой объект View и всех его потомков. В листинге 2 приведён пример компоновки уведомления.

Листинг 2. Компоновка уведомления.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/Toast_Layout"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/toast">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center_horizontal">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="200dp"
            android:layout_height="100dp"
            android:layout_weight="1"
            android:paddingLeft="20dp"
            android:src="@drawable/exclamation"
            android:layout_gravity="center_vertical"/>
        <TextView
            android:id="@+id/textView"
            android:text="Всплывающее уведомление!"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="28sp"
            android:layout_gravity="center_vertical"
            android:paddingLeft="20dp"
            android:typeface="serif"
            android:textColor="@android:color/holo_blue_dark"/>
    </LinearLayout>
</LinearLayout>
```

Обратите внимание, что для корневого элемента компоновки создаётся идентификатор. Он потребуется в дальнейшем в методе inflate ().

Для получения компоновки из xml-файла и работы с ней в программе используется класс LayoutInflater и его методы getLayoutInflater () (или getSystemService ()). Метод getLayoutInflater () возвращает объект LayoutInflater, а затем вызовом метода inflate (int, ViewGroup) получают корневой объект View этой компоновки.

В метод inflate (int, ViewGroup) передаются 2 параметра. Первый — это идентификатор ресурса компоновки уведомления, а второй — корневое представление компоновки (листинг 3).

Листинг 3. Получение корневого представления.

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.toast_layout,
        (ViewGroup) findViewById(R.id.Toast_Layout));
```

После получения корневого представления из него можно получить все дочерние представления уже известным методом findViewById () и определить информационное наполнение для этих элементов (если это не было сделано в компоновке) (листинг 4).

Листинг 4. Получение дочернего представления.

```
TextView text = (TextView) layout.findViewById(R.id.textView);
text.setText("Всплывающее уведомление!");
```

Затем создается объект Toast и устанавливаются нужные свойства, такие, например, как положение (метод setGravity ()) и продолжительность времени показа уведомления (метод setDuration ()).

После этого вызывается метод setView (), которому передается компоновка уведомления, и метод show (), чтобы отобразить уведомление с собственной компоновкой (листинг 5).

