Activity. Intent. Action. Intent filter

Многоэкранные приложения

До сих пор мы создавали приложения, которые содержали только **один экран** (Activity). Но вы замечали, что экранов в приложении обычно больше. Если рассмотреть, например, почтовое приложение, то в нем есть следующие экраны: список аккаунтов, список писем, просмотр письма, создание письма, настройки и т.д.

Второй экран

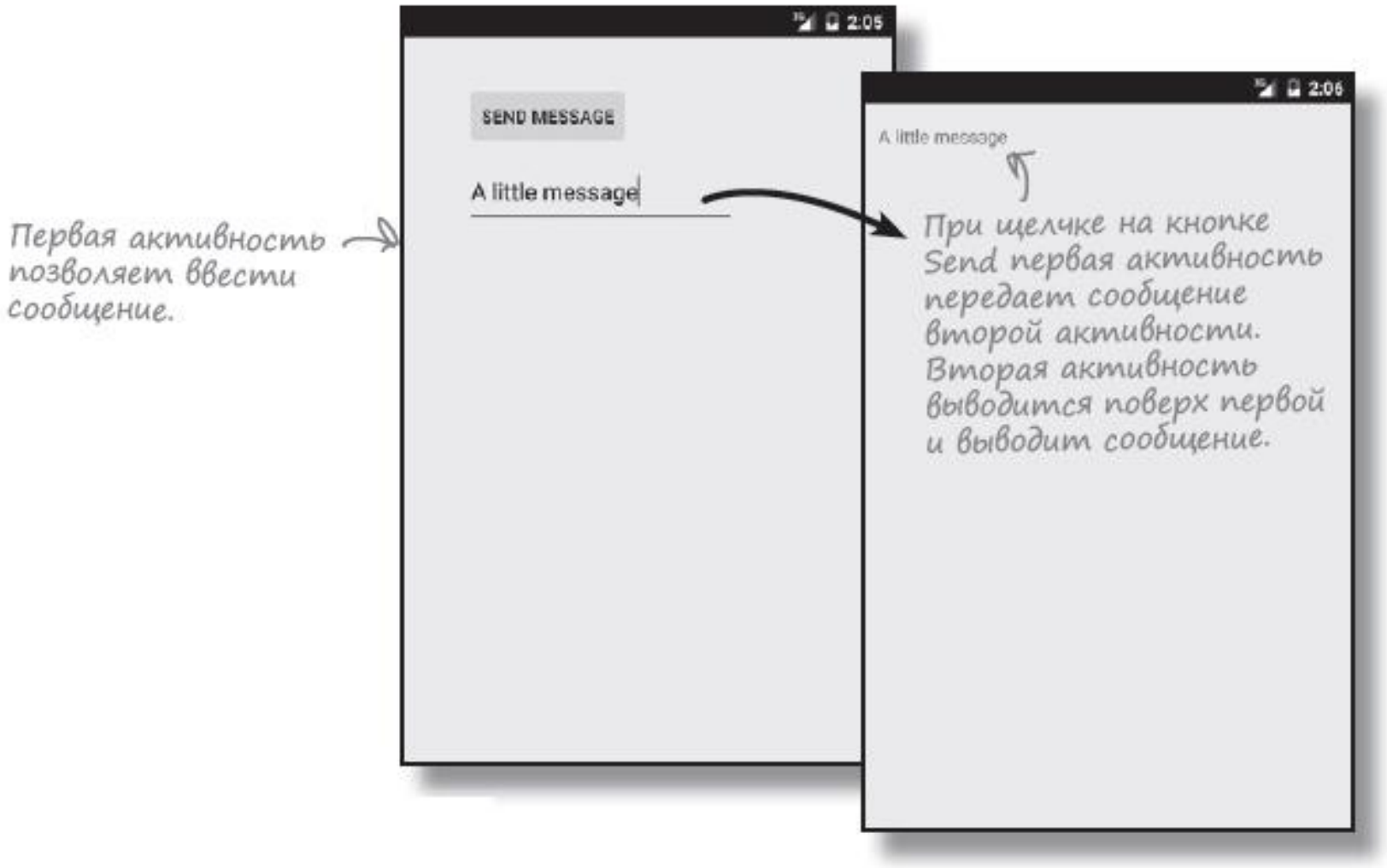
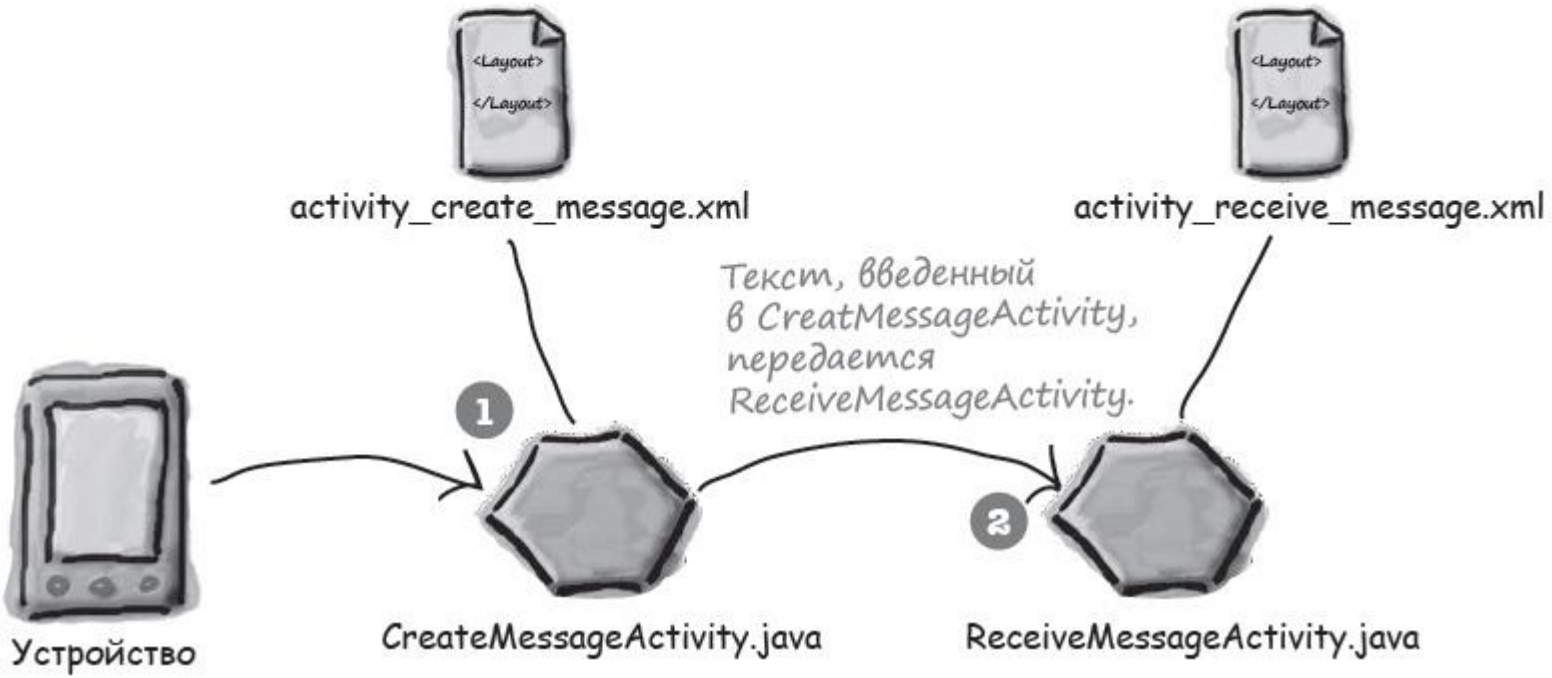


