



CHƯƠNG 1: Tổng quan về lập trình Android(tt)

GV: Trương Bá Thái

ĐT: 0932.577765

Email: truongbathai@tdc.edu.vn



| Nội dung

- Quản lý trạng thái Activity.
 - Xây dựng Activity
 - Vòng đời của Activity
- Debug chương trình trong Android studio





1.5. Quản lý trạng thái Activity.



| 1.5.1. Xây dựng Activity

- Trong ứng dụng Android, Activity đóng vai trò là một màn hình, nơi người dùng có thể tương tác với ứng dụng, ví dụ: chụp hình, xem bản đồ, gửi mail...
- Một ứng dụng có thể có một hoặc nhiều Activity, Activity được khởi chạy đầu tiên khi ứng dụng hoạt động được gọi là “MainActivity”.

| 1.5.1. Xây dựng Activity

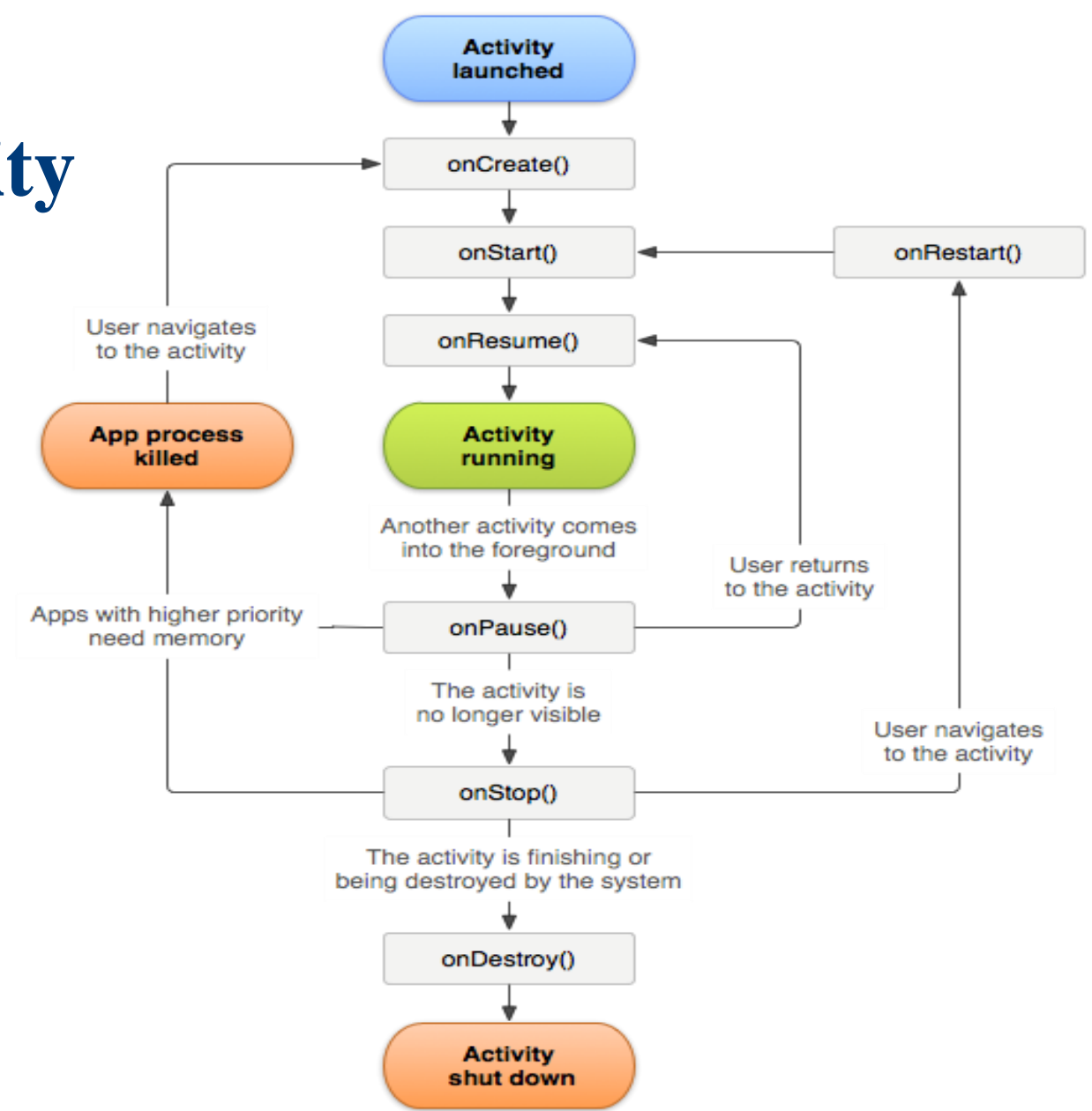
- Activity có thể hiển thị ở chế độ toàn màn hình, hoặc ở dạng cửa sổ với một kích thước nhất định.
- Các Activity có thể gọi đến các Activity khác, Activity được gọi sẽ nhận được tương tác ở thời điểm đó.

| 1.5.1. Xây dựng Activity

- Thực hiện tạo Activity cho ứng dụng:
 - Tạo mới lớp kế thừa từ lớp Activity
 - Thực thi các phương thức quản lý trạng thái Activity
 - Xây dựng giao diện trong tài nguyên **res/layout**
 - Khai báo Activity trong tập tin AndroidManifest.xml

1.5.2. Vòng đời của Activity

➤ Sơ đồ trạng thái



1.5.2. Vòng đời của Activity

- Activity bao gồm bốn trạng thái:
 - Resumed: đang trong trạng thái nhận tương tác.
 - Paused: không thể tương tác nhưng vẫn được thấy bởi người dùng.
 - Stopped: thực hiện chạy ở chế độ ngầm.
 - *Inactive*: Khi hệ thống bị thiếu bộ nhớ, nó sẽ giải phóng các tiến trình theo nguyên tắc ưu tiên.

| 1.5.2. Vòng đời của Activity

➤ Thực hiện gọi các phương thức quản lý trạng thái:

- onStart
- onRestart
- onCreate
- onPause

- onResume
- onStop
- onDestroy

| 1.5.2. Vòng đời của Activity

❖ Entire lifetime:

– Từ phương thức onCreate() cho tới onDestroy(): từ lúc Activity được gọi ra cho đến lúc bị huỷ.

❖ Visible lifetime:

- Từ sau khi gọi onStart() cho tới lúc gọi onStop() : trong trường hợp này ta vẫn có thể thấy màn hình Activity.

❖ *Foreground lifetime:*

- Xảy ra từ khi gọi onResume() cho tới lúc gọi onPause(): trong suốt thời gian này Activity luôn nằm ở trên cùng và ta có thể tương tác được với nó.

1.6. Debug chương trình trong Android studio




| Sử dụng Log và Toast

- Khi thực thi ứng dụng muốn kiểm tra xem đã xảy ra lỗi gì hay hệ thống có cảnh báo gì, hoặc chỉ là các thông tin thực thi bình thường chúng ta có thể sử dụng LogCat.
- Trong quá trình xây dựng ứng dụng, nếu muốn in ra giá trị của một biến đơn nào đó tại một thời điểm bất kỳ chúng ta có sử dụng:
 - Log
 - Toast

Debug

- Cũng như nhiều IDE khác, Android Studio có cung cấp khả năng debug ứng dụng rất hiệu quả cho các ứng dụng chạy trên máy thật lẫn máy ảo. Bạn có thể:
 - Chọn thiết bị để debug.
 - Đặt các breakpoint (điểm dừng) trong code.
 - Quan sát và kiểm tra các giá trị biến / biểu thức trong runtime.
 - Chụp ảnh màn hình ứng dụng