Листинг 5. Отображение всплывающего уведомления.

```
Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

Если не создавать xml-файл компоновки, а использовать динамическое создание представлений, то корневое представление также передаётся в метод setView (View).

На рисунке 2 показано всплывающее уведомление с индивидуальным дизайном.

Рисунок 2. Всплывающее уведомление с индивидуальным дизайном.

Уведомления в строке состояния

Уведомления, отображающиеся в строке состояния (англ. Status Bar Notification), изменялись с появлением новых версией Android. Уведомления отображаются в разных местах и в разных форматах, например, в виде значка в строке состояния, более подробной записи в панели уведомлений, в виде наклейки на значке приложения и на спаренных носимых устройствах.

Когда уведомление отправляется, оно сначала появляется в виде значка в строке состояния (рис. 3).

Рисунок 3. Значки уведомлений в левой части строки состояния.

Пользователи могут провести пальцем вниз по строке состояния, чтобы открыть панель уведомлений, где они смогут просмотреть дополнительные сведения и выполнить действия с уведомлением (рис. 4).

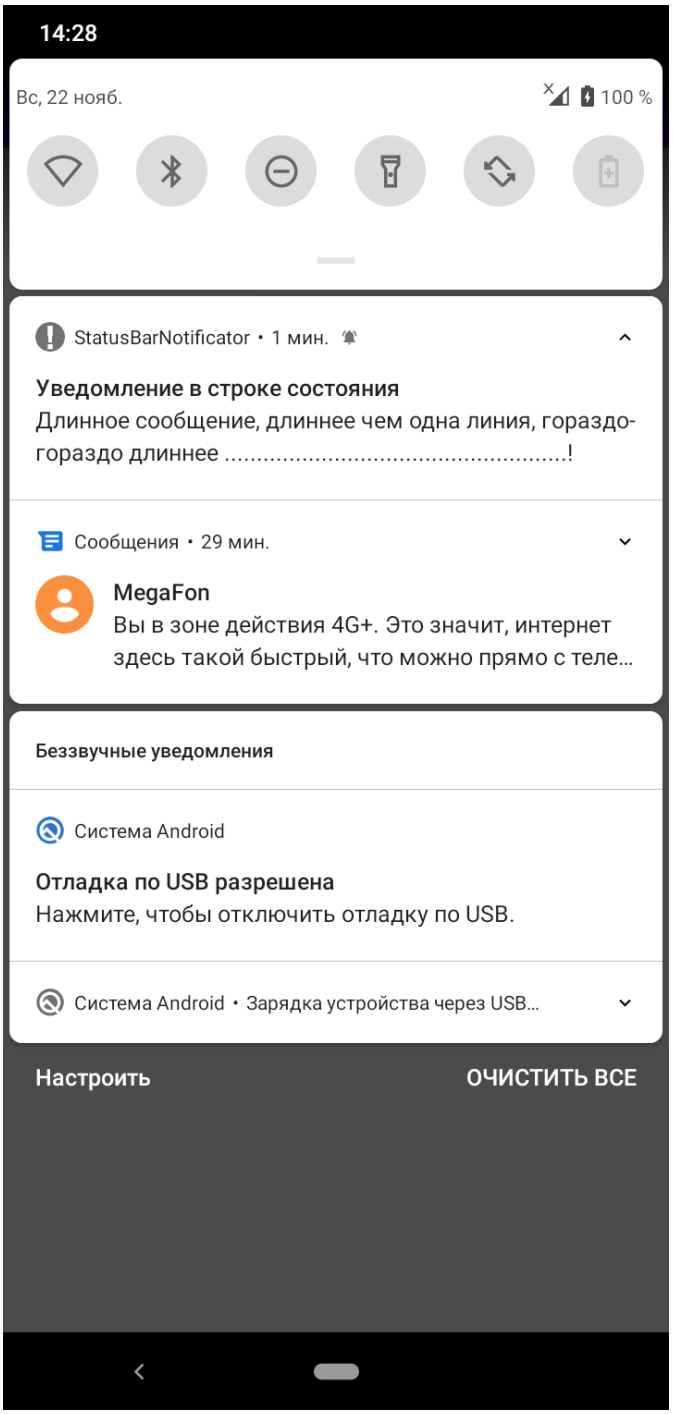

Рисунок 4. Уведомления в панели уведомлений.

Пользователи могут открыть расширенное представление уведомления, в котором отображается дополнительный контент и кнопки действий, если они предусмотрены.

Уведомление остается видимым в панели уведомлений до тех пор, пока оно не будет отклонено приложением или пользователем.

Начиная с Android 5.0, уведомления могут на короткое время появляться во всплывающем окне, называемом *heads-up*. Такое поведение обычно характерно для важных уведомлений, которые пользователь должен получить немедленно. Такие уведомления появляются только в том случае, если устройство разблокировано. Всплывающее уведомление типа *heads-up*, появляется на определённое время (так же, как и Toast), но после исчезновения с экрана остается видимым в панели уведомлений.

Начиная с Android 5.0 уведомления также могут появляться на экране блокировки.

Дизайн уведомления определяется системными шаблонами. В приложении определяется содержимое для каждой части шаблона. Некоторые детали уведомления отображаются только в развернутом виде. На рисунке 5 обозначены разные части шаблона уведомления.

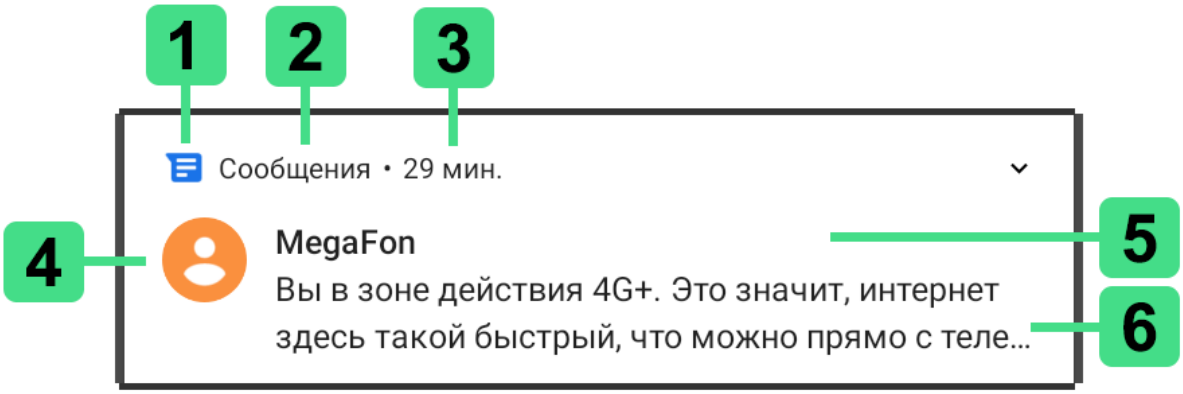

Рисунок 5. Внешний вид уведомления.

Наиболее распространенные части шаблона уведомления:

1. Маленький значок: обязательная часть, устанавливается с помощью метода setSmallIcon ().
2. Имя приложения: предоставляется системой.
3. Отметка времени: предоставляется системой, но её можно переопределить с помощью метода setWhen () или скрыть с помощью метода setShowWhen (false).

4. Большой значок: необязателен (обычно используется только для фотографий контактов), устанавливается с помощью метода setLargeIcon ().
5. Заголовок: не является обязательным, устанавливается с помощью метода setContentTitle ().
6. Текст: не является обязательным, устанавливается с помощью метода setContentText ().

Рассмотрим, как создать уведомление в строке состояния с различными функциями для Android 4.0 (уровень API 14) и выше.

Для начала необходимо настроить содержимое и канал уведомления с помощью объекта NotificationCompat.Builder. NotificationCompat ― это класс, который выступает помощником для доступа к функциям уведомлений.

В листинге 6 показано, как создать уведомление со следующим содержимым:

- Маленький значок, установленный методом setSmallIcon (). Это единственное обязательное содержимое уведомления.
- Заголовок уведомления, установленный методом setContentTitle ().
- Основной текст уведомления, установленный методом setContentText ().
- Приоритет уведомления, установленный методом setPriority (). Приоритет определяется для Android 7.1 и ниже. Для Android 8.0 и выше вместо него устанавливается важность канала. Возможные уровни приоритета уведомлений:
  - Крайняя важность: издает звук и отображается как всплывающее heads-up уведомление.
  - Высокий приоритет: издает звук.
  - Средний приоритет: без звука.

- Низкий приоритет: без звука, не отображается в строке состояния.

Листинг 6. Создание уведомления.

