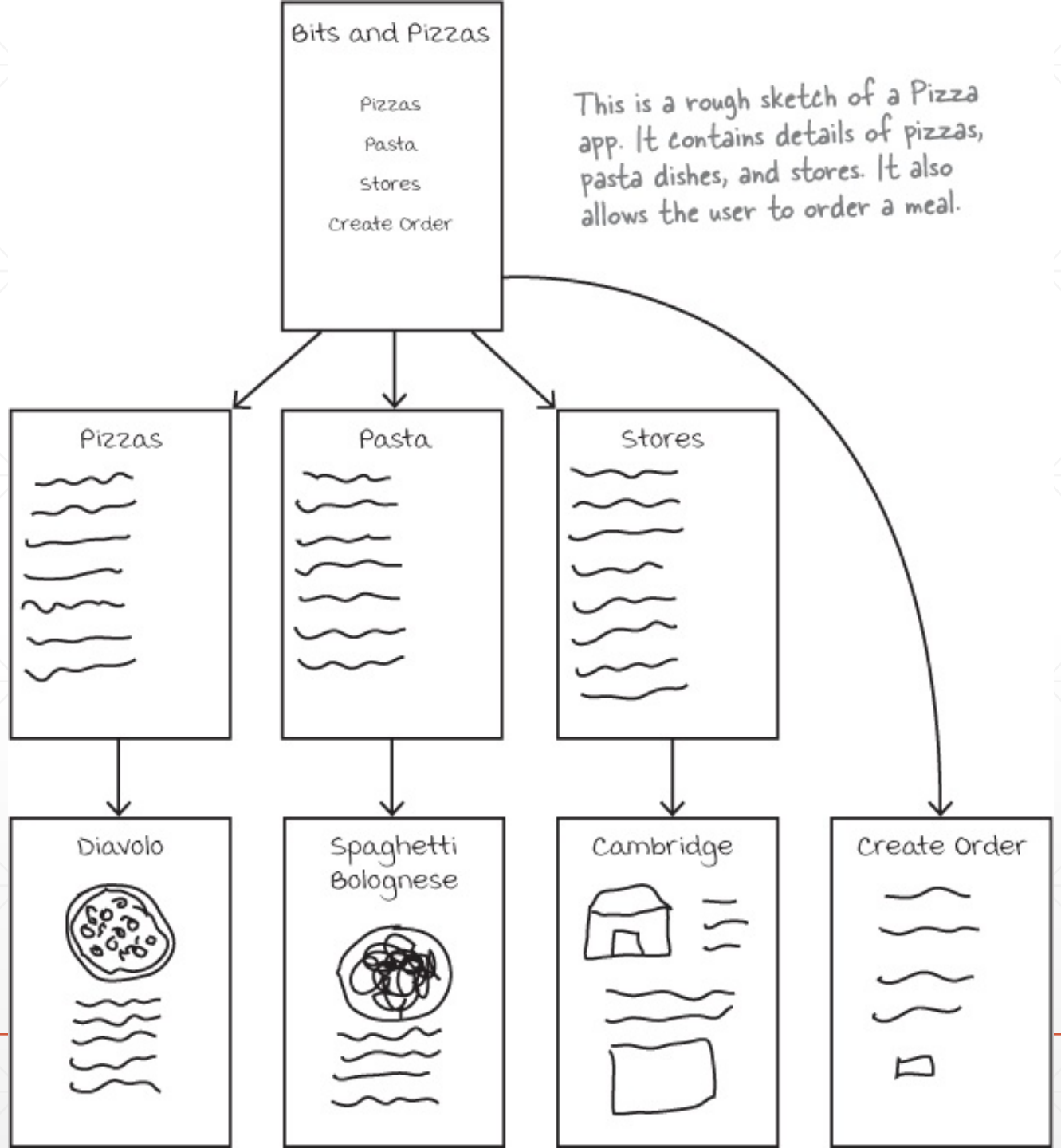
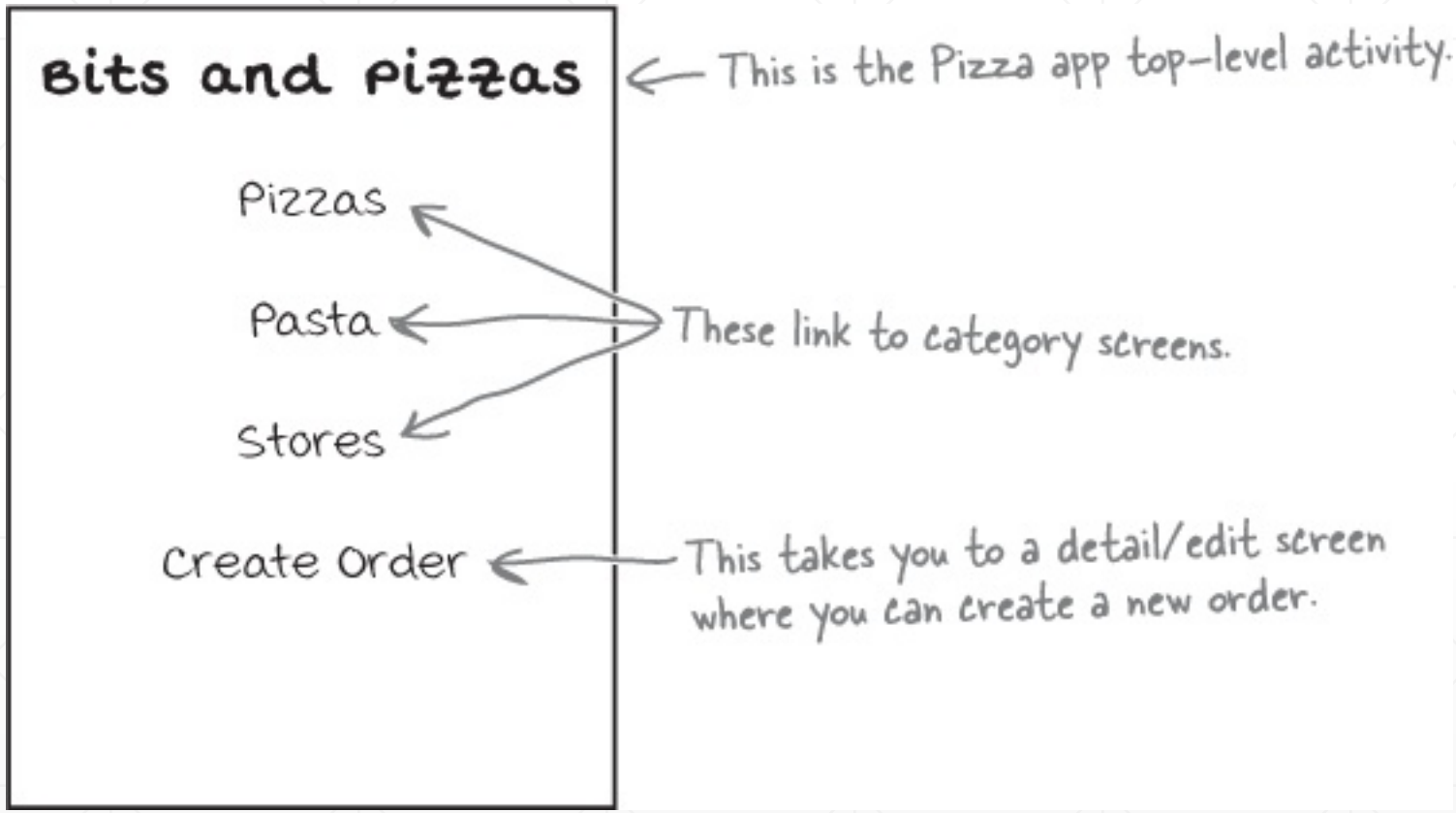
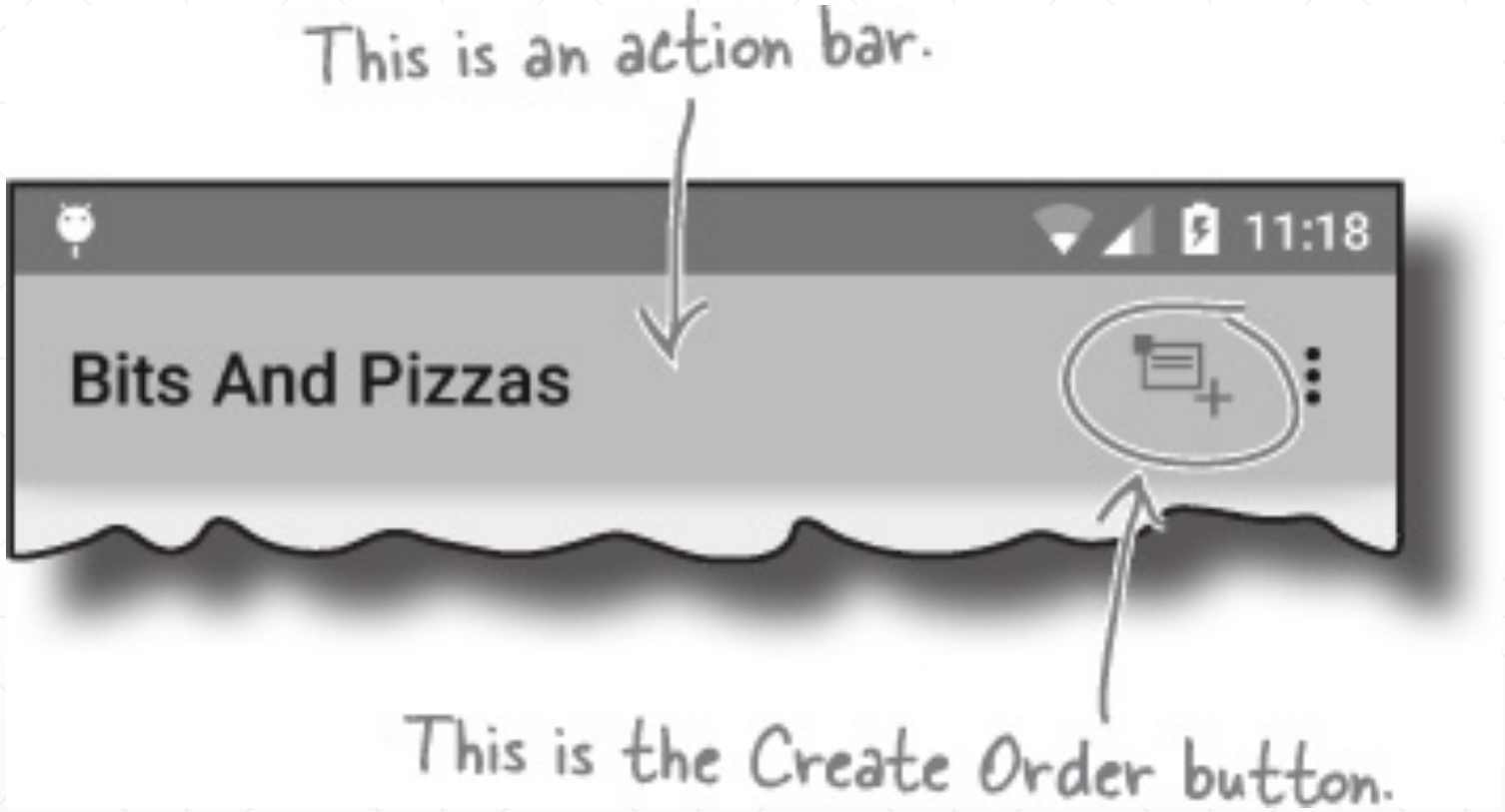
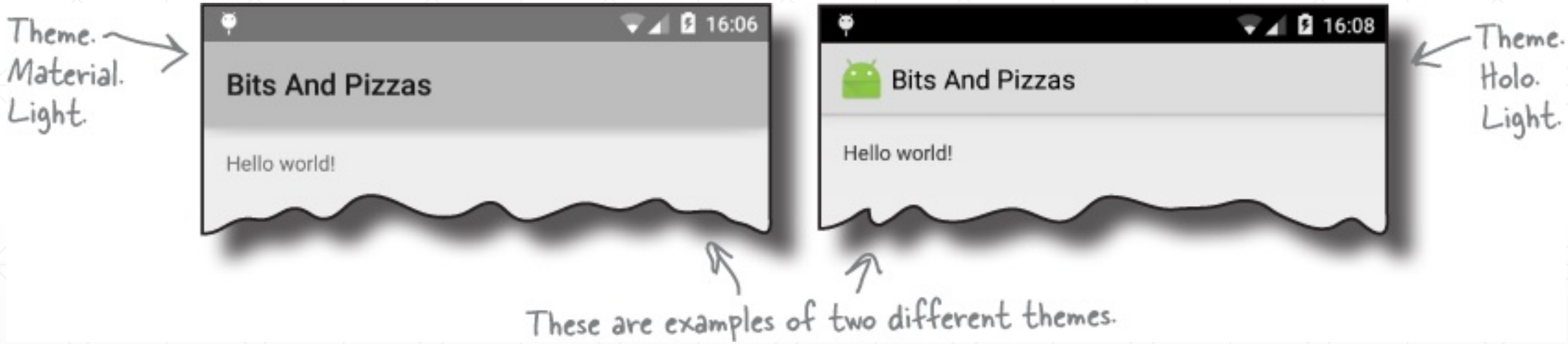

Bits and Pizzas
Pizzas
Pasta
Stores
Create Order
This is a rough sketch of a Pizza app. It contains details of pizzas, pasta dishes, and stores. It also allows the user to order a meal.
Pizzas
Pasta
Stores
Diavolo
Spaghetti Bolognese
Cambridge
Create Order


Bits and Pizzas
This is the Pizza app top-level activity.
Pizzas
Pasta
Stores
These link to category screens.
Create Order
This takes you to a detail/edit screen where you can create a new order.