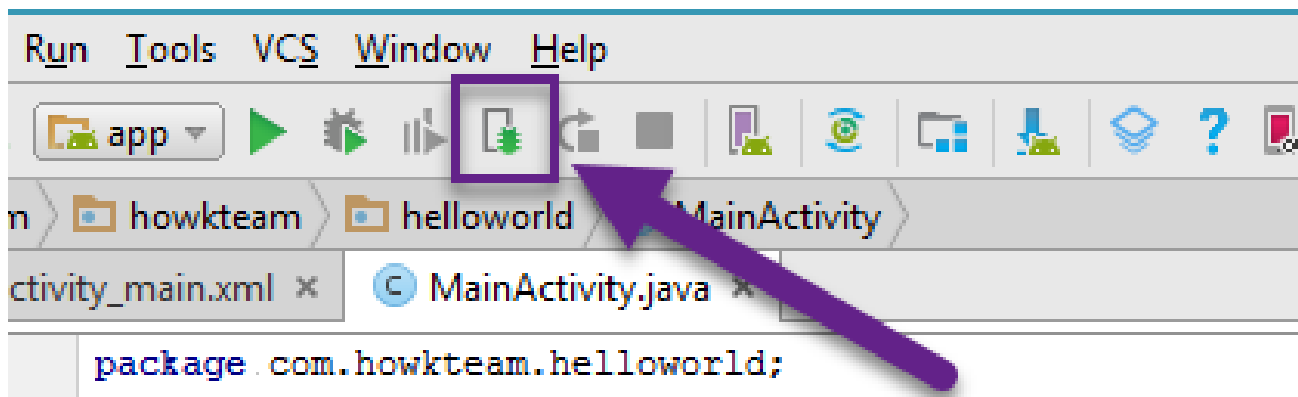
| Ứng dụng của bạn chưa được chạy ở chế độ Debug

- Để bắt đầu debug, các bạn nhấn vào nút **Debug**  trên thanh công cụ (khi trỏ vào sẽ có chữ “Debug ‘app’”).
- Lúc này Android Studio sẽ build ứng dụng ra file APK, ký (sign) file APK bằng key debug, và cài đặt lên thiết bị của bạn. Cuối cùng, cửa sổ **Debug** sẽ được mở ra:

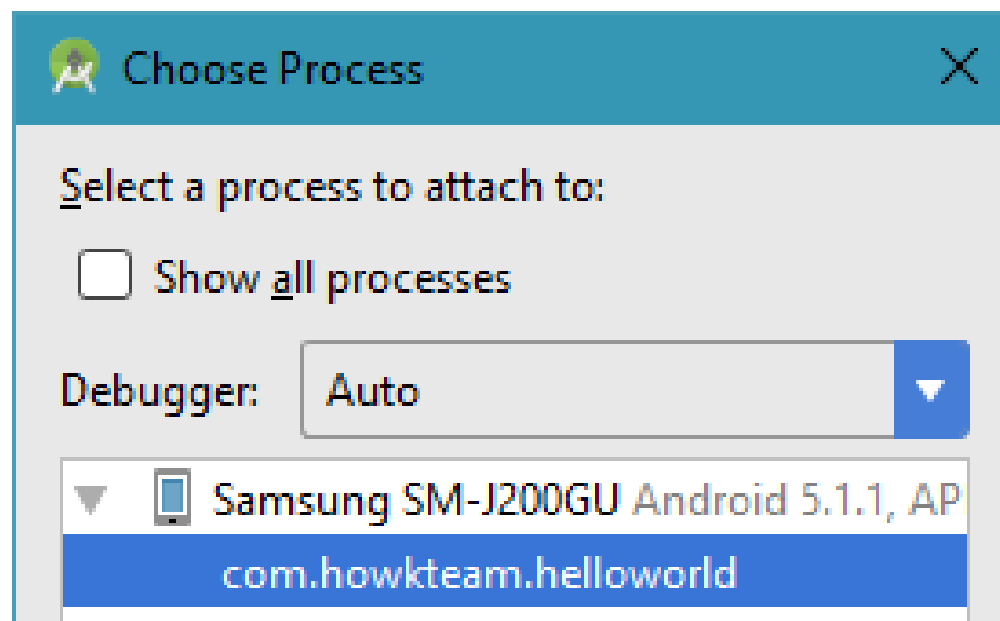
| Ứng dụng của bạn đã chạy (trên máy ảo/máy thật)

➤ Trong trường hợp này, bạn có thể bắt đầu Debug bằng các bước sau:

➤ Click vào nút **Attach Debugger to Android Process**:



➤ Cửa sổ **Choose Process** sẽ hiện lên. Lúc này bạn sẽ chọn process tương ứng để trình Debug của Android Studio theo dõi nó



| Bước 1: Đặt Log trong code:

➤ Đặt Log trong code

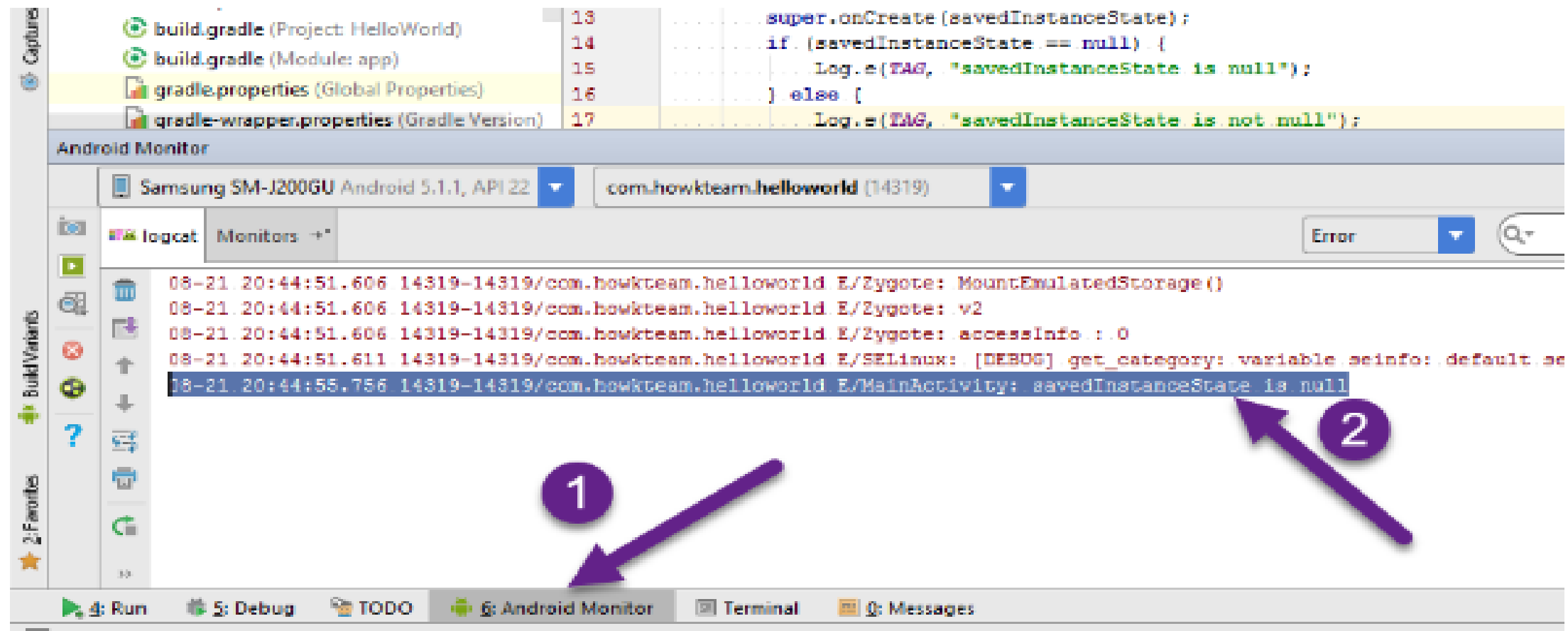
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (savedInstanceState == null) {
        Log.e(TAG, "savedInstanceState is null");
    } else {
        Log.e(TAG, "savedInstanceState is not null");
    }