Схема работы

1. **В начале работы приложения запускается активность CreateMessageActivity.**
   Эта активность использует макет *activity_create_message.xml.*

2. **Пользователь щелкает на кнопке в CreateMessageActivity.**
   Кнопка запускает активность `ReceiveMessageActivity`, которая использует макет *activity_receive_message.xml.*



Последовательность действий

1. Создание базового приложения с одной Activtity и макетом.
2. Добавление второй Activtity и макета.
3. Организация вызова второй Activtity из первой.
4. Организация передачи данных из первой Activtity во вторую.
5. Получение данных во второй Activtity

Шаг 1. Создание базового приложения

На данном этапе создаем проект также, как мы делали это раньше. Таким образом, в нашем проекте будет первая Activity и ее разметка (xml-файл).

Шаг 2. 2-я Activity

В Android Studio имеется мастер для создания новых активностей и макетов в приложениях. По сути это упрощенная версия мастера, используемого при создании приложений.
Чтобы создать новую активность, выполните команду:

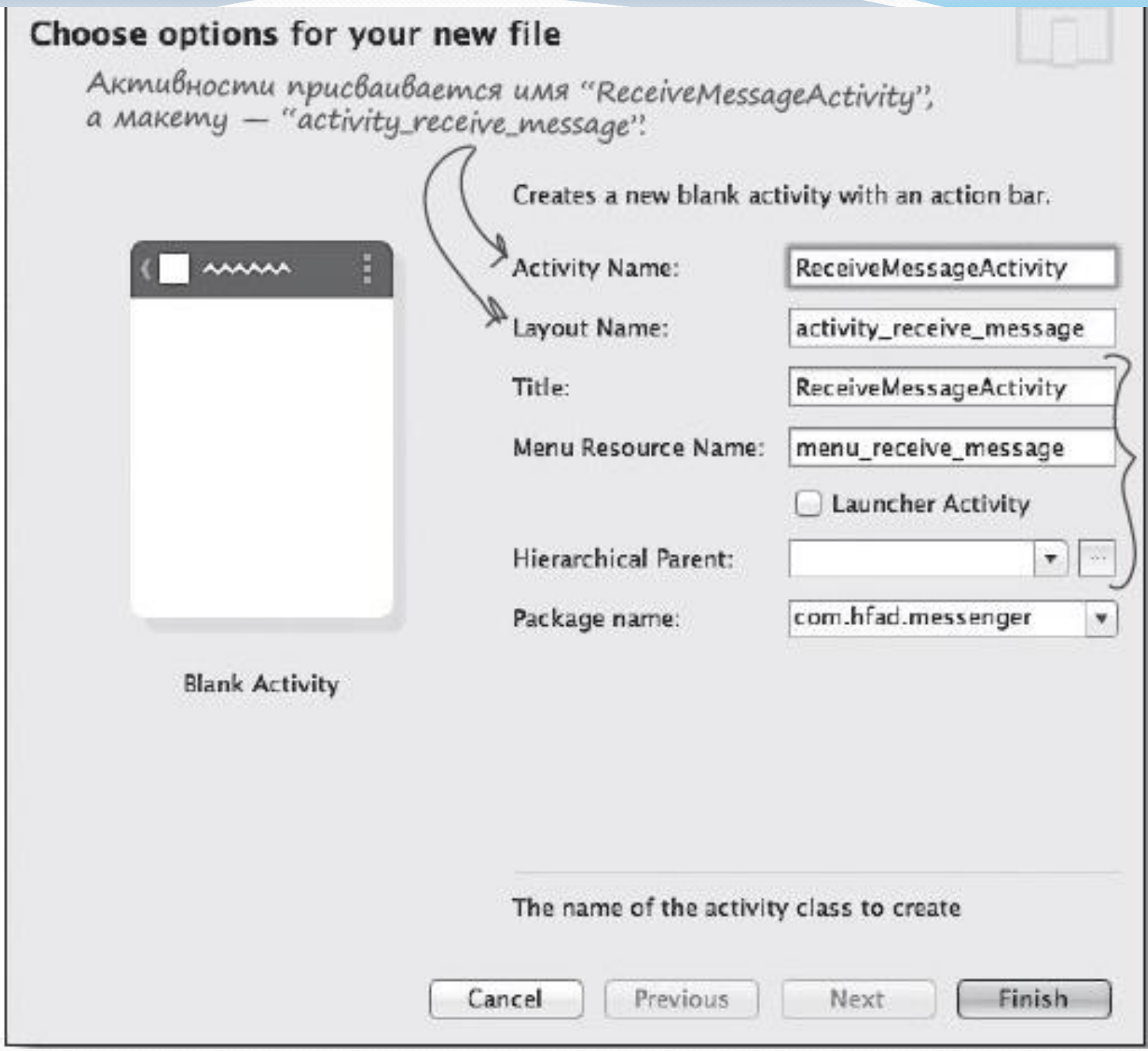
File → New → Activity

и выберите один из вариантов. На экране появляется окно, в котором вы сможете выбрать параметры новой активности.

Создаем новое Activity

Каждой создаваемой новой активности и макету необходимо присвоить имя.

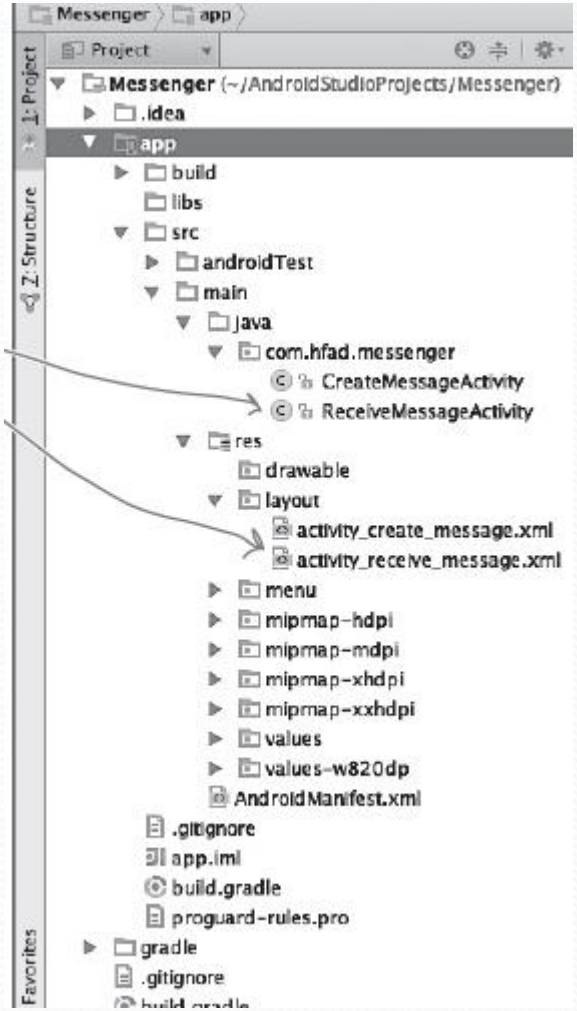
Необходимо также убедиться в правильности присвоенного имени пакету.