This is an action bar.
11:18
Bits And Pizzas
This is the Create Order button.


Theme.
Material.
Light.
16:06
Bits And Pizzas
Hello world!
16:08
Bits And Pizzas
Hello world!
Theme.
Holo.
Light.
These are examples of two different themes.

v4 support library
Includes the largest set of features, such as support for application components and user interface features.

v7 appcompat library
Includes support for action bars on API level 7 and above, also creating and using material design.

v7 cardview library
Adds support for the `CardView` widget, allowing you to show information inside cards.

These are just some of the support libraries.

v7 gridlayout library
Adds support for the `GridLayout` class.

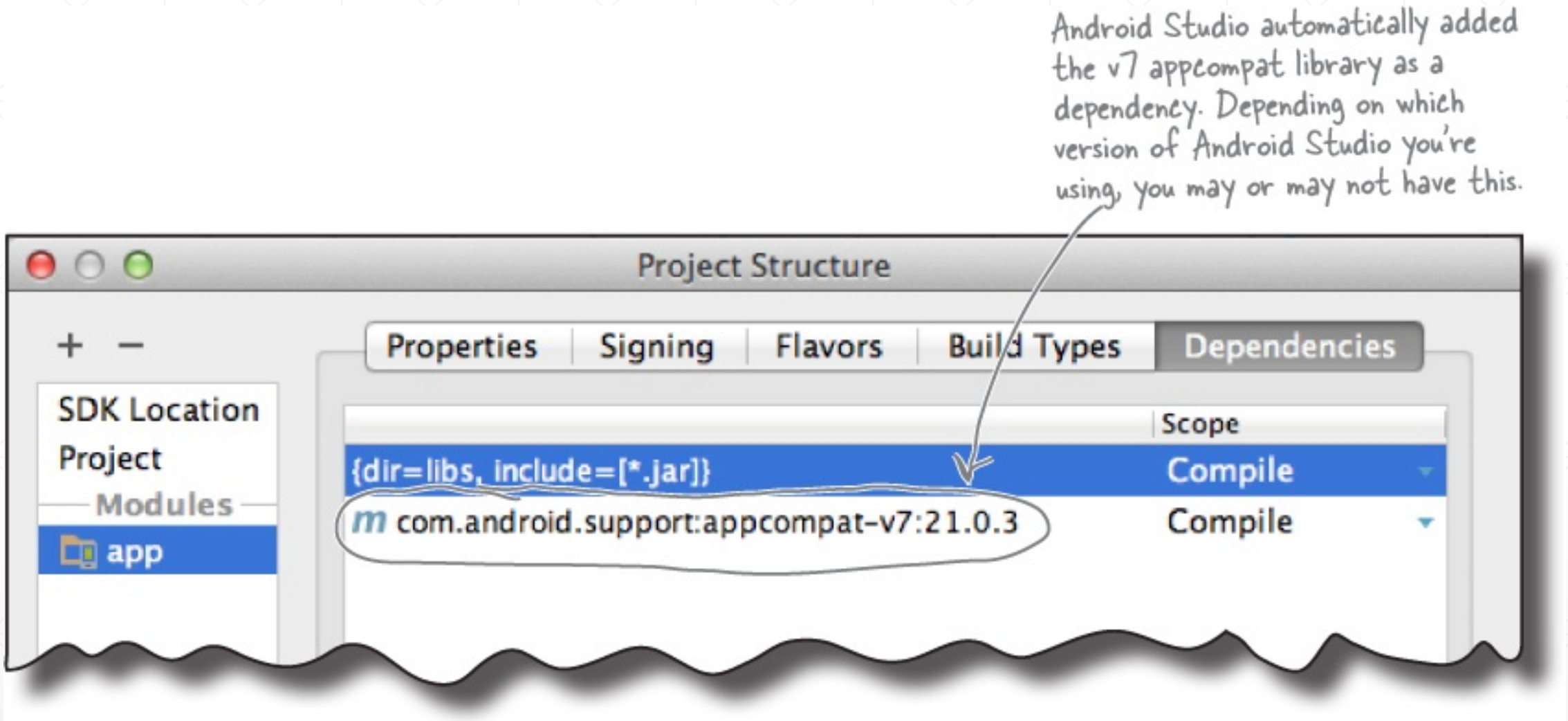
v7 recyclerview library
Adds support for the `RecyclerView` widget.

v17 leanback library
Includes APIs allowing you to build user interfaces for TVs.

```
package com.hfad.bitsandpizzas;

import android.support.v7.app.ActionBarActivity;
...
public class MainActivity extends ActionBarActivity {
    ...
}
```

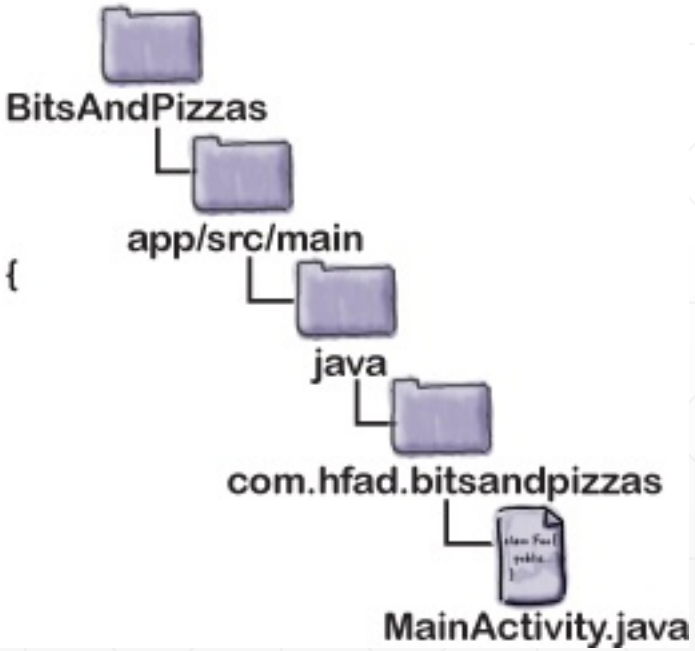
The android.support.v7 in the ActionBarActivity import tells you it's from v7 appcompat library.

Your MainActivity.java code may look different. It depends on the behavior of the IDE you're using.


Android Studio automatically added the v7 appcompat library as a dependency. Depending on which version of Android Studio you're using, you may or may not have this.
Project Structure
Properties
Signing
Flavors
Build Types
Dependencies
SDK Location
Project
Modules
app
Scope
{dir=libs, include=[*.jar]}
Compile
com.android.support:appcompat-v7:21.0.3
Compile

```
package com.hfad.bitsandpizzas;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

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

Make sure your activity extends Activity, not ActionBarActivity. If you use ActionBarActivity, you can't use the Holo or Material themes, Android forces you to use an AppCompat theme.




```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hfad.bitsandpizzas" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            ...
        </activity>
    </application>
</manifest>
```

The app's icon. Android Studio provides one by default.

User-friendly name of the app

The theme

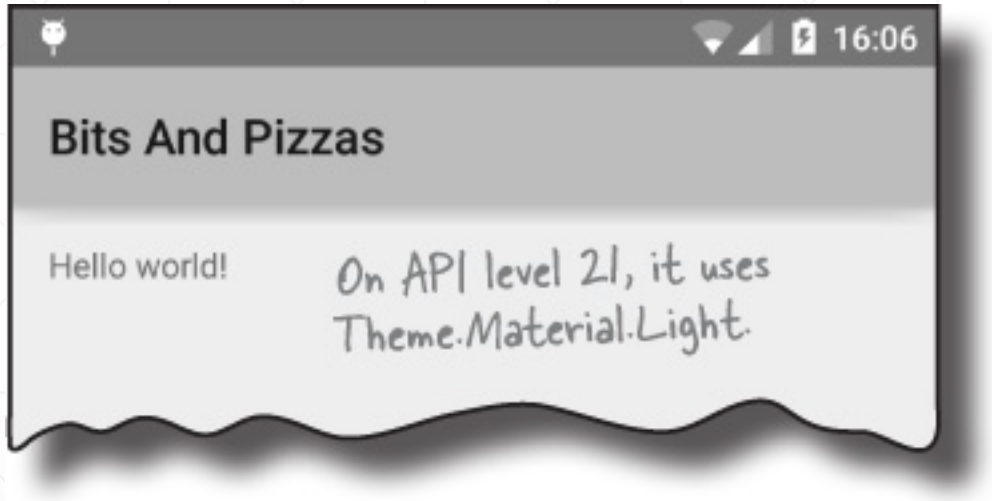
User-friendly name of the activity

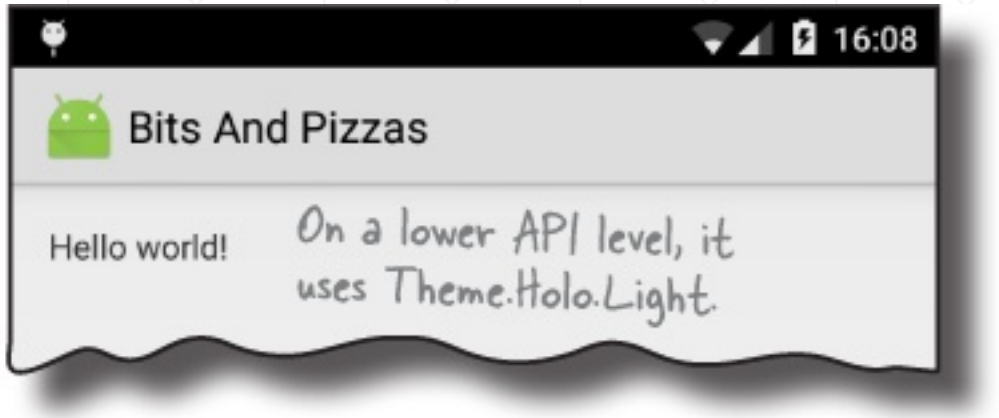
```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>
</resources>
```

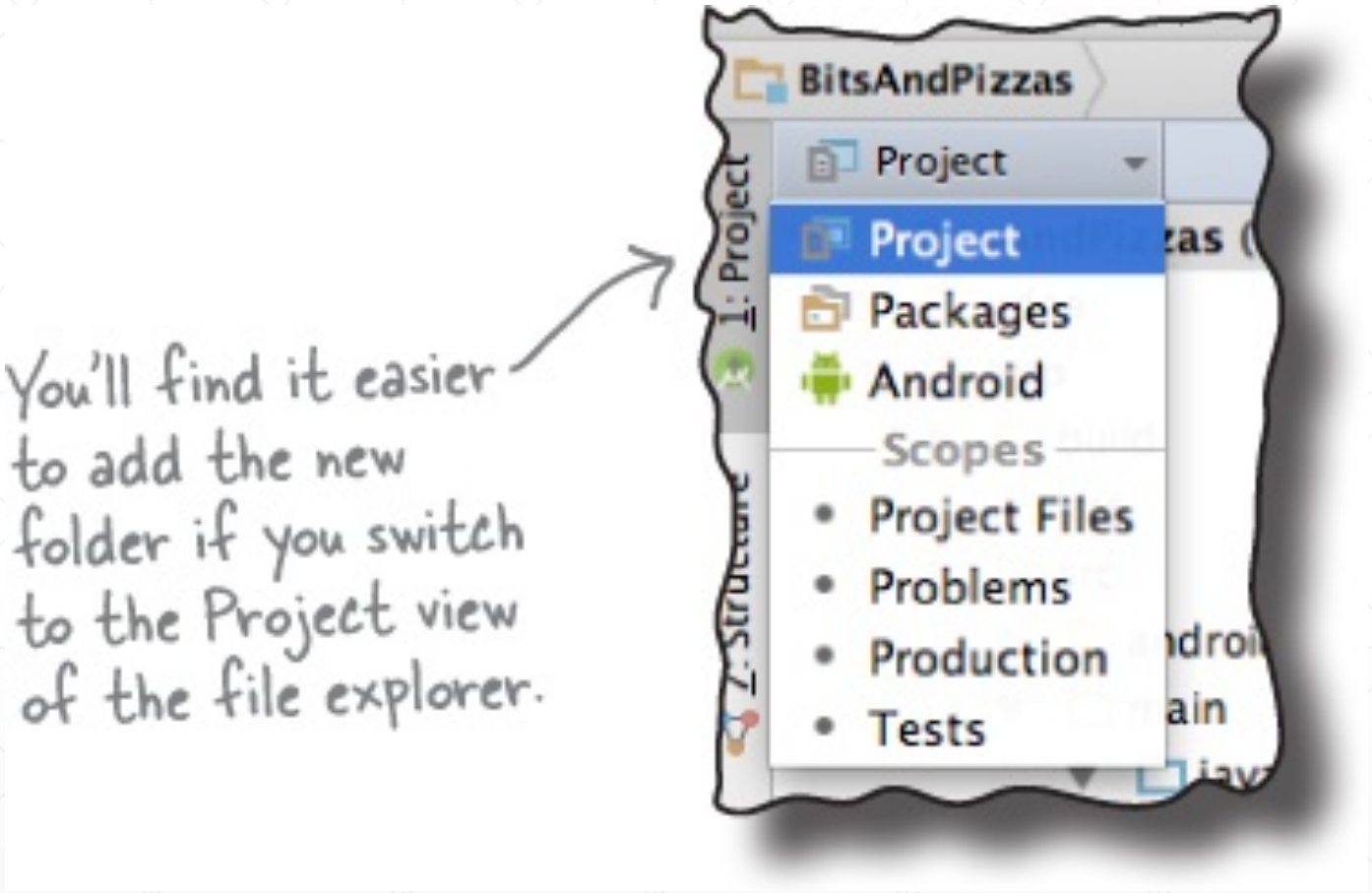
Don't worry if Android Studio has used a different theme—we'll change it on the next page.

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="android:background">#FF0000</item>
    </style>
</resources>
```