    setContentView(R.layout.activity_main);
}
```

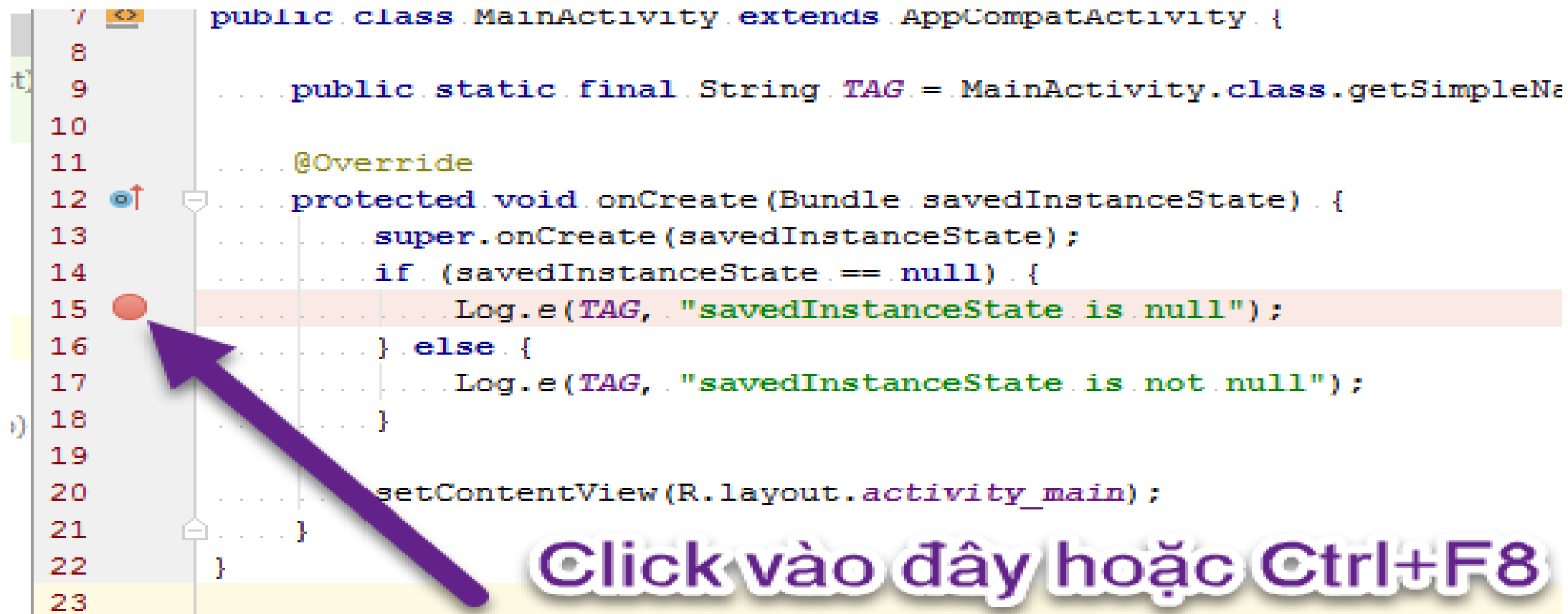
```
.....@Override
.....protected void onCreate(Bundle savedInstanceState) {
? android.util.Log? Alt+Enter savedInstanceState);
.....if (savedInstanceState == null) {
.....    Log.e(TAG, "savedInstanceState is null");
.....} else {
.....    Log.e(TAG, "savedInstanceState is not null");
.....}