Оставьте остальным параметрам значения по умолчанию, так как нас сейчас интересует только создание новой активности и макета. Мы заменим большую часть кода, сгенерированного Android Studio.

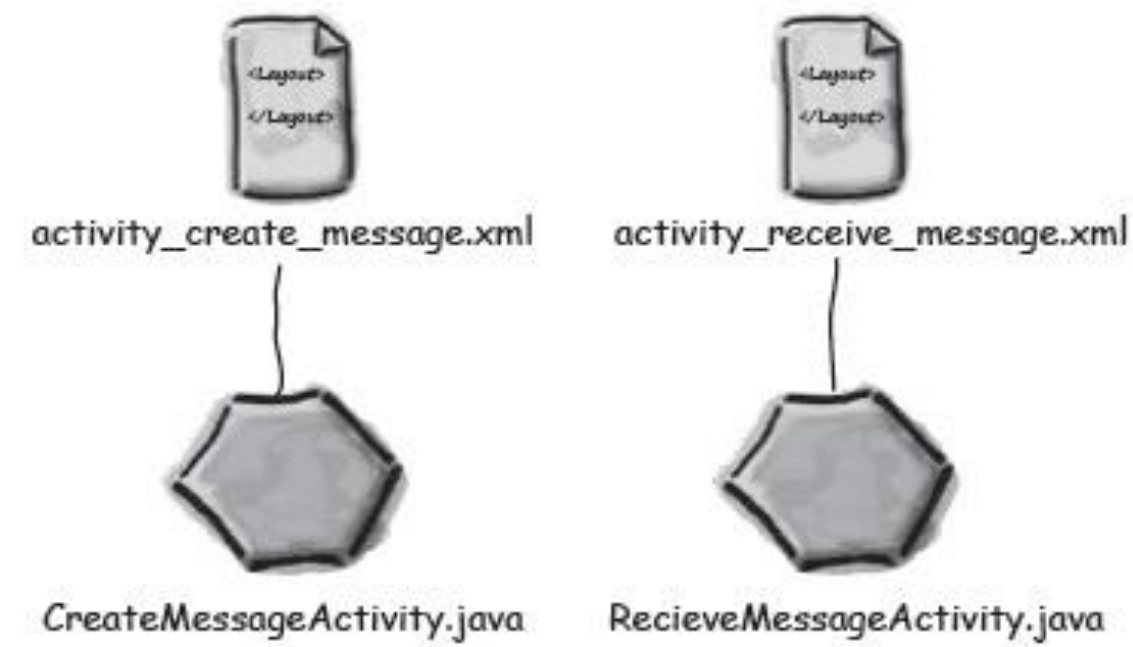
Структура проекта

Android Studio создал новый файл активности вместе с новым макетом. В папке app/src/main/java появился новый файл с именем ReceiveMessageActivity.java, а в папке app/src/main/res/layout — файл с именем activity_receive_message.xml.



Activities и их макеты

CreateMessageActivity использует макет *activity_create_message.xml, а ReceiveMessageActivity — макет activity_receive_message.xml.*



AndroidManifest

Android Studio также незаметно изменяет конфигурацию приложения в файле с именем *AndroidManifest.xml.*

Файл AndroidManifest.xml содержит важнейшую информацию о приложении:

какие активности в нем есть!

какие библиотеки ему необходимы и другие объявления.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hfad.messenger" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".CreateMessageActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".ReceiveMessageActivity"
            android:label="@string/title_activity_receive_message" >
        </activity>

    </application>

</manifest>
```

Первая активность, Create Message Activity.

Это означает, что активность является главной активностью приложения.

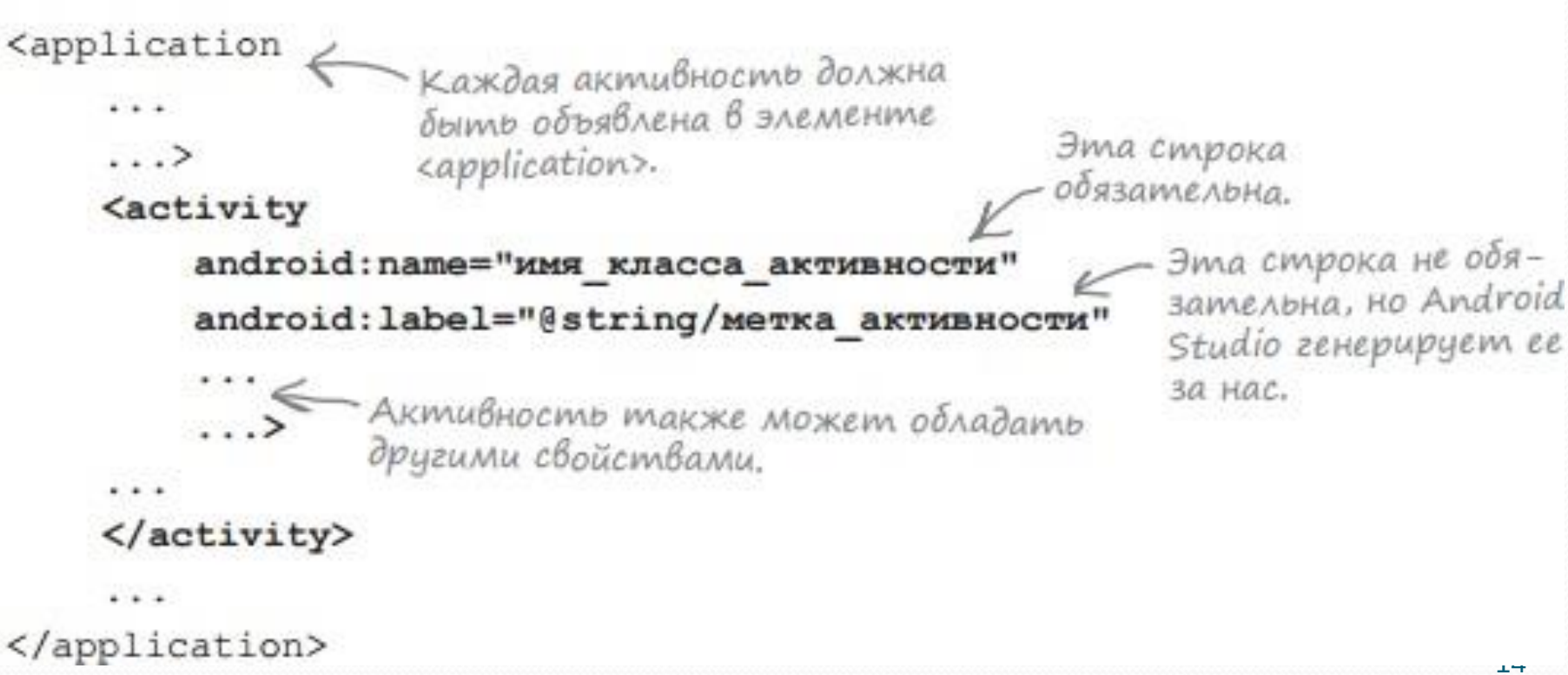
А это означает, что активность может использоваться для запуска приложения.

Вторая активность, Receive Message Activity.

Android Studio добавляет эти строки при добавлении второй активности.

Activity в AndroidManifest

Все Activity должны быть объявлены в файле AndroidManifest.xml. Если Activity не объявлена в файле, то система не будет знать о ее существовании.



Activity в AndroidManifest

Все Activity должны быть объявлены в файле AndroidManifest.xml. Если Activity не объявлена в файле, то система не будет знать о ее существовании.