This line turns the backgrounds of your activities red.

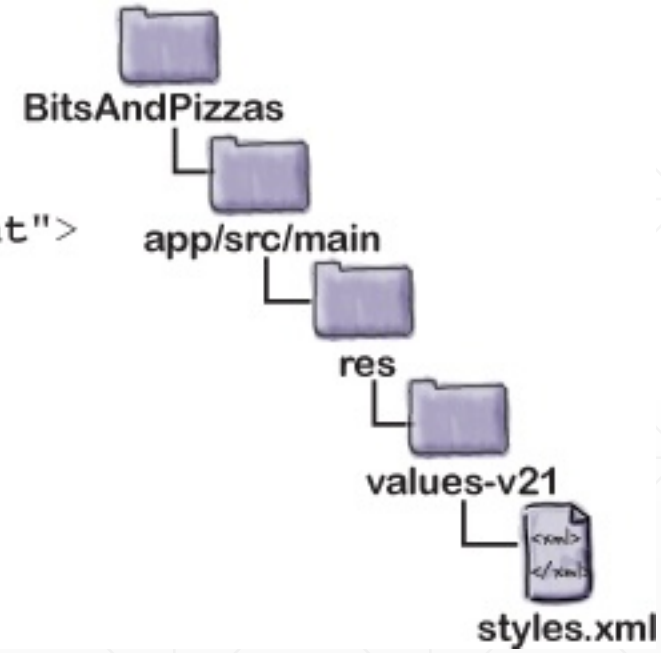

16:06
Bits And Pizzas
Hello world!
On API level 21, it uses Theme.Material.Light.

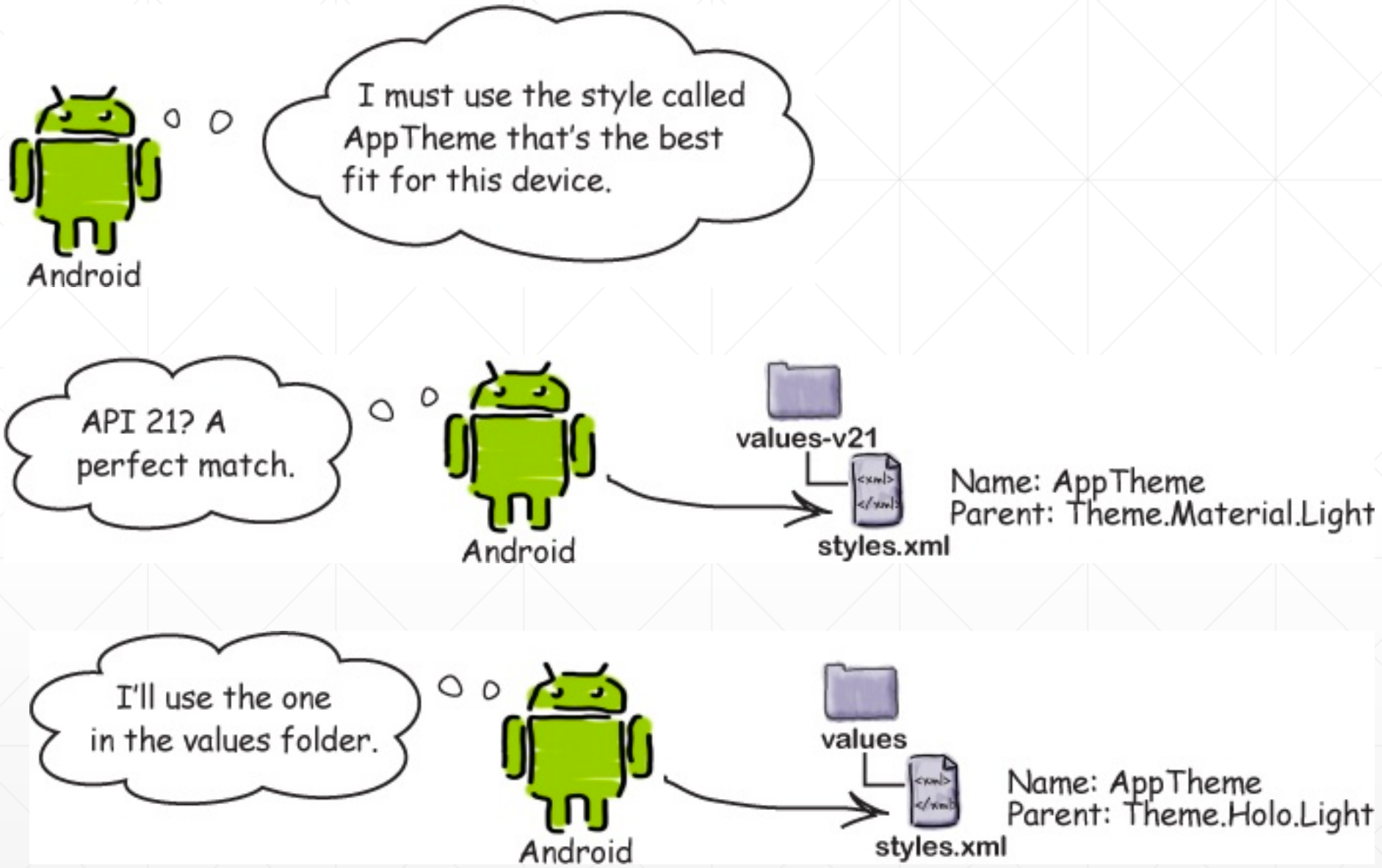

16:08
Bits And Pizzas
Hello world!
On a lower API level, it uses Theme.Holo.Light.

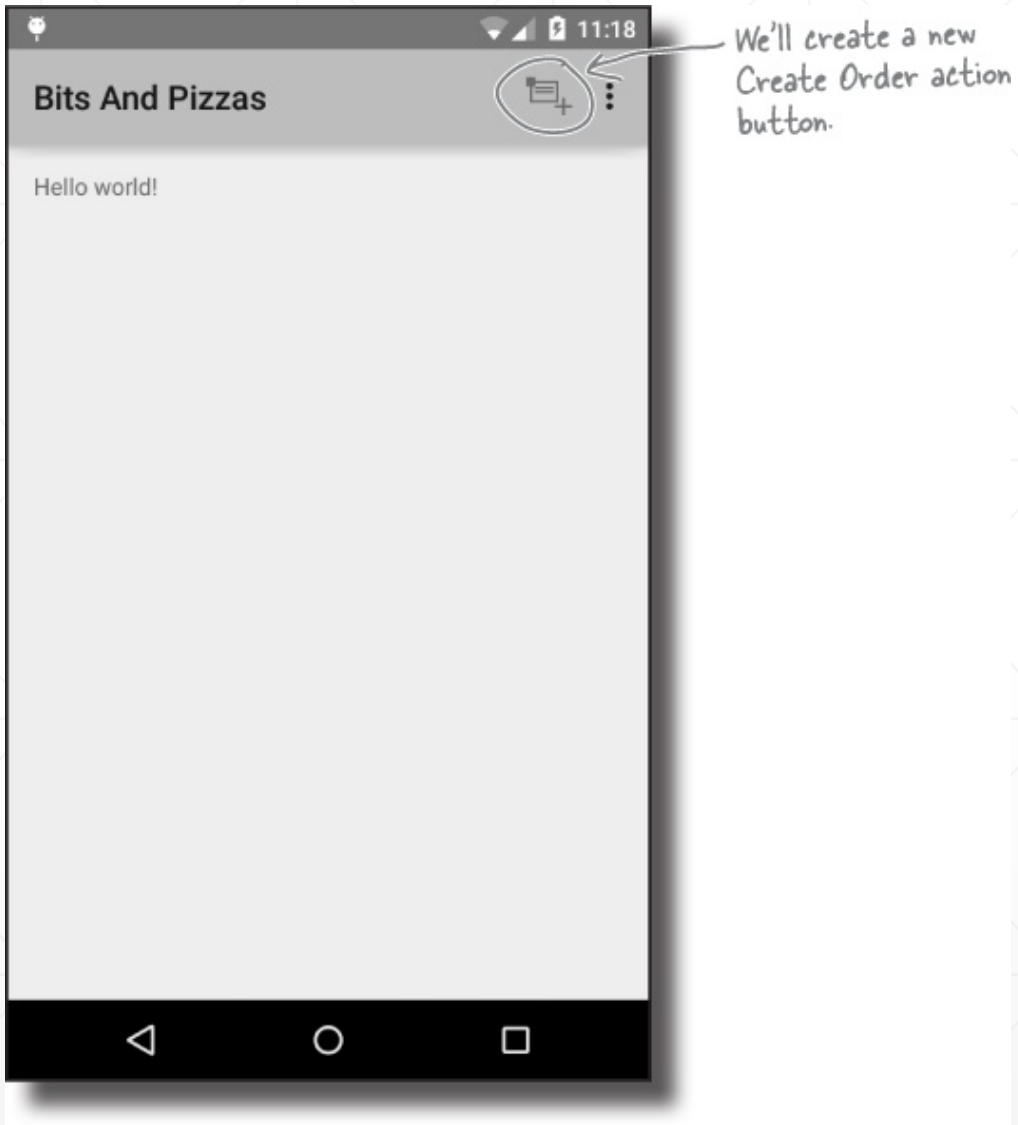

You'll find it easier to add the new folder if you switch to the Project view of the file explorer.
BitsAndPizzas
Project
1: Project
Project
Packages
Android
Scopes
Project Files
Problems
Production
Tests
7: Structure

```
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light">
        <!-- Customize your theme here. -->
    </style>
</resources>
```

We'll use this theme if the device is running API level 21.