.....setContentView(R.layout.activity_main);
.....}
}
```

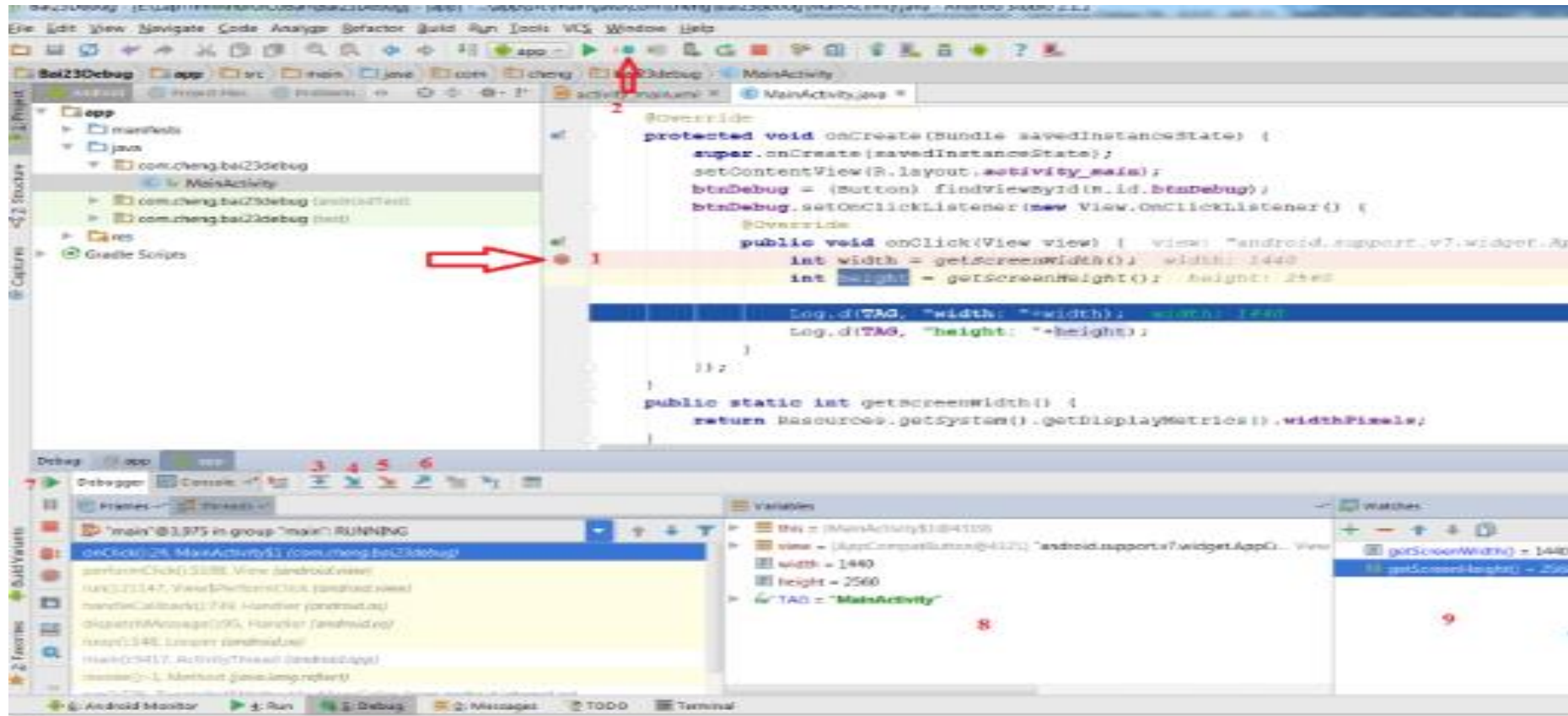
Bước 2: Đọc Log



Làm việc với Breakpoint



Debug



- Số 1: cái này gọi là **BreakPoint**, chính là điểm mà muốn debug, khi ứng dụng chạy ở chế độ debug thì chạy tới đây nó mới vào Debug cho các bạn xem.
- Số 2: Khởi chạy ứng dụng ở chế độ debug.
- Số 3: **Step Over** -> khi bấm vào đây thì ứng dụng sẽ chạy vào dòng code tiếp theo thứ tự từ trên xuống, nó sẽ không can thiệp sâu vào code bên trong từng phương thức.

➤ **Số 4: Step Into** -> ứng dụng sẽ chạy sâu vào bên trong phương thức của một đối tượng nào đó, và chạy từng dòng gặp phương thức nào lại đi sâu vào phương thức đó cho đến hết.



➤ **Số 5: Force Step Into** -> ứng dụng sẽ chạy vào bên trong phương thức vào sau đó ra ngoài mà không chạy từng dòng lệnh trong phương thức đó, nó khác Step Into ở chỗ đó.



➤ **Số 6: Step Out** -> chạy từ đầu đến cuối, nếu có **Breakpoint** thì sẽ chạy vào, nếu không có thì chế độ debug sẽ dừng ứng dụng thoát ra. Nếu đang trong phương thức thì sẽ tiến hành chạy ra khỏi phương thức đó.



➤ **Số 7: Resume Program** -> chạy từ đầu đến cuối và chỉ chạy vào chỗ nào có Breakpoint thôi, nếu không có thì chế độ debug sẽ dừng ứng dụng vẫn chạy.



➤ Số 8: đây là nơi để show ra các giá trị tương ứng của các đối tượng tương ứng với vị trí line hiện tại của debug đang chạy đến.



- Số 9: xem bất kì giá trị của một đối tượng nào đó khi debug chạy đến vị trí đối tượng đó.



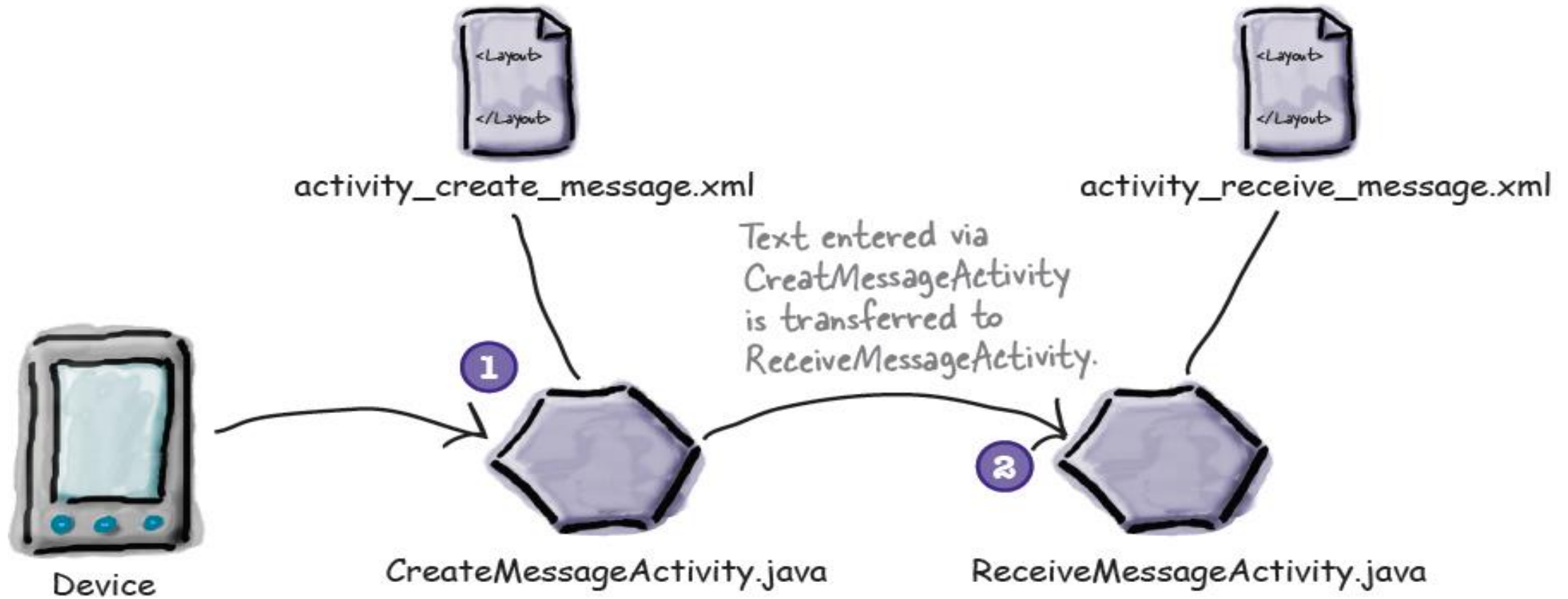
| Hướng dẫn thực hành

➤ Xây dựng ứng dụng



| Các bước thực hiện

- 1 Create a basic app with a single activity and layout.
- 2 Add a second activity and layout.
- 3 Get the first activity to call the second activity.
- 4 Get the first activity to pass data to the second activity.

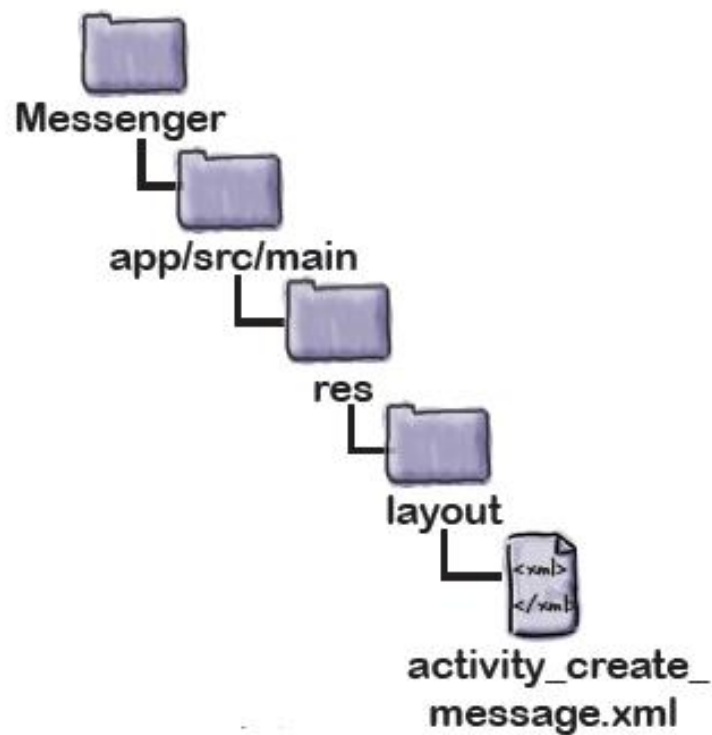


| Create the project

- Sử dụng minimum SDK API 16 .
- Đặt tên Project : CreateMessageActivity



Thiết kế lại Layout :



```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".CreateMessageActivity" >

```

Replace the
<TextView>
Android
Studio gives
you with the
<Button> and
<EditText>.

```

    <Button
        android:id="@+id/send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="36dp"
        android:layout_marginTop="21dp"
        android:onClick="onSendMessage"
        android:text="@string/send" />

```

Clicking on the button runs the
onSendMessage() method in the activity.

```

    <EditText
        android:id="@+id/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/send"
        android:layout_below="@+id/send"
        android:layout_marginTop="18dp"
        android:ems="10" />

```

This is a String
resource.

```

</RelativeLayout>

```

This describes how wide the <EditText>
should be. It should be wide enough to
accommodate 10 letter M's.



The <EditText> element
defines an editable text
field for entering text. It
inherits from the same
Android View class as the
other GUI components
we've seen so far.

| Update strings.xml...