```
<application
    ...
    ...>
    <activity
        android:name="имя_класса_активности"
        android:label="@string/метка_активности"
        ...
        ...>
    ...
    </activity>
    ...
</application>
```

Каждая активность должна быть объявлена в элементе <application>.

Эта строка обязательна.

Эта строка не обязательна, но Android Studio генерирует ее за нас.

Активность также может обладать другими свойствами.

Параметры Activity

- имя_класса_активности — имя класса с префиксом “.”
- android:label="@string/метка_активности«
  Метка выводится в верхней части экрана при выполнении активности.
- разрешения безопасности
- возможность использования активностями других приложений

Шаг 3. Вызов Activity

Чтобы запустить одну активность из другой, необходим Intent. Intent можно рассматривать как своего рода «намерение выполнить некую операцию». Это разновидность сообщений, позволяющая связать разнородные объекты (например, Activity) на стадии выполнения. Если одна Activity хочет запустить другую, она отправляет для этого Intent системе Android. Android запускает вторую Activity и передает ей Intent .

Создание Intent

Intent intent = new Intent(this, Target.class);

Первый параметр сообщает Android, от какого объекта поступил Intent; для обозначения текущей Activity используется ключевое слово this. Во втором параметре передается имя класса Activity, которая должна получить Intent

Отправка Intent

После того как Intent будет создан, он передается Android следующим вызовом:

startActivity(intent);

Этот вызов приказывает Android запустить Activity, определяемую интентом. При получении интента Android убеждается в том, что все правильно, и прика зывает Activity запуститься. Если найти Activity не удалось, инициируется исключение ActivityNotFoundException.

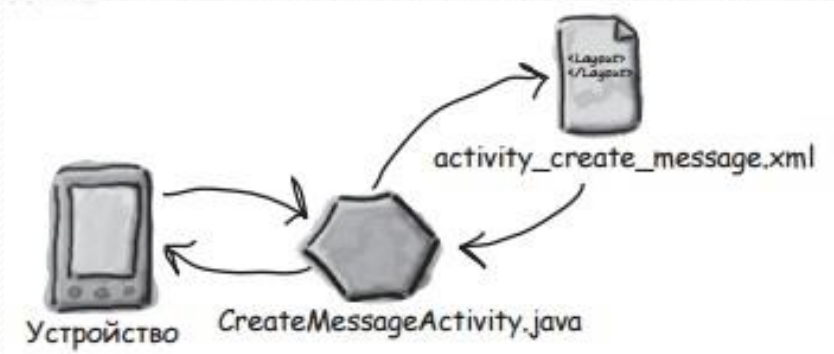
Схема запуска Activity

Запуск Activity

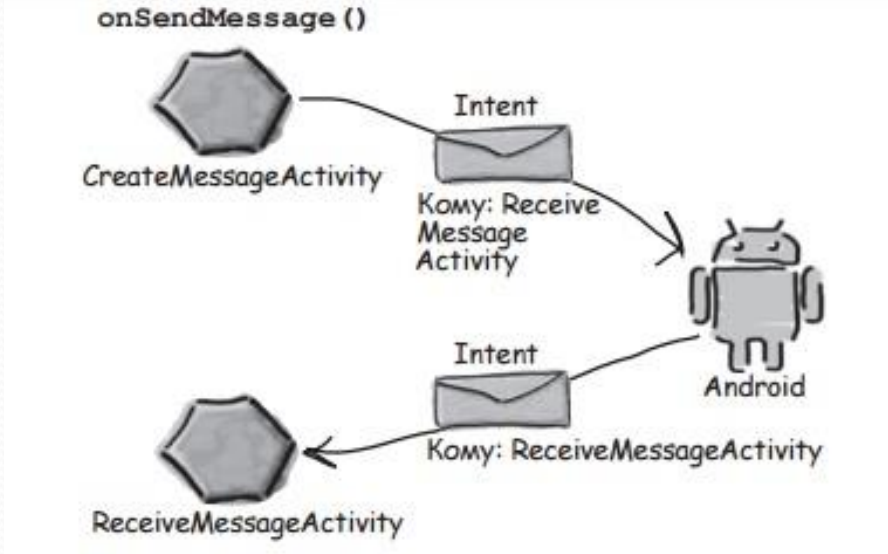
```
//Вызвать onSendMessage() при щелчке на кнопке
public void onSendMessage(View view) {
    Intent intent = new Intent(this, ReceiveMessageActivity.class);
    startActivity(intent);
}
```

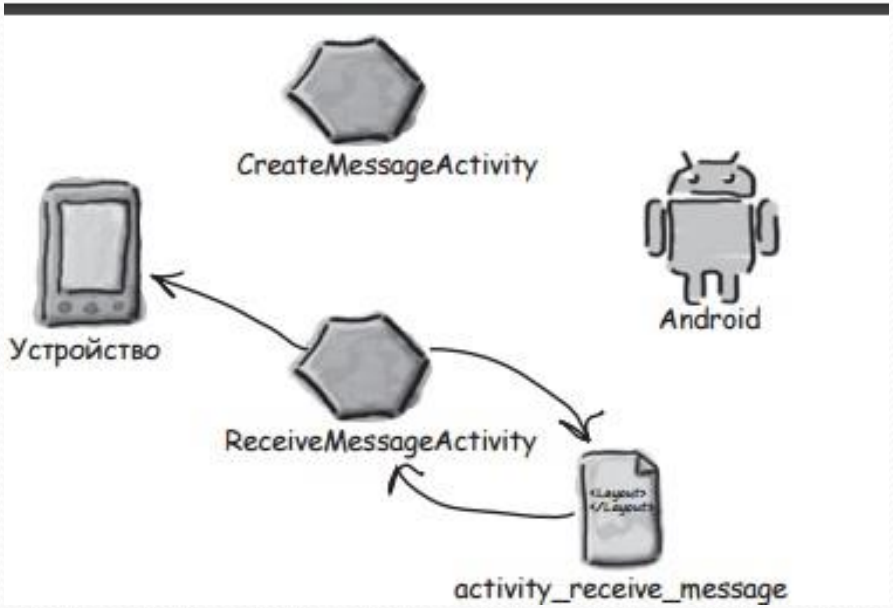
Запустить активность ReceiveMessageActivity.

Схема


activity_create_message.xml
Устройство
CreateMessageActivity.java


onSendMessage()
Устройство
CreateMessageActivity


onSendMessage()
CreateMessageActivity
Intent
Кому: Receive Message Activity
Android
Intent
Кому: ReceiveMessageActivity
ReceiveMessageActivity


CreateMessageActivity
Android
Устройство
ReceiveMessageActivity
activity_receive_message

Приложение готово!

Шаг 4. Отправка данных

В Intent можно добавить дополнительную информацию, которая должна передаваться получателю. В этом случае Activity, получившая Intent, сможет на него как-то среагировать. Для этого используется метод **putExtra**():

intent.putExtra("сообщение", значение);

где сообщение — имя ресурса для передаваемой информации, а значение — само значение. putExtra() позволяет передавать значение многих возможных типов. Например, это может быть примитив, массив примитивов или String. Многократные вызовы putExtra() позволяют включить в Intent несколько экземпляров дополнительных данных.

Отправка данных. Пример