```
int notificationId = 1;
String CHANNEL_ID = "my_channel_01";
...
NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
CHANNEL_ID)
        .setSmallIcon(R.drawable.status_bar_icon)
        .setContentTitle("Уведомление в строке состояния")
        .setContentText("Текст сообщения")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Обратите внимание, что конструктор NotificationCompat.Builder требует, чтобы вы указали идентификатор канала (CHANNEL_ID). Это требуется для совместимости с Android 8.0 (уровень API 26) и выше, но игнорируется более старыми версиями.

По умолчанию текст уведомления обрезается до одной строки. Если вы хотите, чтобы ваше уведомление было длиннее, вы можете включить расширяемое уведомление, добавив шаблон стиля с помощью setStyle (). Например, код в листинге 7 создаёт большую текстовую область.

Листинг 7. Создание уведомления с длинным сообщением.

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
CHANNEL_ID)
        .setSmallIcon(R.drawable.status_bar_icon)
        .setContentTitle("Уведомление в строке состояния")
        .setContentText("Длинное сообщение, длиннее чем одна линия")
        .setStyle(new NotificationCompat.BigTextStyle()
                .bigText("Длинное сообщение, длиннее чем одна линия"))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Для отправки уведомления (листинг 8) необходимо вызывать метод NotificationManagerCompat.notify (), передав ему уникальный идентификатор уведомления и результат NotificationCompat.Builder.build (). Идентификатор уведомления, который вы передаете в NotificationManagerCompat.notify (), нужно сохранить, создав

соответствующую переменную, потому что он понадобится позже, если потребуется обновить или удалить уведомление.