```
<string name="send">Send Message</string>
```



```

public class CreateMessageActivity extends Activity {

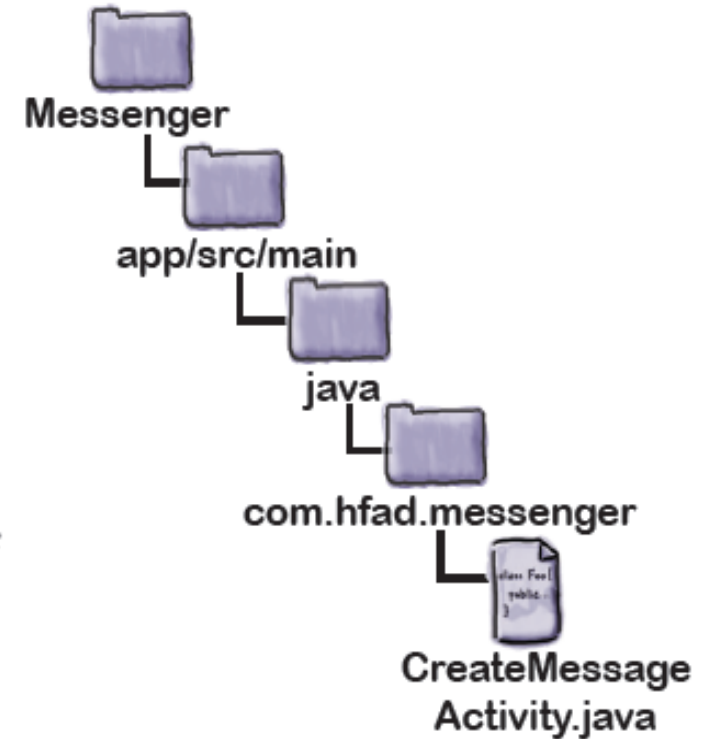
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_message);
    }

    //Call sendMessage() when the button is clicked
    public void sendMessage(View view) {
    }
}

```

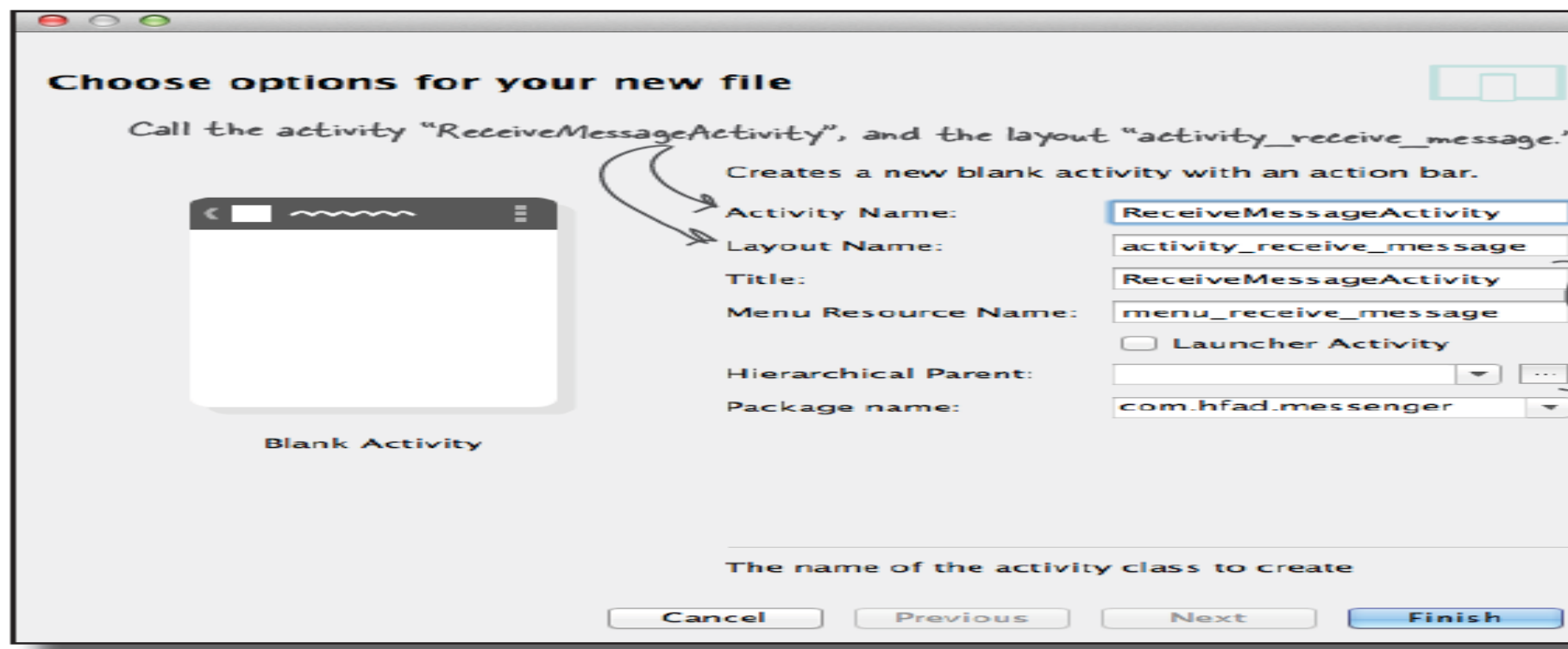
The onCreate() method gets called
when the activity is created.

This method will get called when the
button's clicked. We'll complete the method
body as we work our way through the rest
of the chapter.



Tạo activity and layout thứ 2

➤ File → New → Activity; chọn Blank Activity





activity_create_message.xml



CreateMessageActivity.java



activity_receive_message.xml



RecieveMessageActivity.java

Android manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hfad.messenger" > ← This is the package
                                   name we specified.

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher" ← Android Studio gave our
        android:label="@string/app_name"      app a default icon. We'll
        android:theme="@style/AppTheme" > ← The theme affects the
                                           appearance of the app.
                                           We'll look at this later.

        <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>
        ← This is the first activity, Create Message Activity.

        <activity
            android:name=".ReceiveMessageActivity"
            android:label="@string/title_activity_receive_message" >
        </activity>
        ← This is the second activity, Receive Message Activity.

    </application>
</manifest>
```

Android Studio added these lines for us when we added the second activity.

This bit specifies that it's the main activity of the app.

This says the activity can be used to launch the app.

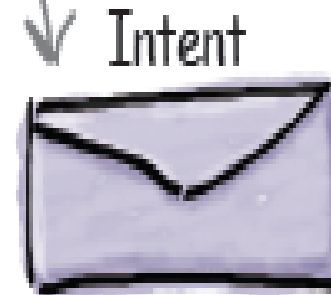
An intent is a type of message

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

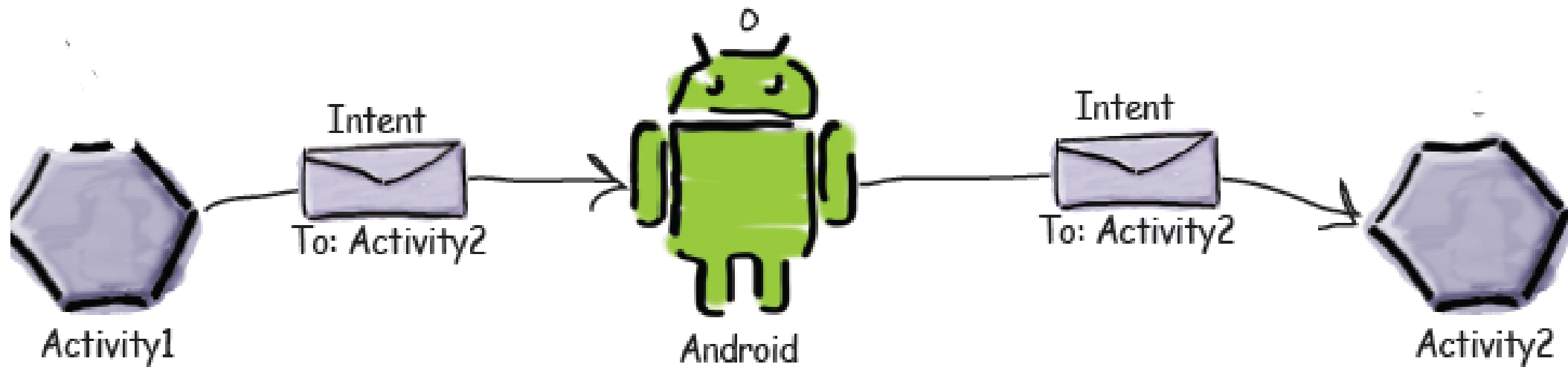
```
startActivity(intent);
```

The intent specifies the activity you want to receive it. It's like putting an address on an envelope.

startActivity() starts the activity specified in the intent.