```
//Вызвать onSendMessage() при щелчке на кнопке
public void onSendMessage(View view) {
    EditText messageView = (EditText) findViewById(R.id.message);
    String messageText = messageView.getText().toString();
    Intent intent = new Intent(this, ReceiveMessageActivity.class);
    intent.putExtra("message", messageText);
    startActivity(intent);
}
```

Получить текст из текстового поля с идентификатором message.

Добавить текст в интент под именем "message".

Шаг 5. Получение данных

getIntent() возвращает интент, запустивший Activity; из полученного интента можно прочитать любую информацию, отправленную вместе с ним. Конкретный способ чтения зависит от типа отправленной информации. Например, если вы знаете, что интент включает строковое значение с именем “message”, используйте следующий вызов:

```
Intent intent = getIntent();
String string = intent.getStringExtra("message");
```

Или intent.getIntExtra() для целых чисел:

```
int intNum = intent.getIntExtra("name", default_value);
```

Получение данных. Пример

```
public class ReceiveMessageActivity extends Activity {

    public static final String EXTRA_MESSAGE = "message";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_receive_message);
        Intent intent = getIntent();
        String messageText = intent.getStringExtra(EXTRA_MESSAGE);
        TextView messageView = (TextView)findViewById(R.id.message);
        messageView.setText(messageText);
    }
}
```

Имя дополнительного значения, передаваемого в интенте.

Получить интент и извлечь из него сообщение вызовом getStringExtra().

Добавить текст в надпись с идентификатором message.

Чужие Activity!

Теперь, когда наше приложение научилось отправлять сообщения другой Activity, его можно изменить так, чтобы оно отправляло сообщения другим людям. Для этого приложение интегрируется с другими приложениями, поддерживающими отправку сообщений и уже установленными на устройстве. В зависимости от того, какие приложения установлены у пользователя, можно организвать отправку сообщений из вашего приложения через Gmail, ВКонтакте, Twitter… и т.д.

Совершенно не обязательно ограничиваться активностями вашего приложения. С таким же успехом можно использовать активности других приложений.

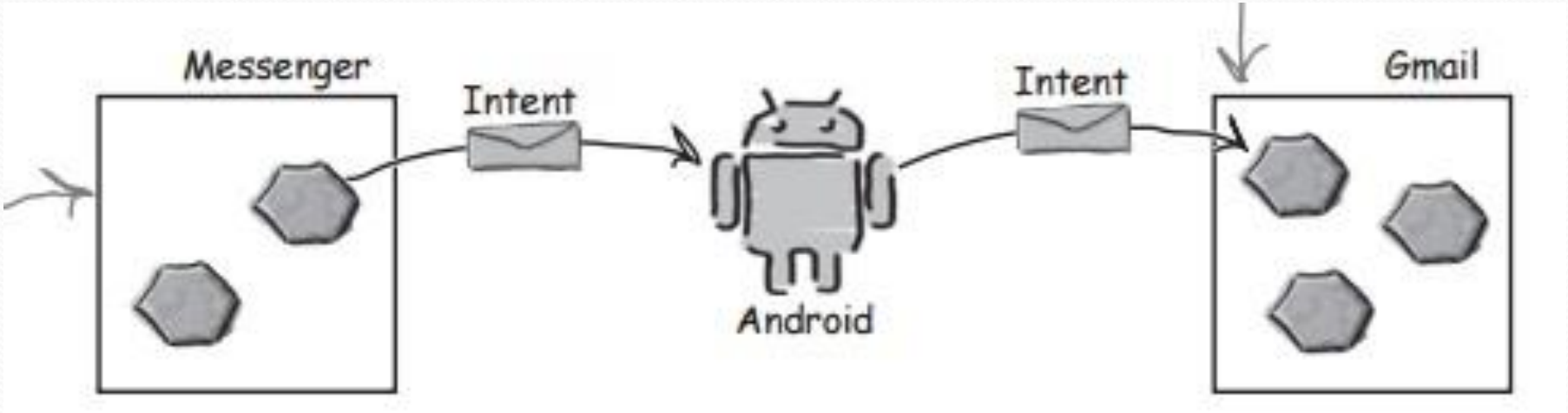
Чужие Activity!

Android-приложения состоят из одной или нескольких Activity. Каждая Activity представляет одну четко определенную операцию, которая может выполняться пользователем. Например, такие приложения, как Gmail, Google+, Сообщения, ВКонтакте Twitter, содержат активности, позволяющие отправлять сообщения, хотя в каждом приложении эта операция выполняется по-своему.

Чужие Activity!

Activity вашего приложения передает Intent андроиду, андроид проверяет его, а затем приказывает второй Activity запуститься — несмотря на то, что эта активность находится в другом приложении. Например, можно воспользоваться интентом для запуска активности Gmail, отправляющей сообщения, и передать ей текст, который нужно отправить. Вместо того, чтобы писать собственные активности для отправки электронной почты, можно воспользоваться готовым приложением Gmail.

Чужие Activity!



Не все так просто…

- Как узнать, какие активности доступны на устройстве пользователя?
- Как узнать, какие из этих активностей подходят для того, что мы собираемся сделать?
- Как узнать, как использовать эти активности?

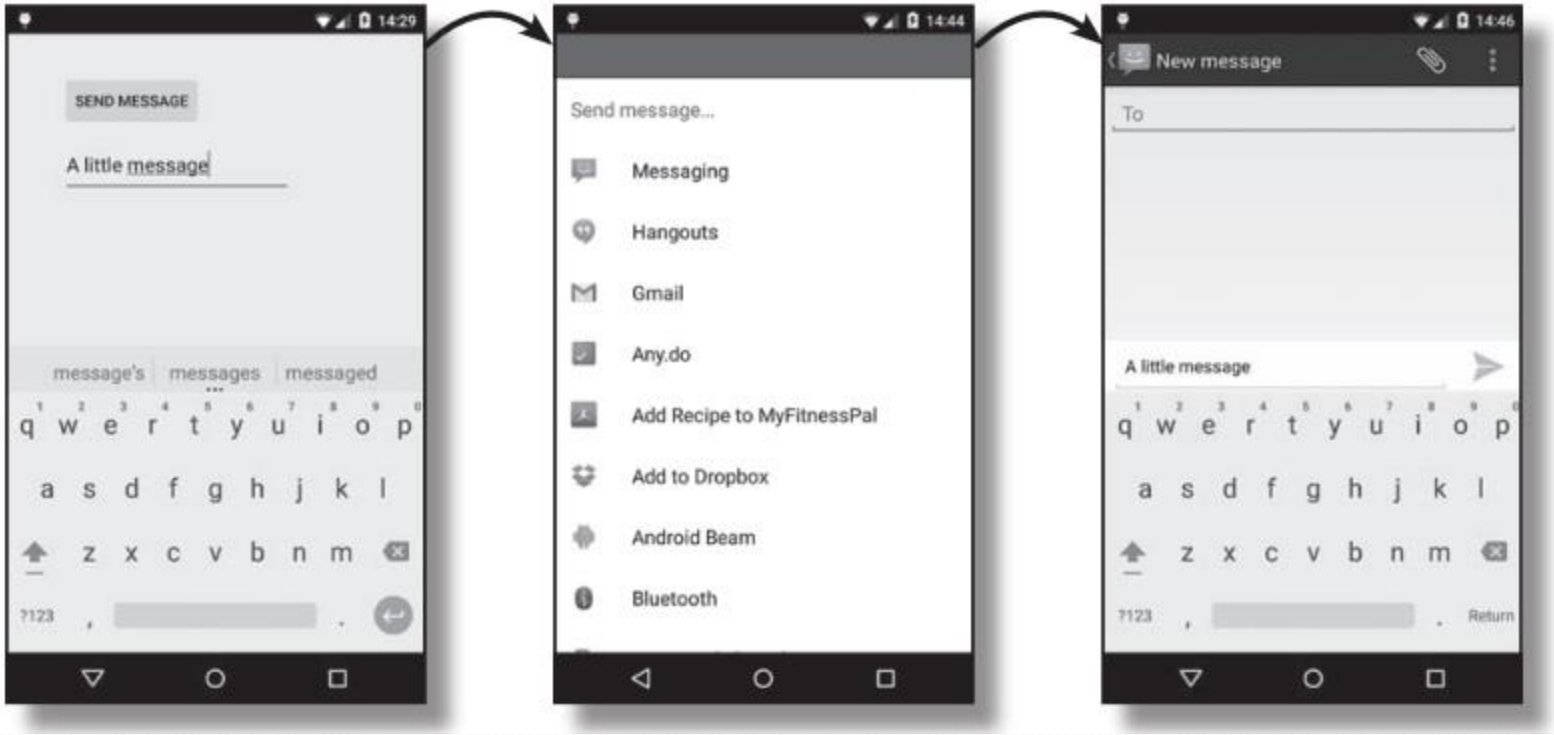
Actions

К счастью, все эти проблемы решаются при помощи Actions. Actions — стандартный механизм, при помощи которого Android узнает о том, какие стандартные операции могут выполняться активностями. Например, Android знает, что все активности, зарегистрированные для действия **send**, могут отправлять сообщения. А теперь нужно научиться создавать интенты, использующие действия для получения набора активностей, которые могут использоваться для выполнения стандартных функций — например, для отправки сообщений.

Наш новый план

1. **Создать интент с указанием действия.**
Интент сообщит Android, что вам нужна активность, умеющая отправлять сообщения. Интент будет включать текст сообщения.

2. **Разрешить пользователю выбрать используемое приложение.**
Скорее всего, на устройстве установлено сразу несколько приложений, способных отправлять сообщения, поэтому пользователь должен выбрать одно из них. Мы хотим, чтобы пользователь мог выбрать приложение каждый раз, когда он щелкает на кнопке Send Message.



Явный и неявный Intent

Ранее мы видели, как создать интент для запуска конкретной активности командой вида

```
Intent intent = new Intent(this, ReceiveMessageActivity.class);
```

Такие интенты называются **явными**; вы явно сообщаете Android, какой класс должна запустить система. Если требуется выполнить некоторое **действие** и вас не интересует, какой активностью оно будет выполнено, создайте **неявный** интент.
При этом вы сообщаете Android, какое действие нужно выполнить, а все подробности по выбору активности, выполняющей это действие, поручаются Android.

Неявный интент

Для создания неявного интента с указанием действия применяется следующий синтаксис:

Intent intent = new Intent(действие);

где действие — тип действия, выполняемого активностью. Android предоставляет целый ряд стандартных вариантов действий. Например, действие Intent.ACTION_DIAL используется для набора номера, Intent.ACTION_WEB_SEARCH — для выполнения веб-поиска, а Intent.ACTION_SEND — для отправки сообщений. Итак, если вы хотите создать интент для отправки сообщения, используйте команду следующего вида: **Intent intent = new Intent(Intent.ACTION_SEND);**

Добавление данных

После определения действия в Intent как и раньше можно включить дополнительную информацию. Допустим, вы хотите добавить текст. Задача решается следующим фрагментом кода:

```
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, текст);
```

где текст — отправляемый текст. Вызов сообщает Android, что активность должна уметь обрабатывать данные с типом данных MIME “text/plain”, а также передает сам текст.

Неявный Intent на практике