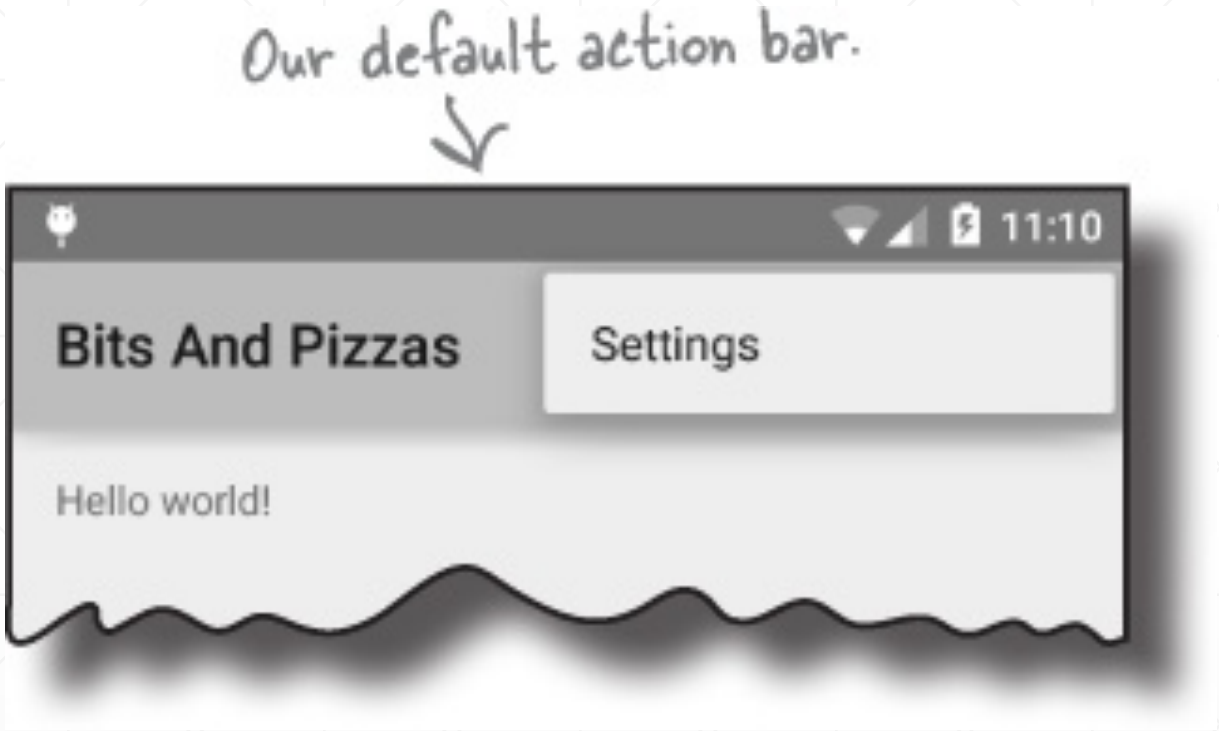



I must use the style called AppTheme that's the best fit for this device.
Android
API 21? A perfect match.
Android
values-v21
styles.xml
Name: AppTheme
Parent: Theme.Material.Light
I'll use the one in the values folder.
Android
values
styles.xml
Name: AppTheme
Parent: Theme.Holo.Light


11:18
Bits And Pizzas
Hello world!
We'll create a new Create Order action button.

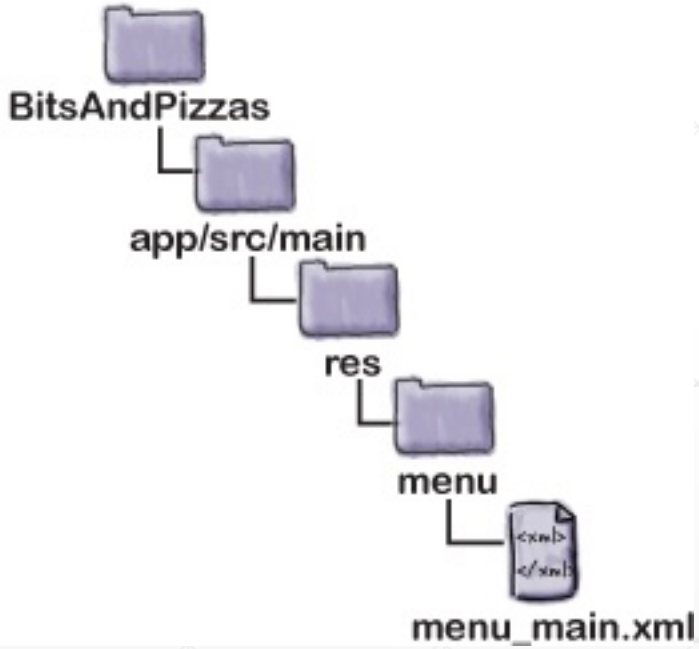
Adding action items to action bar

- Define the action items in a menu resource file.
- Get the activity to inflate the menu resource.
  - You do this by implementing the onCreateOptionsMenu() method.
- Add code to say what each item should do when clicked.
  - You do this by implementing the onOptionsItemSelected() method.


Our default action bar.
11:10
Bits And Pizzas
Settings
Hello world!

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MainActivity">

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />

</menu>
```

This is the Settings action item.



showAsAction	Description
"ifRoom"	Place the item in the action bar if there's space. If there's not space, put it in the overflow.
"withText"	Include the item's title text.
"never"	Put the item in the overflow area, and never in the main action bar.
"always"	Always place the item in the main area of the action bar. This value should be used sparingly; if you apply this to many items, they may overlap each other.

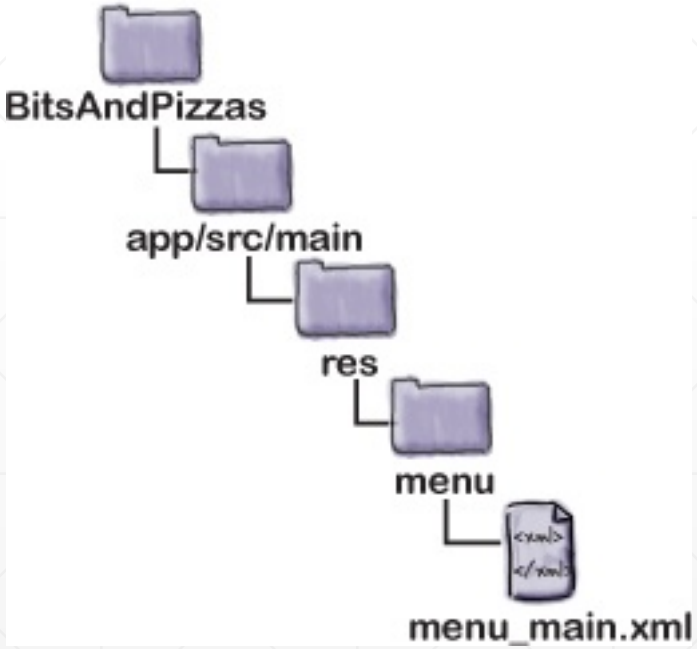
<item>	Description
android:id	Gives the item a unique ID. You need this in order to refer to the item in your activity code.
android:icon	The item's icon. This is a drawable or mipmap resource.
android:title	The item's text. This may not get displayed if your item has an icon if there's not space in the action bar for both. If the item appears in the action bar's overflow, only the text will be displayed.
android:orderInCategory	An integer value that helps Android decide the order in which items should appear in the action bar.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    ...>

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>
```

This adds the app namespace.

The ID, title, and orderInCategory attributes use the android namespace.

showAsAction uses the app namespace.



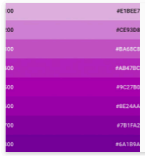

11:18
zzas
The new action item

HOME
DESIGN
DEVELOP
DISTRIBUTE

Design

Downloads

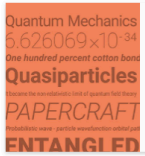
For icons, sticker sheets, and other downloadable resources, visit the Design site or use the links below.



MATERIAL DESIGN
Color Palettes

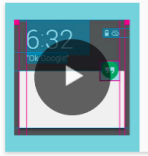
MATERIAL DESIGN
Layout Templates

MATERIAL DESIGN
Sticker Sheets & Icons



MATERIAL DESIGN
Typography: Roboto and Noto Sans fonts

MATERIAL DESIGN
Material icon collection

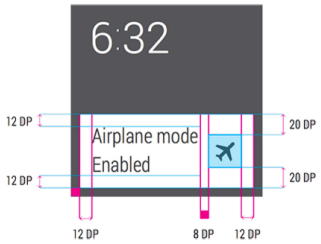


DESIGN
Android Wear Materials

Android Wear Materials

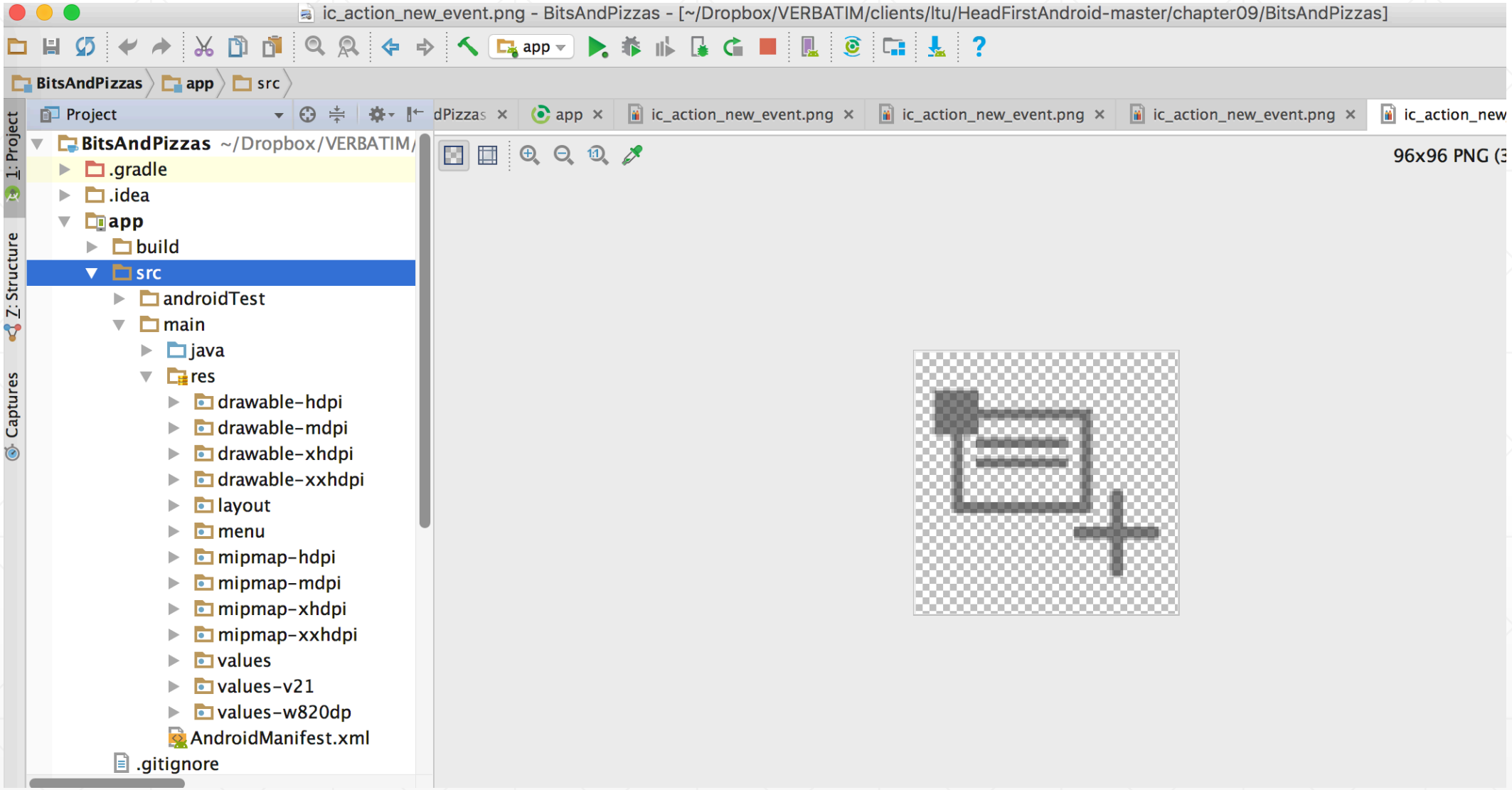
UI toolkit

The toolkit contains detailed specs and measurements of all of the primary Android Wear UI components. Available in PDF and Illustrator formats.