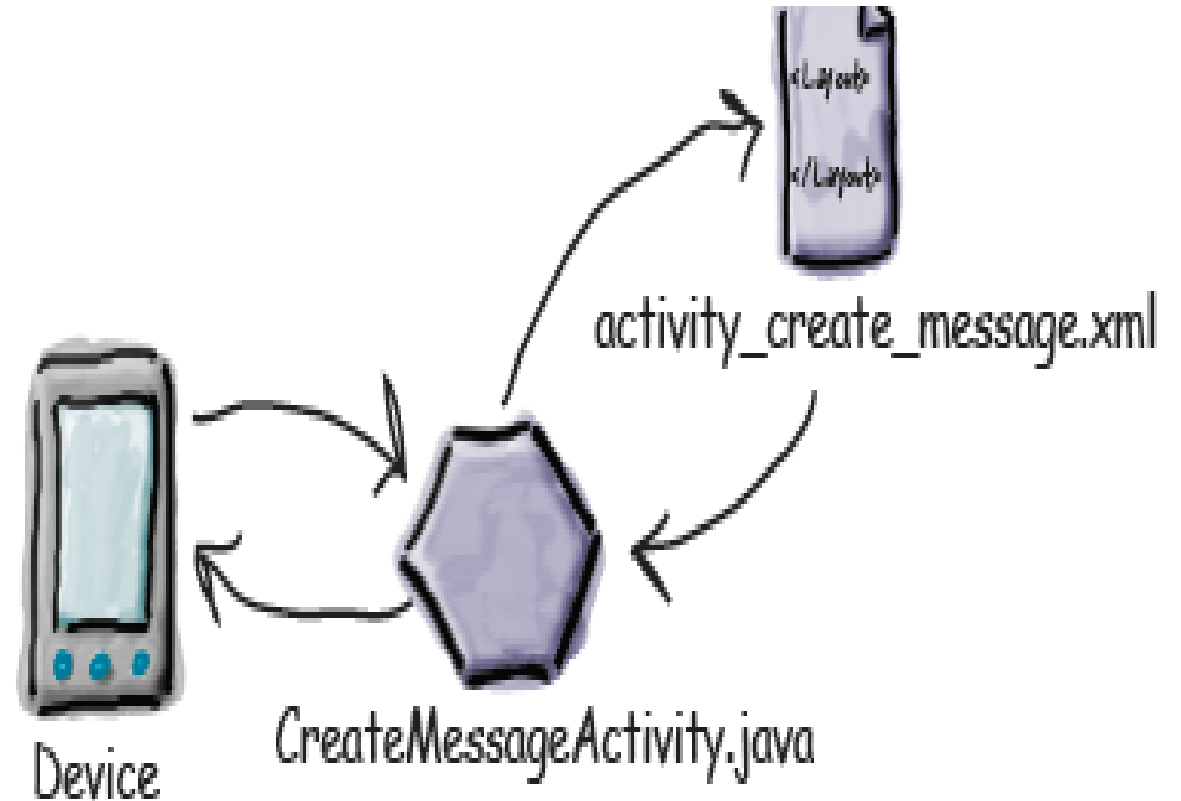
To: AnotherActivity



What happens when you run the app

1 When the app gets launched, the main activity, **CreateMessageActivity** starts.

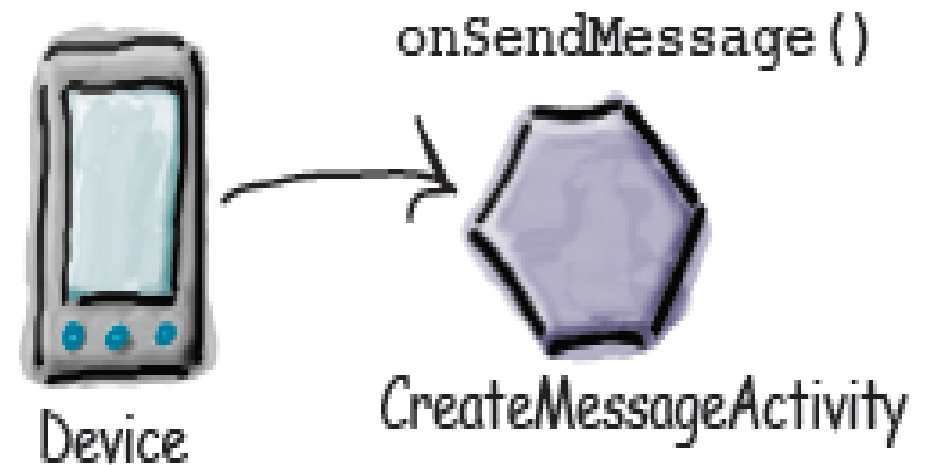
When it starts, the activity specifies that it uses layout *activity_create_message.xml*. This gets displayed in a new window.



2

The user clicks on a button.

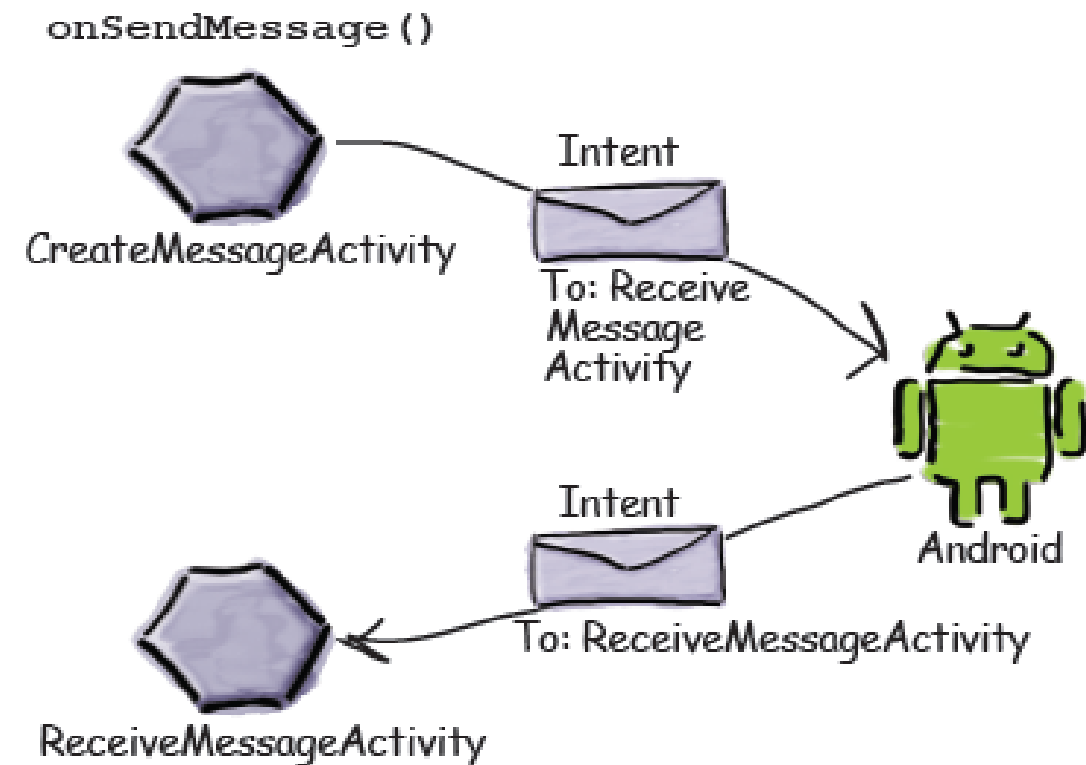
The `onSendMessage()` method in `CreateMessageActivity` responds to the click.