Листинг 8. Отправка уведомления.

```
NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(getApplicationContext());
notificationManager.notify(notificationId, builder.build());
```

Начиная с Android 8.1 (уровень API 27), приложения не могут издавать звуковое уведомление чаще одного раза в секунду. Если приложение отправляет несколько уведомлений за одну секунду, все они отображаются должным образом, но звучит только одно в секунду.

Для доставки уведомления на Android 8.0 и выше, нужно зарегистрировать канал уведомлений вашего приложения в системе (листинг 9), передав экземпляр NotificationChannel в createNotificationChannel ().

Листинг 9. Создание канала уведомлений.

```
private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID,
name, importance);
        channel.setDescription(description);
        NotificationManager notificationManager =
getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Распределяя уведомления по каналам, пользователи могут отключать определенные каналы уведомлений для вашего приложения (вместо отключения всех ваших уведомлений), а также пользователи могут управлять визуальными и звуковыми параметрами для каждого канала в системных настройках Android.

Одно приложение может иметь несколько каналов уведомлений — отдельный канал для каждого типа уведомлений, которые выдает

приложение. Приложение также может создавать каналы уведомлений в ответ на выбор, сделанный пользователями приложения. Например, можно настроить отдельные каналы уведомлений для каждой группы контактов, созданной пользователем в приложении для обмена сообщениями.

В каждом канале также указывается уровень важности уведомлений. Таким образом, все уведомления, отправленные в один и тот же канал уведомлений, ведут себя одинаково.

Канал уведомлений создаётся перед отправкой каких-либо уведомлений на Android 8.0 и выше, поэтому нужно выполнить этот код сразу после запуска приложения.

Обратите внимание, что в конструкторе NotificationChannel требует указать важности канала, используя одну из констант из класса NotificationManager. Для поддержки Android 7.1 и ниже требуется также установить приоритет с помощью setPriority () для объекта NotificationCompat.Builder.

В некоторых случаях система Android может сама изменить уровень важности, также сам пользователь всегда может переопределить уровень важности для данного канала.

В таблице 1 приведены уровни важности (для Android 8.0 и выше) и уровни приоритета уведомлений (для Android 7.1 и ниже).

Таблица 1. Уровни важности и уровни приоритета уведомлений.

Уровень, видимый пользователю	**Важность (Importance)**	**Приоритет (Priority)**
Urgent (Крайняя важность): звук и всплывающее heads-up уведомление	IMPORTANCE_HIGH	PRIORITY_HIGH или PRIORITY_MAX
High (Высокий приоритет): звук	IMPORTANCE_DEFAULT	PRIORITY_DEFAULT
Medium (Средний приоритет): без звука	IMPORTANCE_LOW	PRIORITY_LOW
Low (Низкий приоритет): без уведомлений	IMPORTANCE_MIN	PRIORITY_MIN

Каждое уведомление должно реагировать на событие касания, обычно для открытия того активити приложения, которое соответствует уведомлению. Для этого следует указать намерение, определенное с помощью объекта PendingIntent, и передать его в метод setContentIntent ().

В листинге 10 показано, как создать базовое намерение для открытия активити MainActivity при нажатии на уведомление.

Листинг 10. Установка открытия активити при нажатии на уведомление.

```
Intent intent = new Intent(this, MainActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
PendingIntent pendingIntent = PendingIntent.getActivity(this,0,intent,0);

NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
CHANNEL_ID)
    .setSmallIcon(R.drawable.status_bar_icon)
    .setContentTitle("Новое уведомление")
    .setContentText("Нажмите на уведомление для перехода в приложение")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .setContentIntent(pendingIntent)
    .setAutoCancel(true);
```

Метод setAutoCancel () автоматически удаляет уведомление, когда пользователь нажимает на него.

Диалоговые окна

Диалог — это обычно маленькое окно, которое появляется перед текущим активити. Диалоги обычно используются для уведомлений и коротких действий, которые непосредственно касаются приложения. Обычно в диалоге пользователю предлагается принять решение или ввести дополнительную информацию.

Класс Dialog является базовым классом для диалогов, но следует избегать создания экземпляров Dialog напрямую. Вместо этого можно использовать один из следующих подклассов:

- AlertDialog — это диалоговое окно, которое может отображать заголовок, до трех кнопок, список выбираемых элементов или настраиваемый пользовательский макет.
- DatePickerDialog и TimePickerDialog — это диалоги с предопределенным пользовательским интерфейсом, который позволяет пользователю выбрать дату и время соответственно.

Также платформа Android включает еще один класс диалогового окна — ProgressDialog (диалог с индикатором выполнения задачи). Однако этот виджет считается устаревшим, поскольку не позволяет пользователям взаимодействовать с приложением до завершения задачи.

Названные классы определяют стиль и структуру диалога, но в качестве контейнера для диалога следует использовать DialogFragment. Класс DialogFragment был добавлен в Android 3.0 (уровень API 11) и предоставляет все элементы управления, необходимые для создания диалогового окна и управления его внешним видом, вместо вызова методов объекта Dialog.

Использование DialogFragment для управления диалоговым окном гарантирует, что он будет правильно обрабатывать события жизненного цикла, например, когда пользователь нажимает кнопку «Назад» или поворачивает экран. Класс DialogFragment также позволяет повторно использовать пользовательский интерфейс диалогового окна в качестве встраиваемого компонента в более крупный пользовательский интерфейс (например, когда вы хотите, чтобы пользовательский интерфейс диалогового окна отображался по-разному на больших и малых экранах).

AlertDialog

Диалоговое окно AlertDialog состоит из трех частей:

- Заглавие — является необязательным и должно использоваться только тогда, когда область содержимого занята подробным сообщением, списком или настраиваемым макетом (компоновкой). Если нужно

сформулировать простое сообщение или вопрос, заголовок не используется.

- Область содержимого — может отображать сообщение, список или другой настраиваемый макет (компоновку с произвольным дизайном).
- Кнопки действий — в диалоговом окне должно быть не более трех кнопок действий (положительный, отрицательный и нейтральный ответ).

Класс AlertDialog.Builder предоставляет API-интерфейсы, которые позволяют создавать AlertDialog с этими типами содержимого.

Создадим диалоговое окно путем расширения класса DialogFragment и создания AlertDialog в методе обратного вызова onCreateDialog (). В листинге 11 приведён класс, наследуемый от DialogFragment. В нём создаётся AlertDialog с двумя кнопками («Да» и «Нет»). При нажатии на кнопку положительного ответа, активити, из которого был вызван диалог, завершается, а при нажатии на кнопку отрицательного ответа закрывается диалог.

Листинг 11. Класс MyDialogFragment.

```
package com.example.dialogapp;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import androidx.fragment.app.DialogFragment;
public class MyDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());
        builder.setMessage(R.string.dialog_message);
        builder.setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        getActivity().finish();   }        });
        builder.setNegativeButton(R.string.no, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();          }        });
        return builder.create();
    }
}
```

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

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

Метод getActivity () возвращает активити, связанную с фрагментом. Класс Activity является потомком класса Context, поэтому в конструктор класса AlertDialog.Builder можно передавать активити.

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

builder.setMessage(R.string.dialog_message);

С помощью метода setTitle () можно задать заглавие сообщения.

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

Для каждой кнопки используется один из методов: setPositiveButton (), setNegativeButton () или setNeutralButton (), которые принимают в качестве параметров надпись для кнопки, и интерфейс DialogInterface.OnClickListener, который определяет действие, когда пользователь нажимает кнопку.

Чтобы пользователь не мог закрыть диалог кнопкой Back (Назад), вызывается метод setCancelable ():

builder.setCancelable(false);

После определения класса, наследуемого от DialogFragment, в классе активити нужно создать экземпляр этого класса и вызывать метод show () для этого объекта, когда потребуется отобразить активити (листинг 12).

Листинг 12. Создание и вызов диалога.