```
//Вызвать onSendMessage() при щелчке на кнопке
public void onSendMessage(View view) {
    EditText messageView = (EditText)findViewById(R.id.message);
    String messageText = messageView.getText().toString();
    Intent intent = new Intent(this, ReceiveMessageActivity.class);
    intent.putExtra(ReceiveMessageActivity.EXTRA_MESSAGE, messageText);
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("text/plain");
    intent.putExtra(Intent.EXTRA_TEXT, messageText);
    startActivity(intent);
}
}
```

Удалите эти две строки.

Вместо того, чтобы создавать интент, предназначенный конкретно для ReceiveMessageActivity, мы создаем интент с указанием действия отправки.

Схема. Шаг 1.

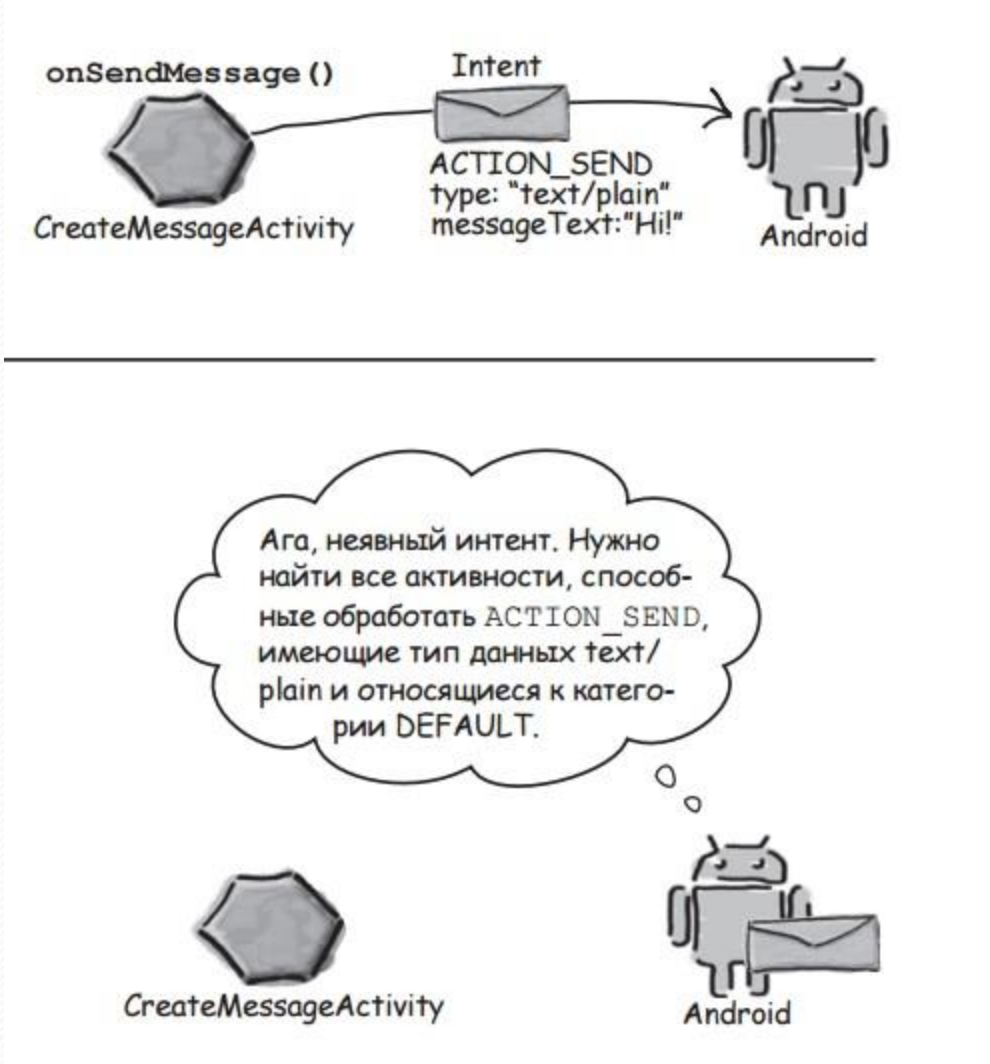

onSendMessage()
Intent
ACTION_SEND
type: "text/plain"
messageText:"Hi!"
CreateMessageActivity
Android
Ага, неявный интент. Нужно найти все активности, способные обработать ACTION_SEND, имеющие тип данных text/plain и относящиеся к категории DEFAULT.
CreateMessageActivity
Android

Схема. Шаг 2. Вариант 1.

Если только одна активность способна обработать интент, Android приказывает этой актив- ности запуститься и передает ей интент.

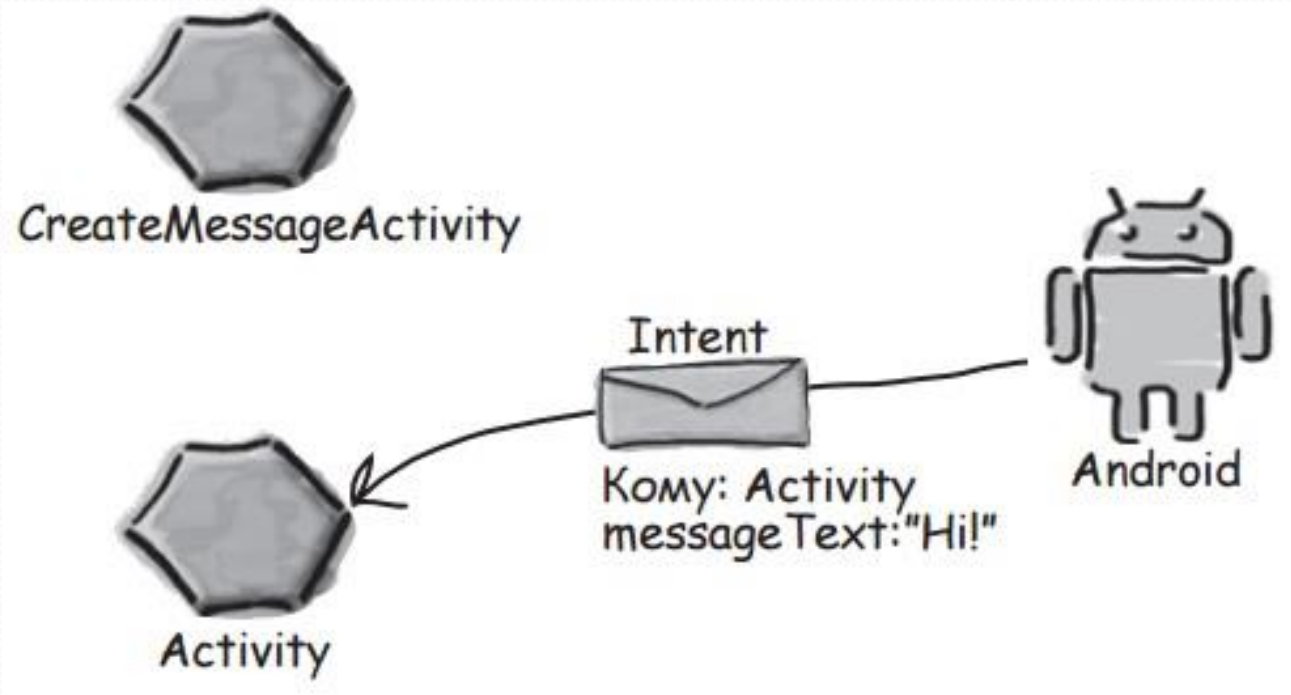