3

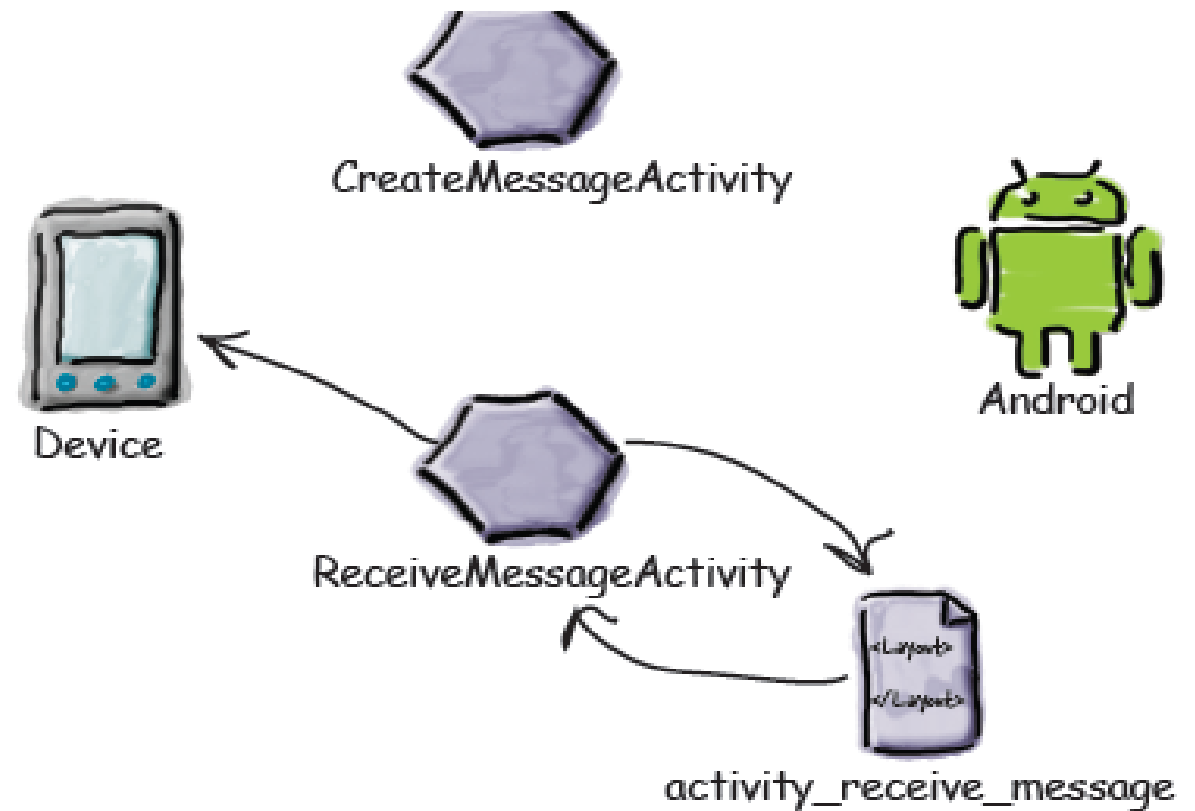
The `onSendMessage()` method tells Android to start activity `ReceiveMessageActivity` using an intent.

Android checks that the intent is OK, and then it tells `ReceiveMessageActivity` to start.



4

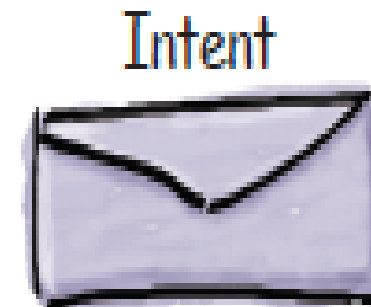
When `ReceiveMessageActivity` starts, it specifies that it uses layout `activity_receive_message.xml` and this gets displayed in a new window.



| putExtra() puts extra information in an intent

```
intent.putExtra("message", value);
```

putExtra() lets
you put extra
information in
the message
you're sending.



To: ReceiveMessageActivity
message: "Hello!"

//Call onSendMessage() when the button is clicked

```
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);
```

```
    startActivity(intent);
```

```
}
```

Get the text that's in
the EditText



retrieve extra information from an intent

```
Intent intent = getIntent();
```

← Get the intent

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

← Get the string passed along with the intent that has a name of "message".



To: ReceiveMessageActivity
message: "Hello!"



```
public class ReceiveMessageActivity extends Activity {
```

```
    public static final String EXTRA_MESSAGE = "message";
```

↑
This is the name of the extra value we're passing in the intent.

```
    @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);
```

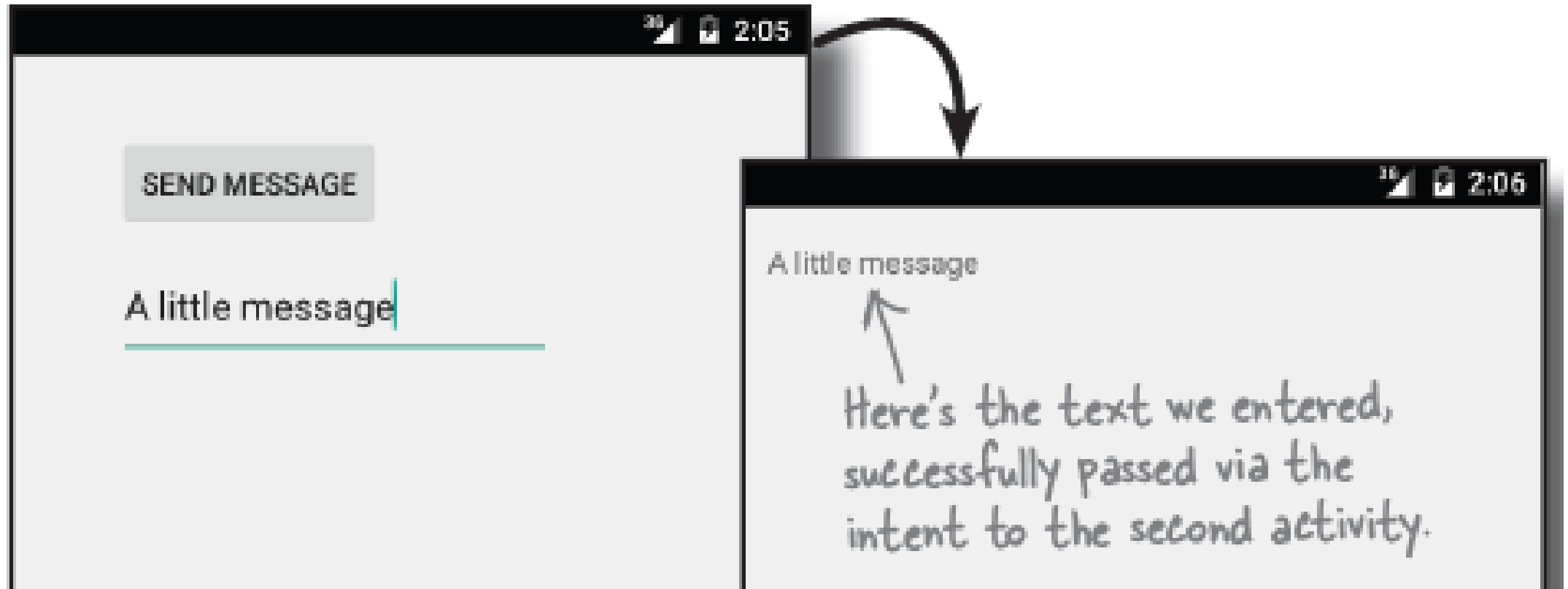
```
        messageView.setText(messageText);
```

```
    }
```