```
FragmentManager manager = getSupportFragmentManager();
MyDialogFragment myDialogFragment = new MyDialogFragment();
myDialogFragment.show(manager, "myDialog");
```

На рисунке 6 приведено получившееся диалоговое окно.

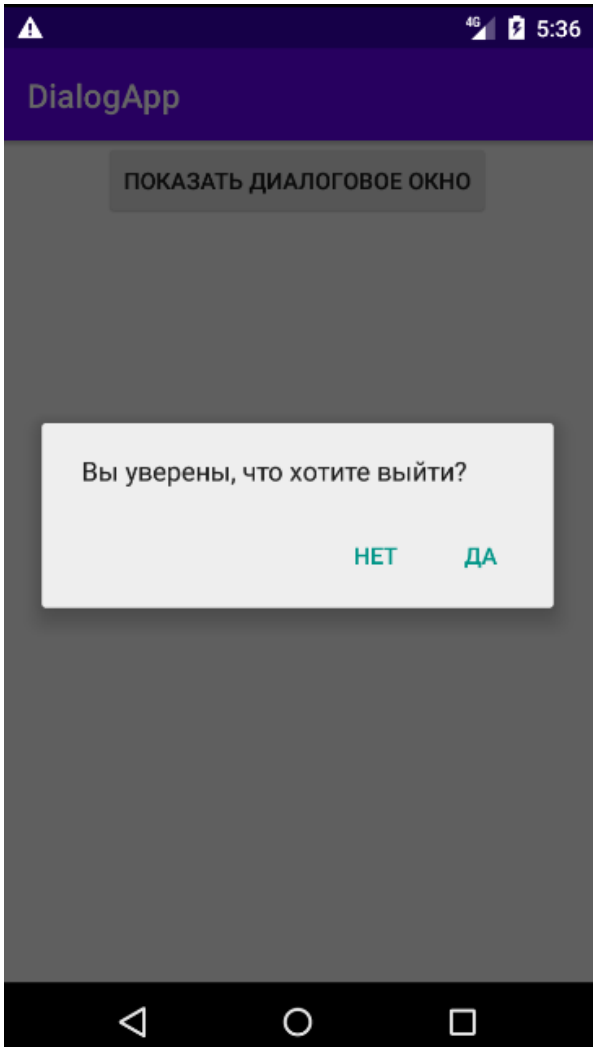

Рисунок 6. Диалоговое окно AlertDialog с двумя кнопками.

Также как при работе с Toast Notification, можно использовать компоновки для создания произвольного дизайна диалогового окна с использованием дополнительных элементов, например, изображений ImageView.

После создания компоновки необходимо передать корневой объект компоновки в код создания диалога. Чтобы получить корневой объект компоновки, используется класс LayoutInflater. Этот класс нужен для

преобразования xml-компоновки в соответствующие объекты View в коде программы. Далее создается объект AlertDialog.Builder и методом setView () для него устанавливается полученная ранее компоновка (листинг 13).

Листинг 13. Создание дизайна диалога из компоновки.

```
LayoutInflater inflater = requireActivity().getLayoutInflater();
builder.setView(inflater.inflate(R.layout.dialog_layout, null));
```

Задание

Разработайте приложение, в котором будут кнопки отправки всплывающего уведомления (Toast Notification) с индивидуальным дизайном, уведомления в строке состояния, а также кнопка для отображения диалогового окна.

Контрольные вопросы:

1. Какие типы уведомлений существуют?
2. Что такое Toast Notification?
3. Что такое уведомление в строке состояния?
4. Каково назначение метода getApplicationContext?
5. Каково назначение метода Toast.makeText?
6. С помощью каких констант задаётся продолжительность отображения Toast уведомления?
7. Каково назначение метода setGravity?
8. Каково назначение метода getLayoutInflater?
9. Каково назначение метода setDuration?
10. Из каких шести частей состоит шаблон уведомления на панели уведомлений?
11. Что такое канал уведомлений?
12. Для чего нужен класс NotificationCompat?
13. Какие уровни важности и уровни приоритета уведомлений существуют?
14. Что такое диалог?

15. Какие классы диалогов существуют?
16. Что может содержать AlertDialog?
17. Сколько кнопок может быть в AlertDialog?
18. Каково назначение метода setCancelable?
19. Каково назначение метода setView?