Adobe® Illustrator® Toolkit

PDF Toolkit

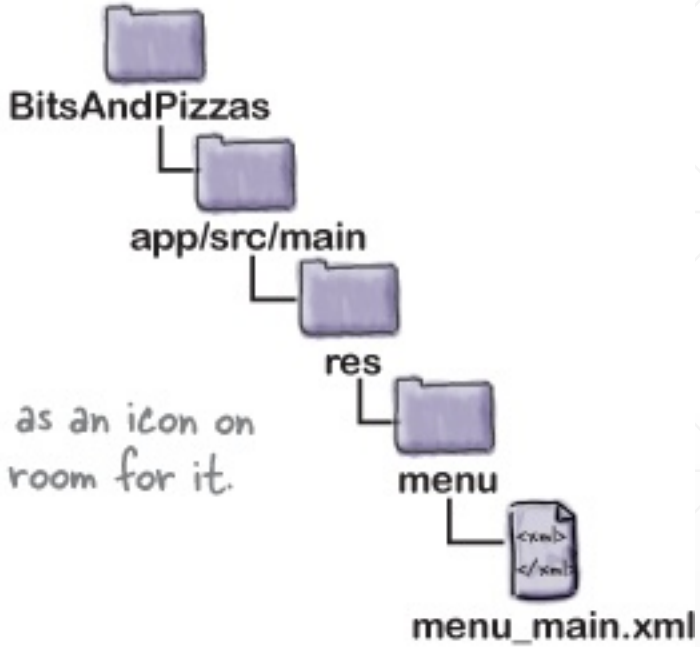

ic_action_new_event.png - BitsAndPizzas - [~/Dropbox/VERBATIM/clients/ltu/HeadFirstAndroid-master/chapter09/BitsAndPizzas]
app
BitsAndPizzas
app
src
Project
1: Project
7: Structure
Captures
BitsAndPizzas ~/Dropbox/VERBATIM/
.gradle
.idea
app
build
src
androidTest
main
java
res
drawable-hdpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
menu
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
values
values-v21
values-w820dp
AndroidManifest.xml
.gitignore
app
ic_action_new_event.png
ic_action_new_event.png
ic_action_new_event.png
ic_action_new
96x96 PNG

```
<string name="action_create_order">Create Order</string>
```

We'll use this for the action item's title.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MainActivity">

    <item android:id="@+id/action_create_order"
        android:title="@string/action_create_order"
        android:icon="@drawable/ic_action_new_event"
        android:orderInCategory="1"
        app:showAsAction="ifRoom" />

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>
```

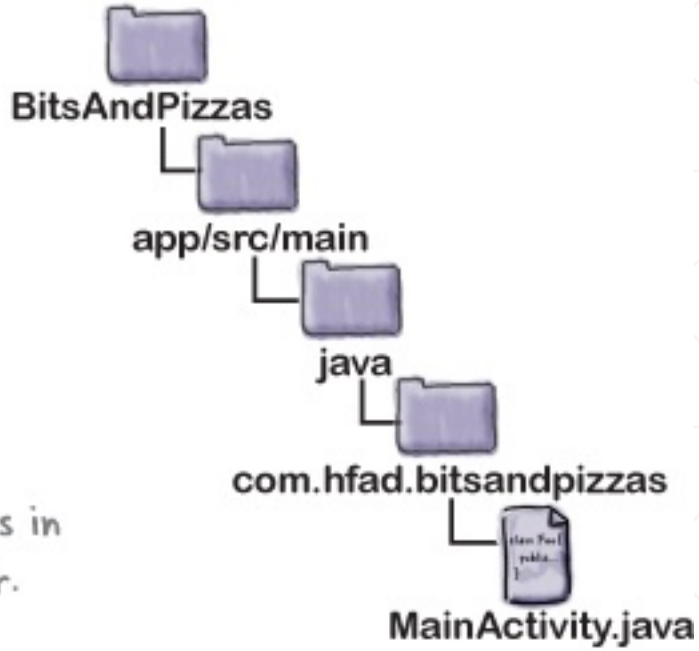
The new item is displayed as an icon on the action bar if there's room for it.



```
package com.hfad.bitsandpizzas;

import android.view.Menu;
...

public class MainActivity extends Activity {

    ...

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return super.onCreateOptionsMenu(menu);
    }
}
```

The onCreateOptionsMenu() method uses the Menu class.

Implementing this method adds any items in the menu resource file to the action bar.



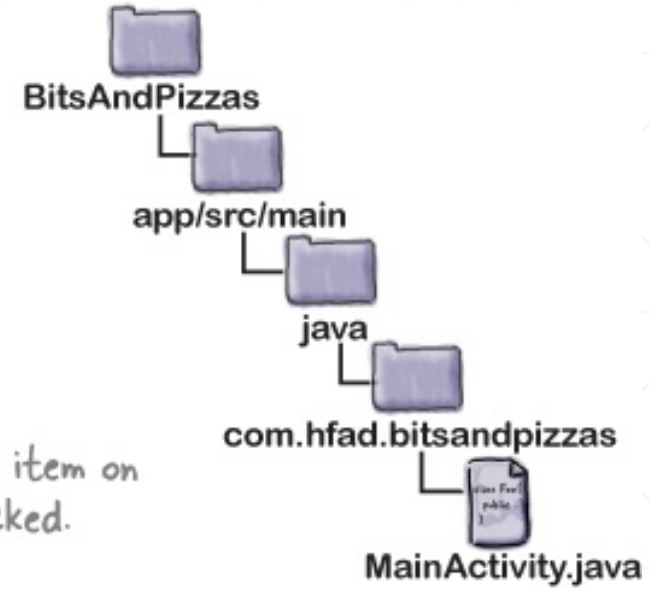
```
getMenuInflater().inflate(R.menu.menu_main, menu);
```

This is a Menu object that represents the action bar.

This is the menu resource file.

```
package com.hfad.bitsandpizzas;

import android.view.MenuItem;
...

public class MainActivity extends Activity {

    ...

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_create_order:
                //Code to run when the Create Order item is clicked
                return true;
            case R.id.action_settings:
                //Code to run when the settings item is clicked
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

The onOptionsItemSelected() method uses this class.



The MenuItem object is the item on the action bar that was clicked.

Check which item was clicked.

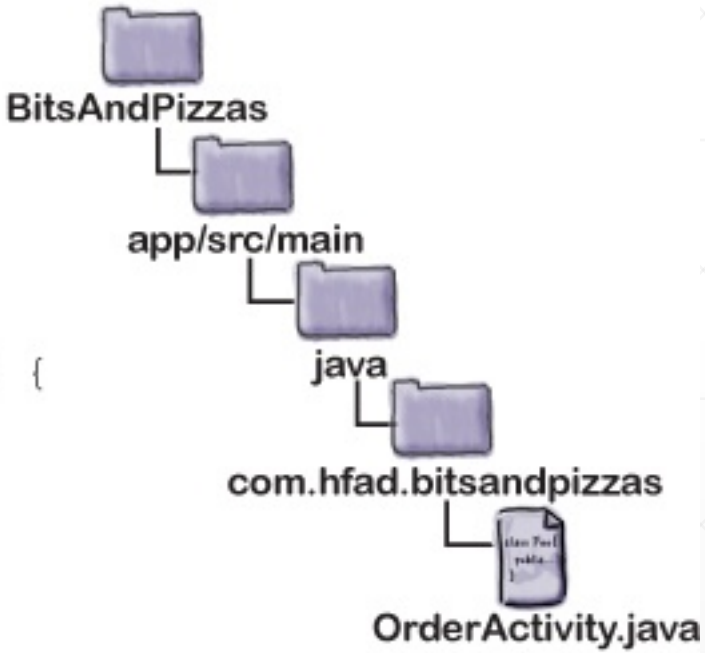
We need to get the Create Order item to do something.

Android Studio created a Settings item for us. You'd put code to get it to do something here.

Returning true tells Android you've dealt with the item being clicked.

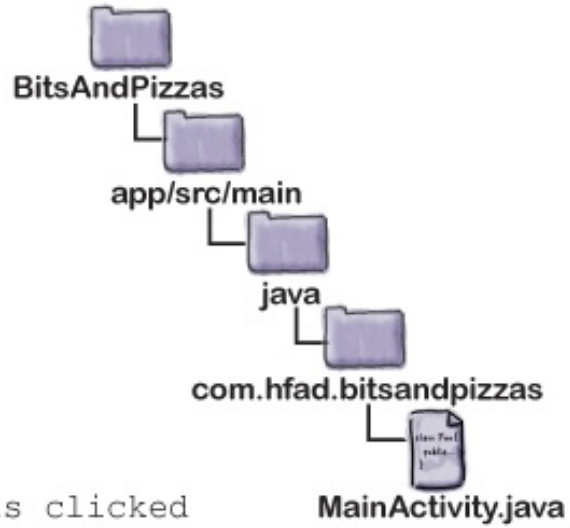
```
package com.hfad.bitsandpizzas;

import android.app.Activity;
import android.os.Bundle;

public class OrderActivity extends Activity {

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

Make sure that OrderActivity extends Activity, not ActionBarActivity.

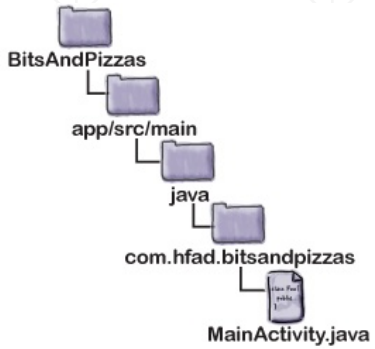


```
package com.hfad.bitsandpizzas;

import android.content.Intent;
...

public class MainActivity extends Activity {
...
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_create_order:
                //Code to run when the Create Order item is clicked
                Intent intent = new Intent(this, OrderActivity.class);
                startActivity(intent);
                return true;
            case R.id.action_settings:
                //Code to run when the settings item is clicked
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

We need to use the Intent class.



This intent is used to start OrderActivity when the Create Order action item is clicked.