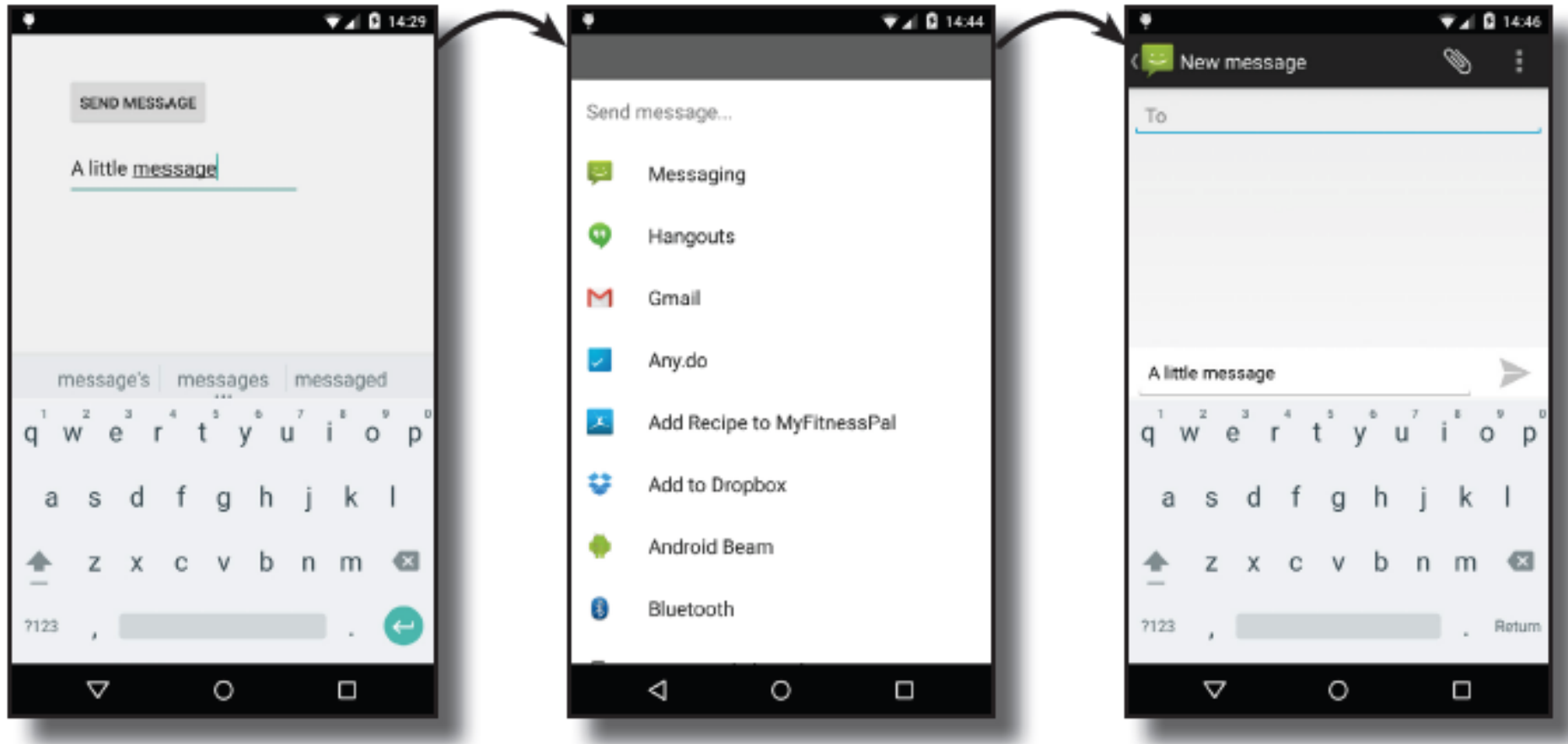
```
}
```

Get the intent, and get the message from it using `getStringExtra()`.

↑
Add the text to the message text view.



Intents can start activities in other apps



//Call onSendMessage() when the button is clicked

```
public void onSendMessage(View view) {
```

```
    EditText messageView = (EditText)findViewById(R.id.message);
```

```
    String messageText = messageView.getText().toString();
```

Remove these
two lines.

```
    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);
```

```
}
```

```
}
```

Instead of creating an intent that's explicitly for ReceiveMessageActivity, we're creating an intent that uses a send action.

| Bài tập thực hành 1:

➤ Đề bài:

✓Viết chương trình sử dụng các phương thức đặt các trạng thái cho Activity.

➤ Mục tiêu:

✓Hiểu và biết cách sử dụng các phương thức đặt các trạng thái cho Activity.

✓Biết cách viết mã lệnh ở vị trí phù hợp để phục vụ cho việc xây dựng các ứng dụng.



➤ Gợi ý thực hiện:

- ✓ Tạo mới Android Application Project, đặt tên là TrangThaiActivity.
- ✓ Mở tập tin MainActivity.java tiến hành thêm các phương thức đặt các trạng thái cho Activity.
- ✓ Ghi các trạng thái vào **Log** để theo dõi.

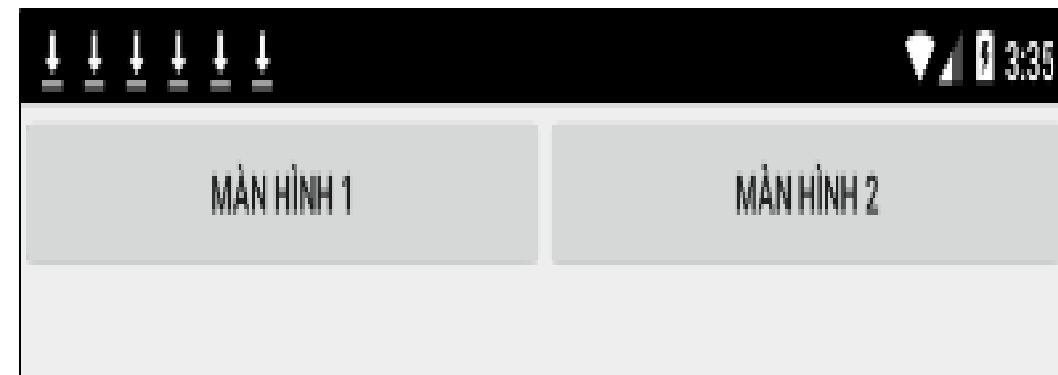


| Bài thực hành 2

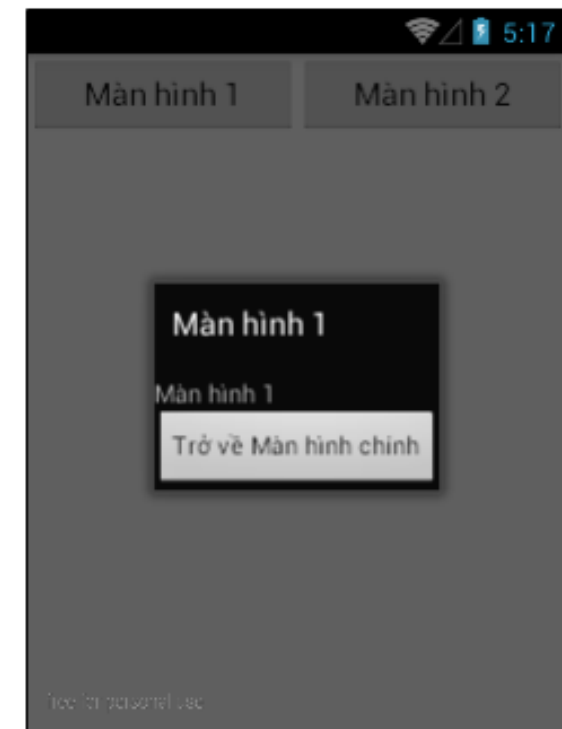
➤ Đề bài:

➤ Tạo 3 Activity đặt tên là: ManHinhChinh, ManHinh1, ManHinh2. Yêu cầu:

- Giao diện màn hình chính như sau:



- Khi Màn hình 1 được kích hoạt thì nó sẽ nằm phía trên Màn hình Chính
 - vẫn thấy được màn hình chính



- Khi ManHinh2 hiển thị thì nó sẽ chiếm toàn bộ màn hình - không thể thấy được màn hình chính.

➤ Bắt các sự kiện khi Activity chuyển đổi trạng thái và thông báo lên màn hình Android.



➤ Mục tiêu:

- Hiểu sâu thêm về các phương thức đặt trạng thái trong Activity.
- Phân biệt được giữa Foreground Lifetime và Visible Lifetime.



➤ Gợi ý thực hiện:

- Tạo mới Android Application Project, đặt tên là VongDoiActivity.
- Tạo 2 activity đặt tên lần lượt như sau: ManHinh1 và ManHinh2.
- Thiết kế giao diện cho ManHinhChinh, ManHinh1, ManHinh2 như yêu cầu của đề bài (sử dụng Button).
- Viết code xử lý trong lớp ManHinhChinh.java.

The End