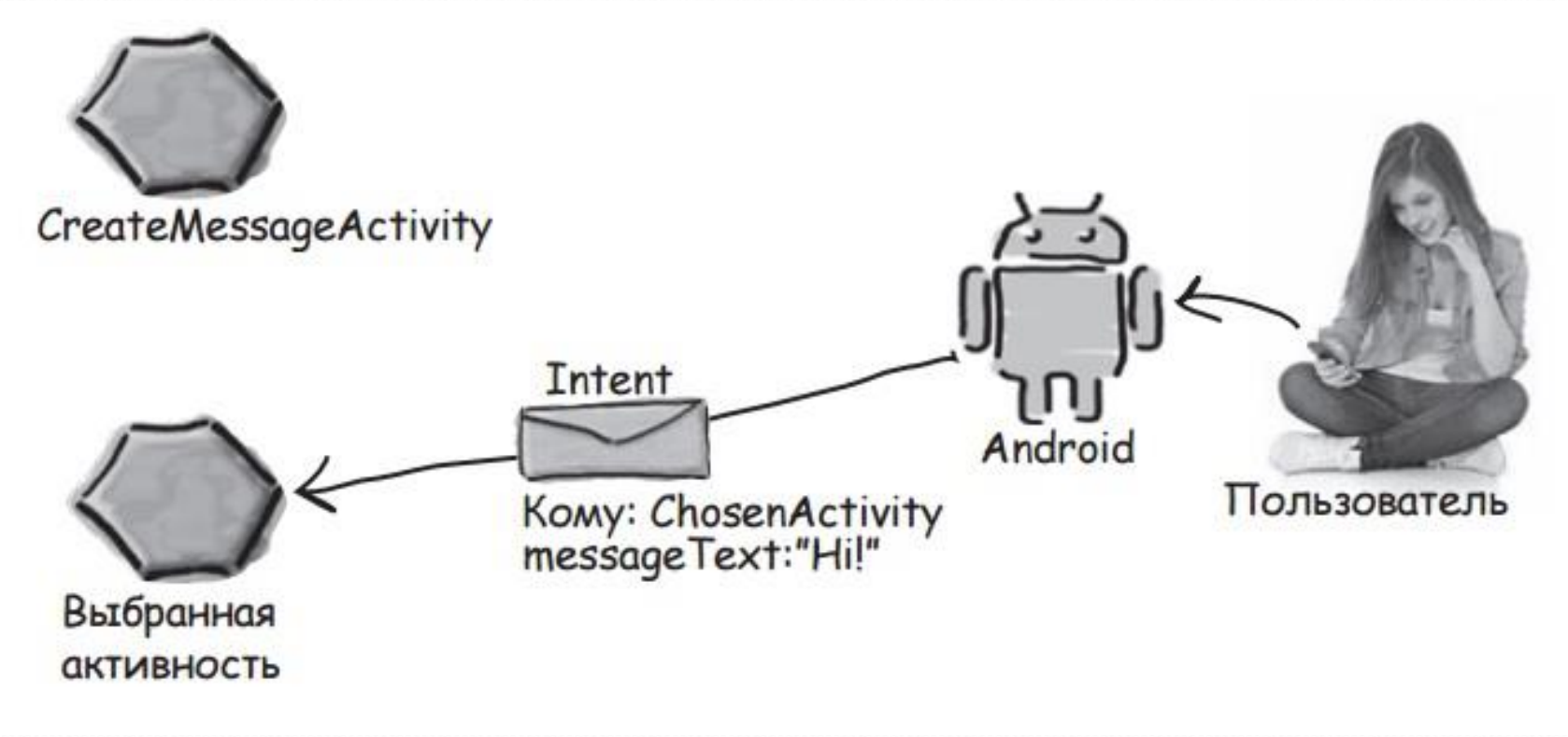


Схема. Шаг 2. Вариант 2.

Если найдется несколько активностей, способных обработать интент, Android открывает диалоговое окно для выбора активности и предлагает пользователю выбрать.

Схема. Шаг 2. Продолжение

Когда пользователь выберет активность, которую он хочет использовать, Android приказывает активности запуститься и передает ей интент.



Способности Activity

Чтобы создать диалоговое окно для выбора активности, система Android должна знать, какие активности способны получить интент.

При получении интента система Android должна определить, какая активность (или активности) может этот интент обработать. Этот процесс называется разрешением интента.

Intent фильтры

При использовании неявного интента система Android использует информацию, содержащуюся в интенте, для определения того, какие компоненты могут его получить. Для этого Android проверяет фильтры интентов, содержащиеся в экземплярах AndroidManifest.xml всех приложений.

Фильтр интентов указывает, какие типы интентов могут обрабатываться каждым компонентом.

Intent-filter в Activity

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
        <data android:mimeType="image/*"/>
    </intent-filter>
</activity>
```