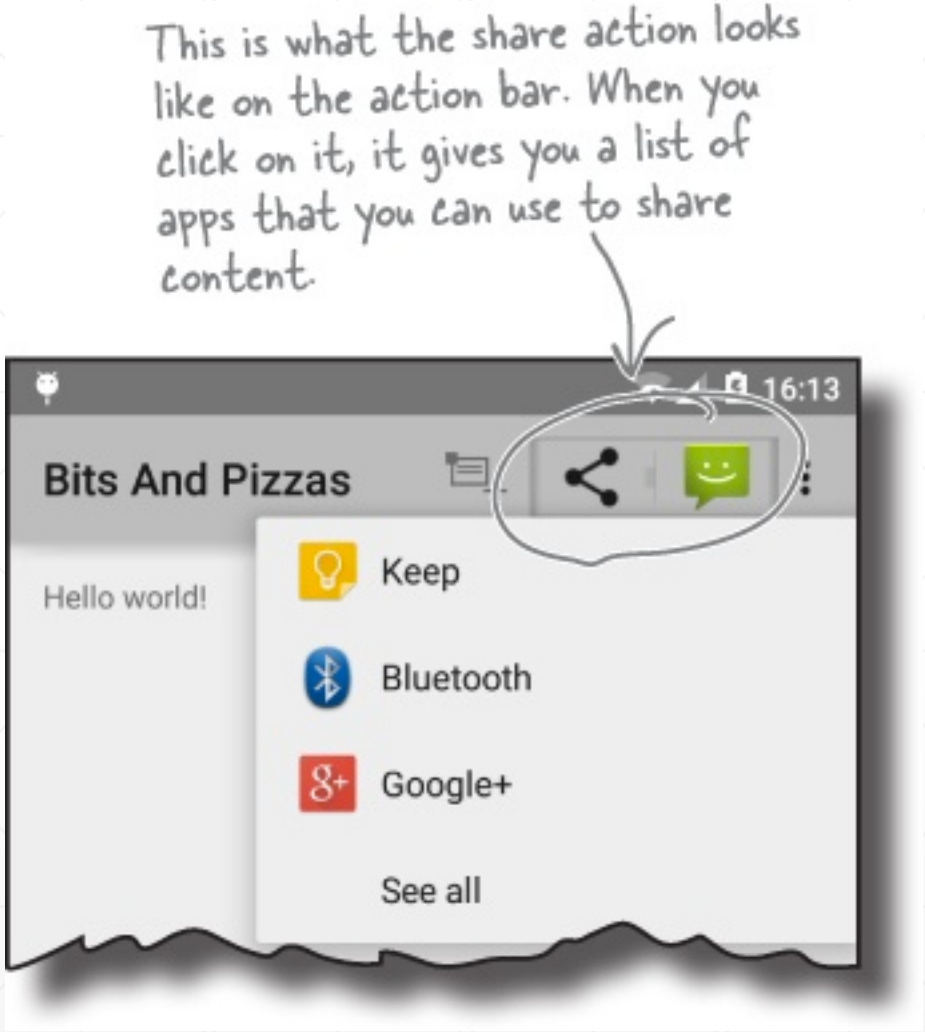


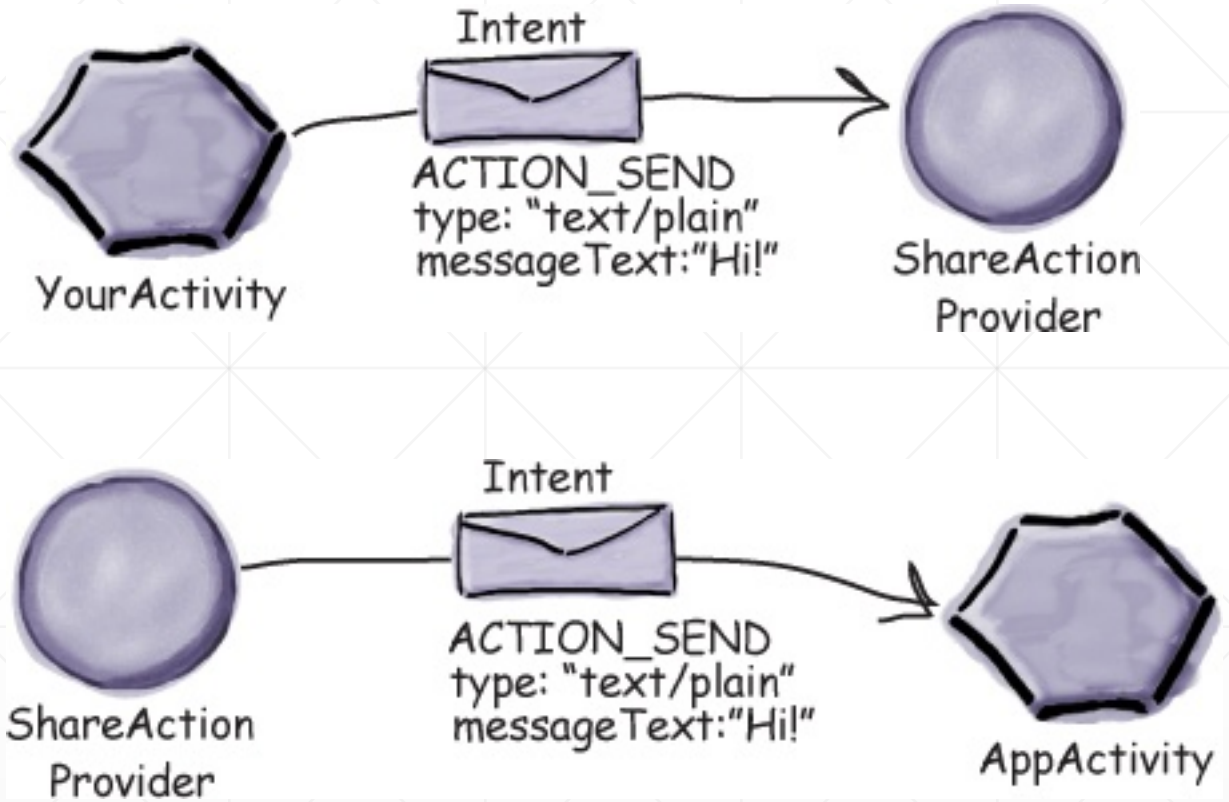

```
package com.hfad.bitsandpizzas;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_create_order:
                //Code to run when the Create Order item is clicked
                Intent intent = new Intent(this, OrderActivity.class);
                startActivity(intent);
                return true;
            case R.id.action_settings:
                //Code to run when the settings item is clicked
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Add items to the action bar.

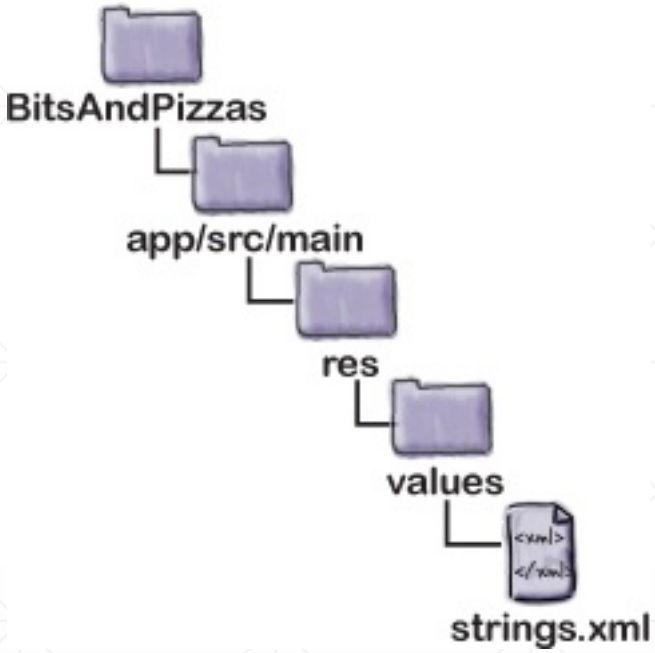
Start OrderActivity when the Create Order item is clicked.


11:18
Bits And Pizzas
Hello world!
Here's the Create Order action item.
11:23
Order Activity
Hello world!
Clicking on the Create Order action item starts OrderActivity.

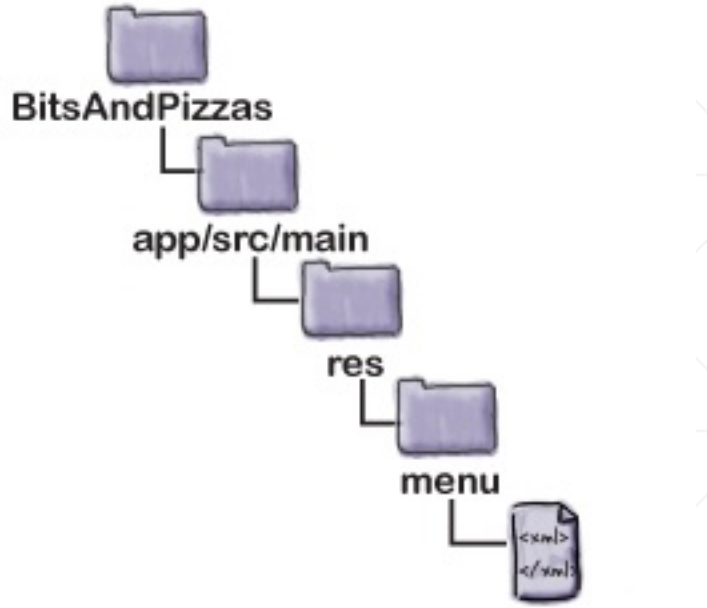

This is what the share action looks like on the action bar. When you click on it, it gives you a list of apps that you can use to share content.
16:13
Bits And Pizzas
Hello world!
Keep
Bluetooth
Google+
See all


Intent
ACTION_SEND
type: "text/plain"
messageText:"Hi!"
YourActivity
ShareAction
Provider
Intent
ACTION_SEND
type: "text/plain"
messageText:"Hi!"
ShareAction
Provider
AppActivity

```
<string name="action_share">Share</string>
```

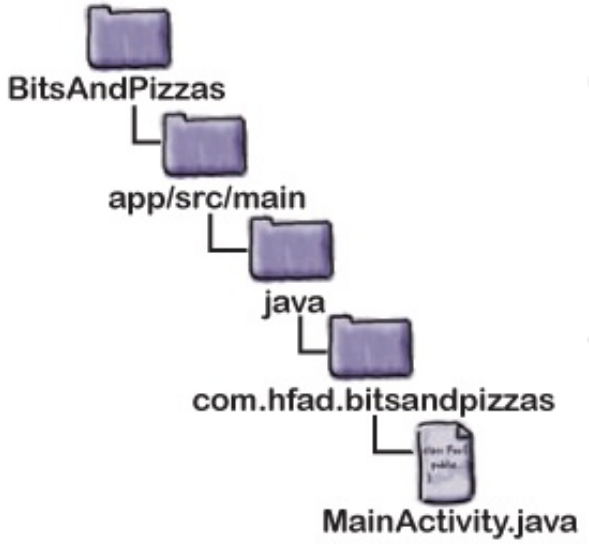


```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">

    <item android:id="@+id/action_create_order"
        ... />

    <item android:id="@+id/action_share"
        android:title="@string/action_share"
        android:orderInCategory="2"
        app:showAsAction="ifRoom"
        android:actionProviderClass="android.widget.ShareActionProvider" />

    <item android:id="@+id/action_settings"
        ... />
</menu>
```



Display the share action provider in the action bar if there's room.

This is the share action provider class.




```
package com.hfad.bitsandpizzas;

...
import android.widget.ShareActionProvider;

public class MainActivity extends Activity {

    private ShareActionProvider shareActionProvider;

    ...
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        MenuItem menuItem = menu.findItem(R.id.action_share);
        shareActionProvider = (ShareActionProvider) menuItem.getActionProvider();
        setIntent("This is example text");
        return super.onCreateOptionsMenu(menu);
    }

    private void setIntent(String text) {
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_TEXT, text);
        shareActionProvider.setShareIntent(intent);
    }
}
```

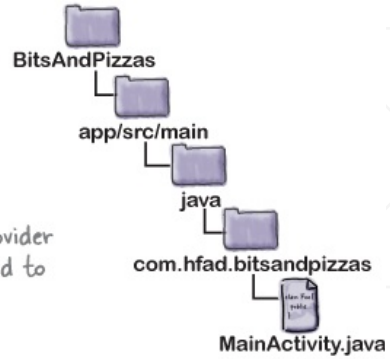
Add a ShareActionProvider private variable.

Get a reference to the share action provider and assign it to the private variable. Then call the setIntent() method.

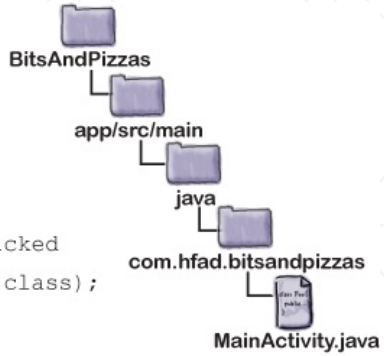
We created the setIntent() method. It creates an intent, and passes it to the share action provider using its setShareIntent() method.

```
package com.hfad.bitsandpizzas;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ShareActionProvider;

public class MainActivity extends Activity {

    private ShareActionProvider shareActionProvider;

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        MenuItem menuItem = menu.findItem(R.id.action_share);
        shareActionProvider = (ShareActionProvider) menuItem.getActionProvider();
        setIntent("This is example text");
        return super.onCreateOptionsMenu(menu);
    }

    private void setIntent(String text) {
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_TEXT, text);
        shareActionProvider.setShareIntent(intent);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_create_order:
                //Code to run when the Create Order item is clicked
                Intent intent = new Intent(this, OrderActivity.class);
                startActivity(intent);
                return true;
            case R.id.action_settings:
                //Code to run when the settings item is clicked
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

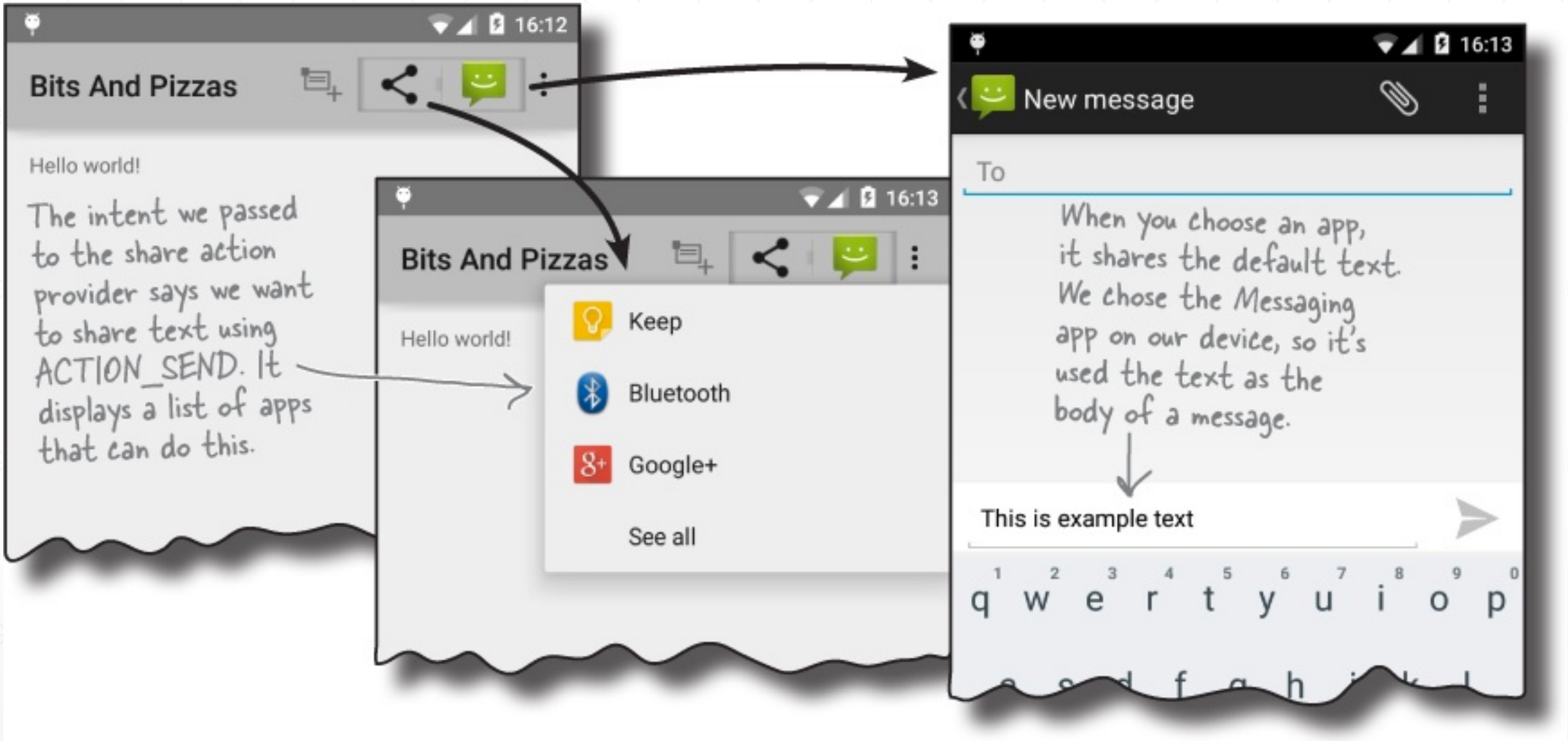
We're using the ShareActionProvider class, so we need to import it.



This sets the default text that the share action provider should share.

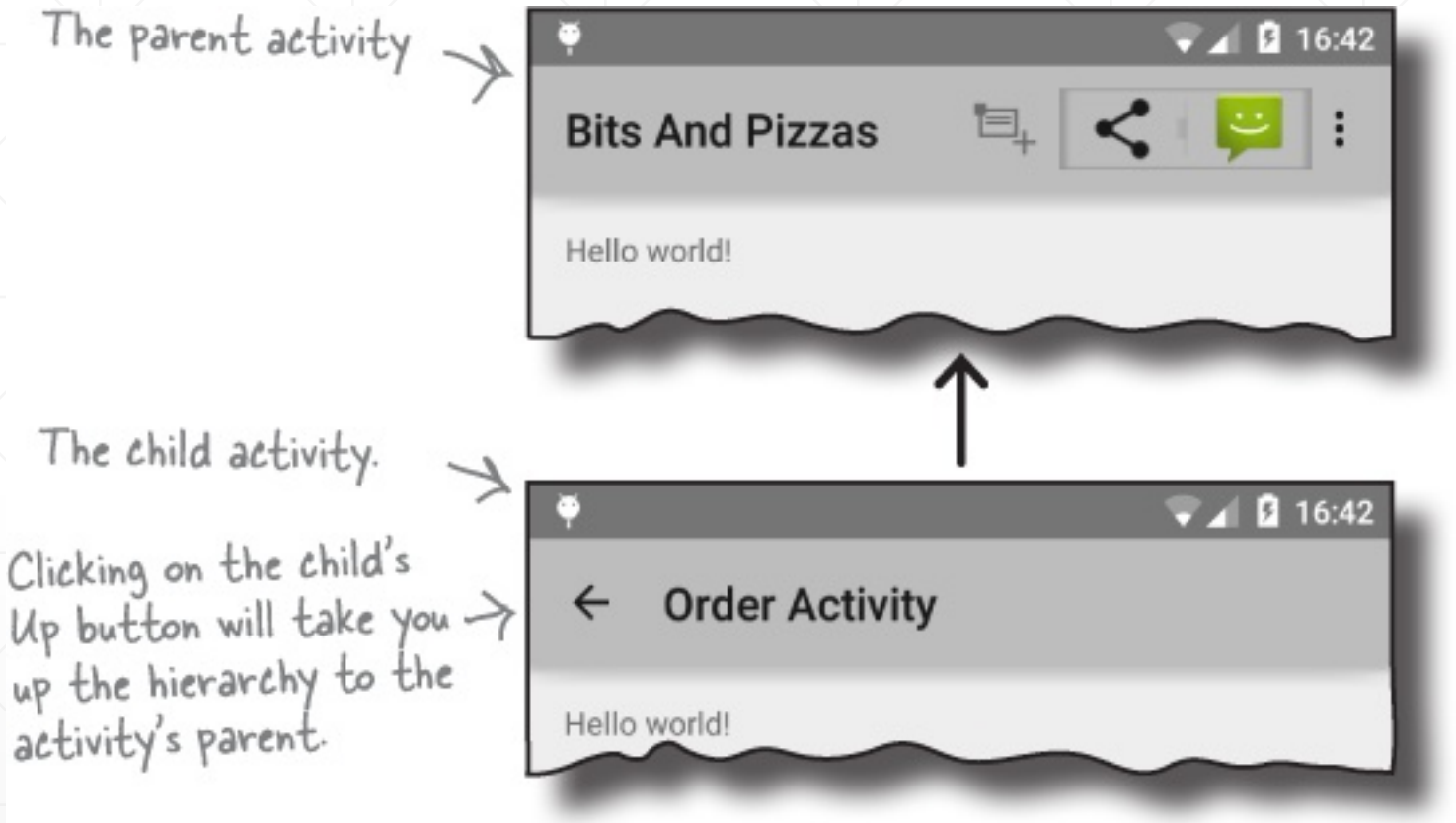


This method hasn't changed.


16:12
Bits And Pizzas
Hello world!
The intent we passed to the share action provider says we want to share text using ACTION_SEND. It displays a list of apps that can do this.
16:13
Bits And Pizzas
Hello world!
Keep
Bluetooth
Google+
See all
16:13
New message
To
When you choose an app, it shares the default text. We chose the Messaging app on our device, so it's used the text as the body of a message.
This is example text


This is the Up button.
16:42
Order Activity


16:42
Bits And Pizzas
Hello world!
Click on the Create Order action to go to OrderActivity.
16:42
Order Activity
Hello world!
Click on the Up button...
16:42
Bits And Pizzas
Hello world!
...to go to MainActivity.


The parent activity
16:42
Bits And Pizzas
Hello world!
The child activity.
Clicking on the child's Up button will take you up the hierarchy to the activity's parent.
16:42
Order Activity
Hello world!

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hfad.bitsandpizzas" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            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=".OrderActivity"
            android:label="@string/title_activity_order"
            android:parentActivityName=".MainActivity">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity" />
        </activity>

    </application>
</manifest>
```

Apps at API level 16 or above use this line. It says that OrderActivity's parent is MainActivity.

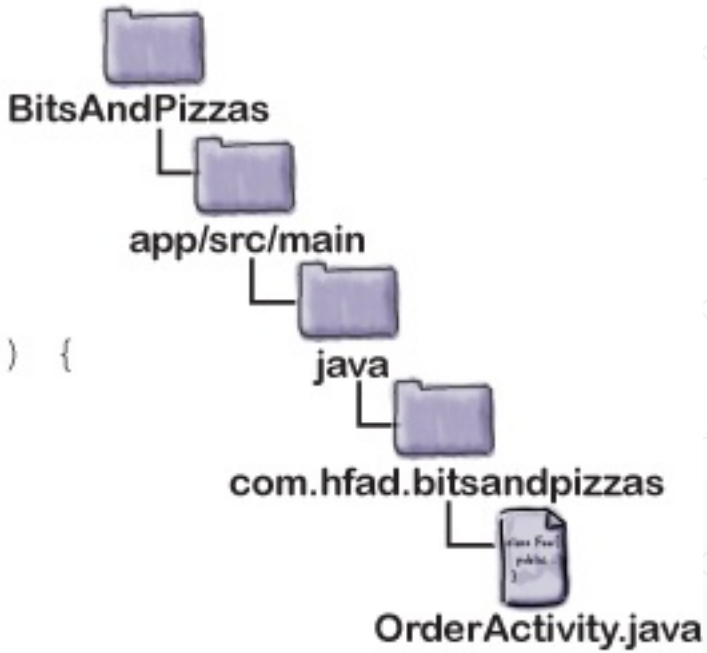
You only need to add the <meta-data> element if you're supporting apps below API level 16. We've only included it so you can see what it looks like, and including it doesn't do any harm.

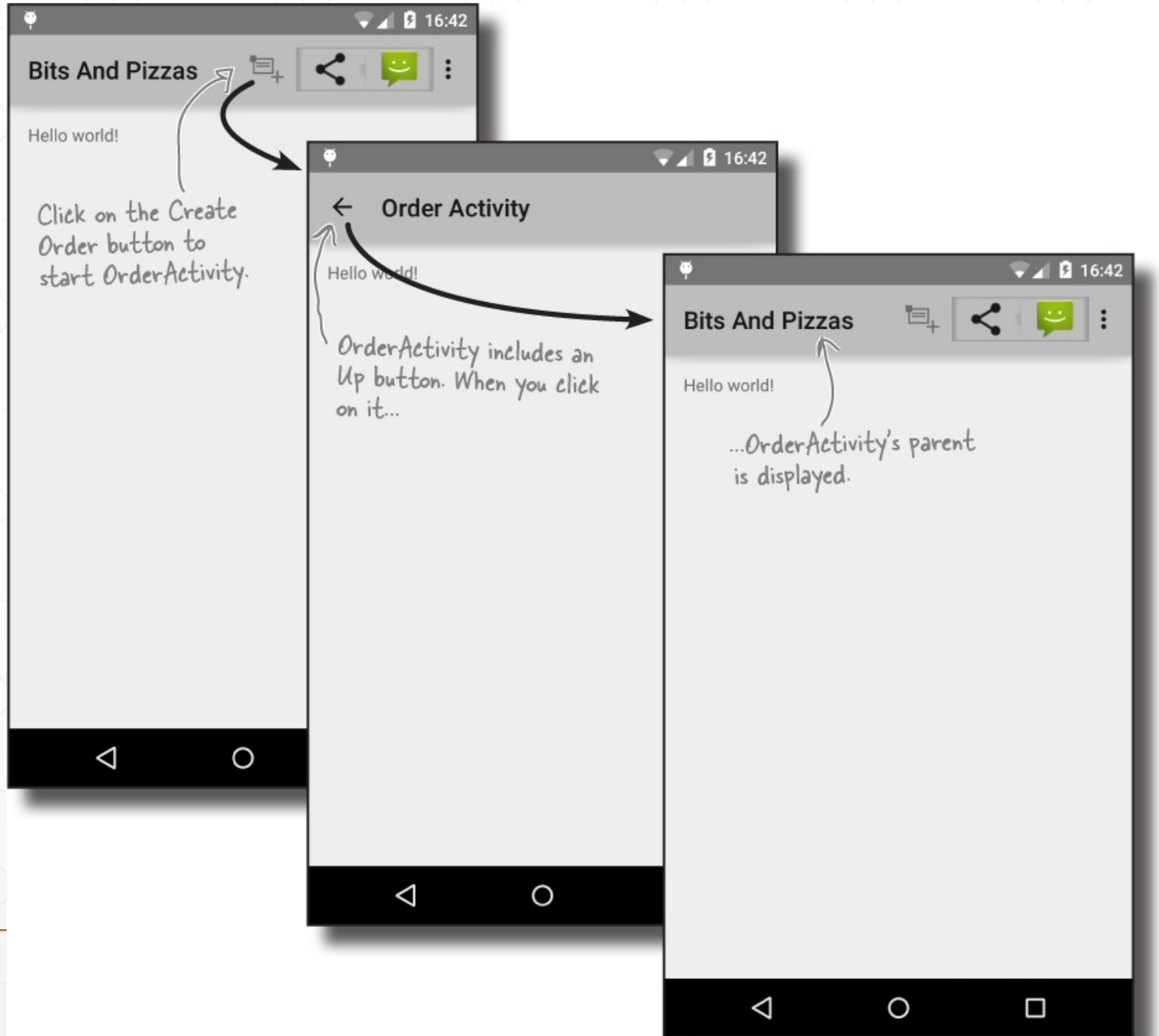
```
ActionBar actionBar = getActionBar();
actionBar.setDisplayHomeAsUpEnabled(true);
```