Сообщает Android, что активность может обрабатывать ACTION_SEND.

Фильтр интентов должен включать категорию DEFAULT; в противном случае он не сможет получать неявные интенты.

Типы данных, которые могут обрабатываться активностью.

Intent фильтры

Получив неявный интент, Android сравнивает информацию из интента с информацией, содержащейся в фильтрах интентов из файла AndroidManifest.xml каждого приложения.

Порядок проверки фильтров

1. Сначала Android рассматривает фильтры интентов, включающие категорию android.intent.category.DEFAULT. Фильтры интентов без этой категории пропускаются, так как они не могут получать неявные интенты.

2. Затем Android сопоставляет интенты с фильтрами интентов, сравнивая действия и тип MIME из интента с указанными в фильтрах.

После проверки фильтров

После того как сравнение интента с фильтрами интентов, назначенных компонентам, будет завершено, Android смотрит, сколько совпадений удалось найти. Если найдено только одно совпадение, Android запускает компонент (в нашем случае это активность) и передает ему интент. Если будет найдено несколько совпадений, Android просит пользователя выбрать один из вариантов.

Резюме

- Новая активность создается так: File → New... → Activity. ☐
- Для каждой активности в файле AndroidManifest.xml должна быть создана запись. ☐
- Интент представляет собой разновидность сообщений, используемых для организации взаимодействия между компонентами Android. ☐
- Явный интент предназначен для конкретного компонента.
- Явный интент создается командой Intent intent = new Intent(this, Target.class); ☐
- Активности запускаются вызовом startActivity(intent).
- Метод putExtra() включает дополнительную информацию в интент. ☐

Резюме

- Метод getIntent() для получения интента, запустившего активность. ☐
- Используйте методы get*Extra() для чтения дополнительной информации, связанной с интентом.
- Метод getStringExtra() читает String, getIntExtra() читает int, и т. д. ☐
- Действие описывает стандартную операцию, которую может выполнять активность. Так, для отправки сообщений используется обозначение Intent.ACTION_SEND. ☐
- Чтобы создать неявный интент с указанием действия, используйте запись Intent intent = new Intent(action);