```
package com.hfad.bitsandpizzas;

import android.app.ActionBar;
import android.app.Activity;
import android.os.Bundle;

public class OrderActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order);
        ActionBar actionBar = getActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
    }
}
```

We're using the ActionBar class, so we need to import it.

This enables the Up button in the action bar.




16:42
Bits And Pizzas
Hello world!
Click on the Create Order button to start OrderActivity.
16:42
Order Activity
Hello world!
OrderActivity includes an Up button. When you click on it...
16:42
Bits And Pizzas
Hello world!
...OrderActivity's parent is displayed.

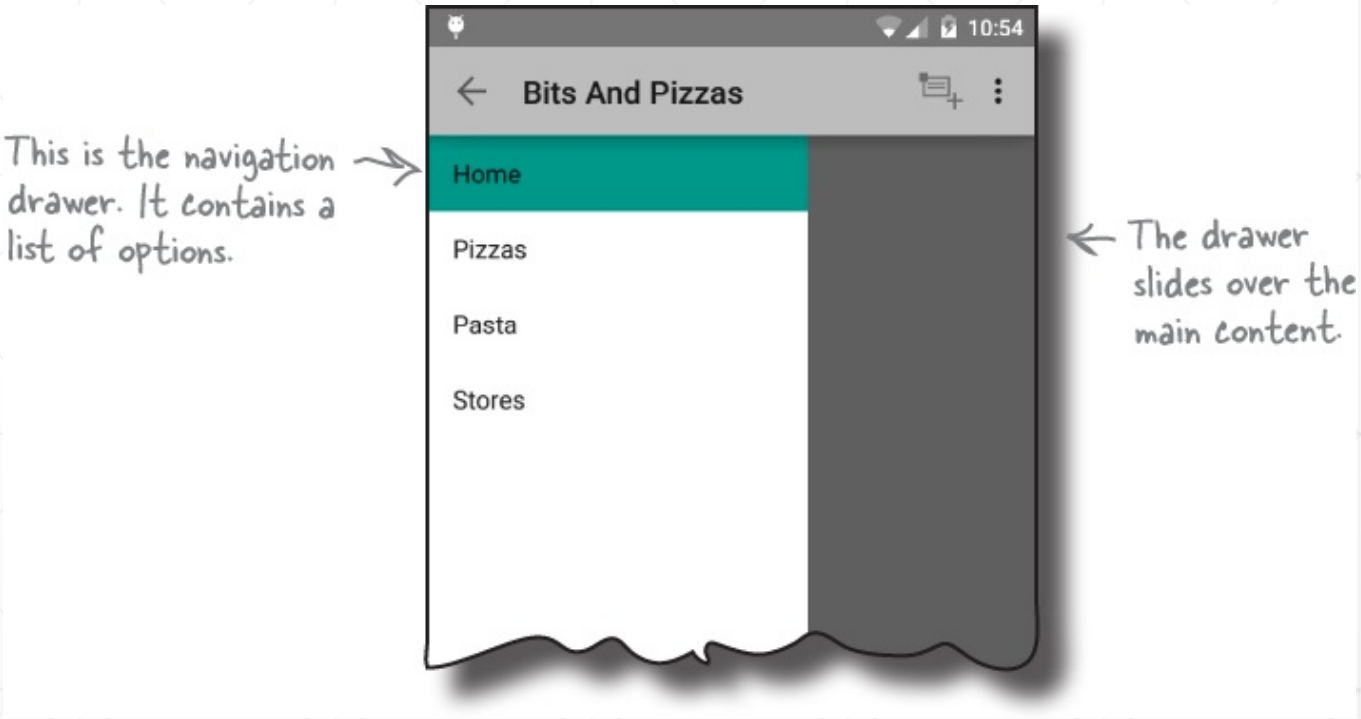
Bullet Points

- To add an action bar to apps supporting API level 11 or above apply one of the Holo or Material themes.
- Add an action bar to apps supporting API level 7 or above by applying an AppCompat theme and using theActionBarActivity class. If you use ActionBarActivity, you must use an AppCompat theme.
- ActionBarActivity and the AppCompat themes are in the v7 appcompat support library.
- The android:theme attribute in *AndroidManifest.xml* specifies which theme to apply.
- You define styles in a style resource file using the <style> element. The name attribute gives the style a name. The parent attribute specifies where the style should inherit its properties from.
- The default folder for style resource files is *app/src/main/res/values*. Put a style resource file in the*app/src/main/res/values-v21* folder if you want it to be used on API level 21.
- Add action items to your action bar by adding items to a menu resource file.

Bullet Points

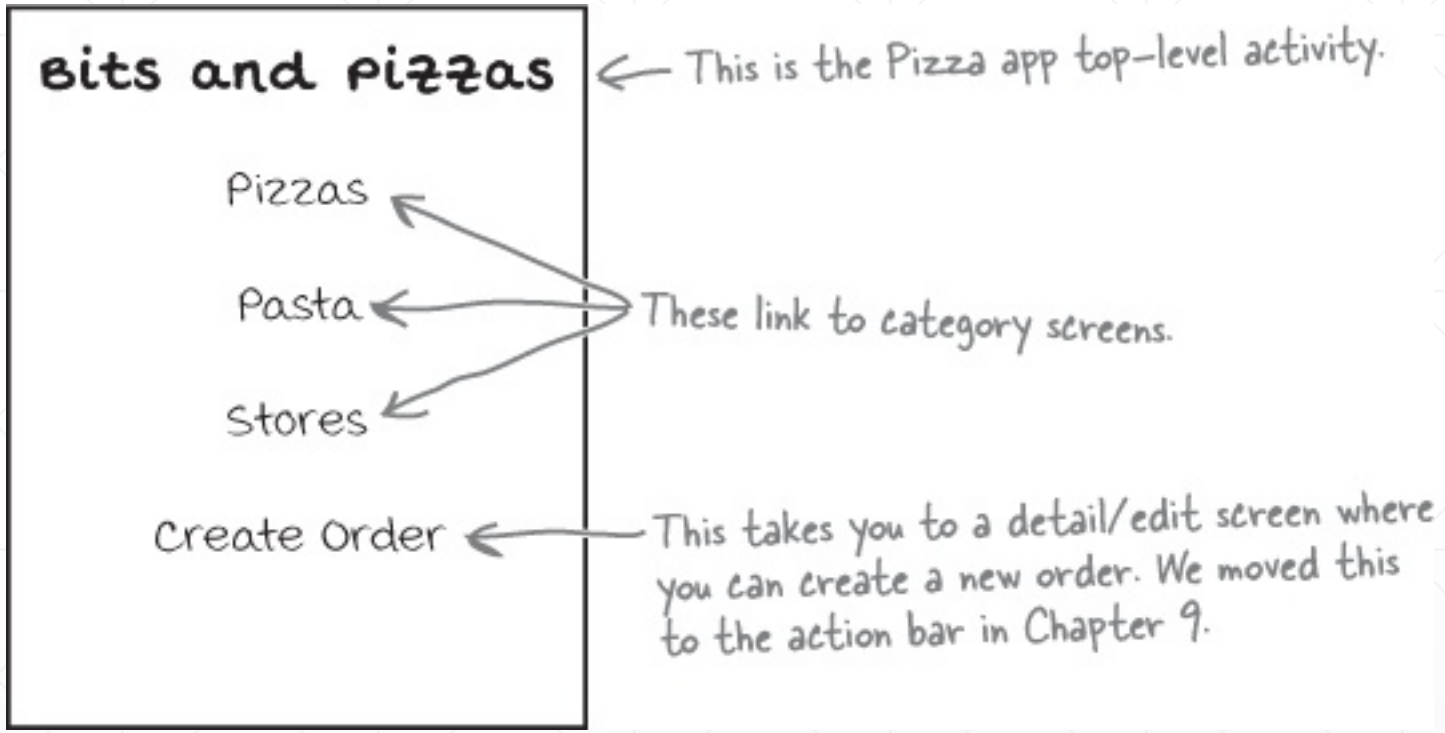
- Add the items in the menu resource file to the action bar by implementing the activity'sonCreateOptionsMenu() method.
- Say what items should do when clicked by implementing the activity's onOptionsItemSelected()method.
- You can share content by adding the share action provider to your action bar. Add it by including it in your menu resource file. Call its setShareIntent() method to pass it an intent describing the content you wish to share.
- Add an Up button to your action bar to navigate up the app's hierarchy. Specify the hierarchy in*AndroidManifest.xml*. Use the ActionBar setDisplayHomeAsUpEnabled() method to enable the Up button.

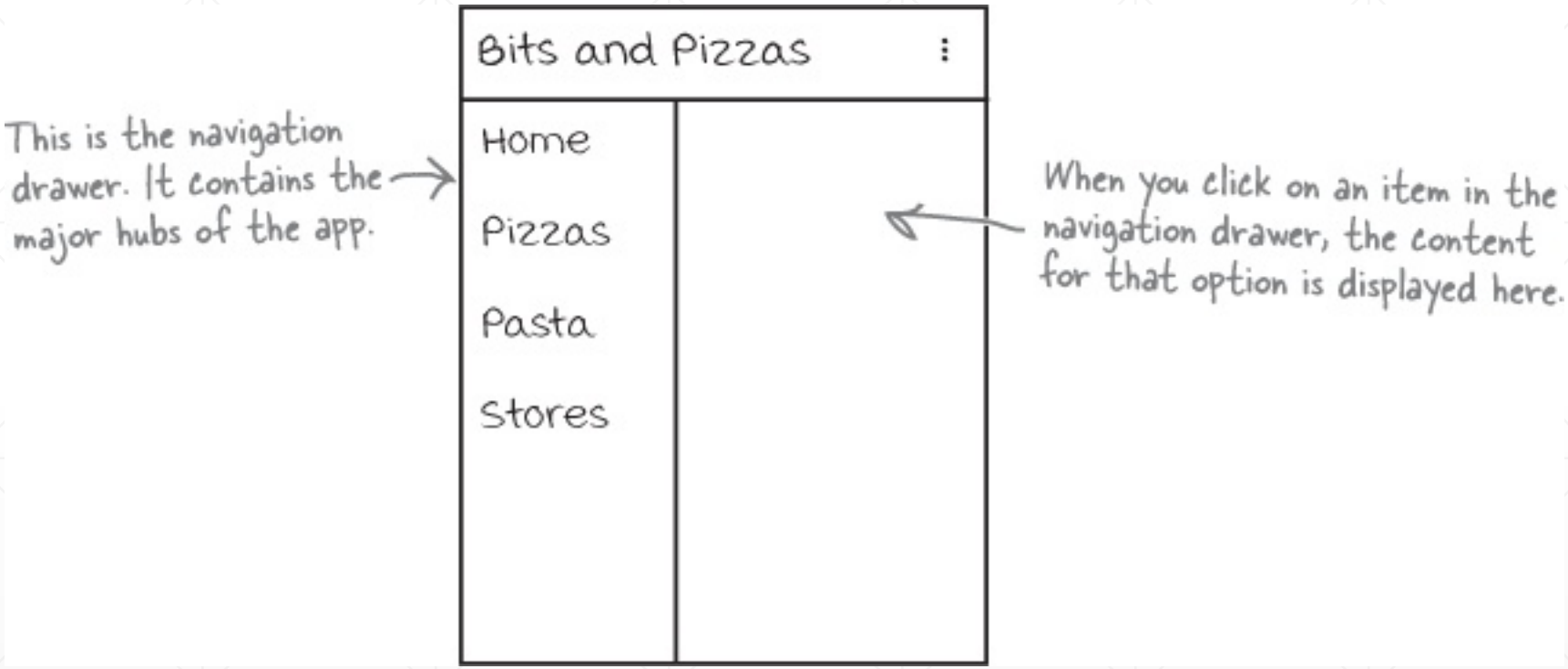
Navigation Drawers



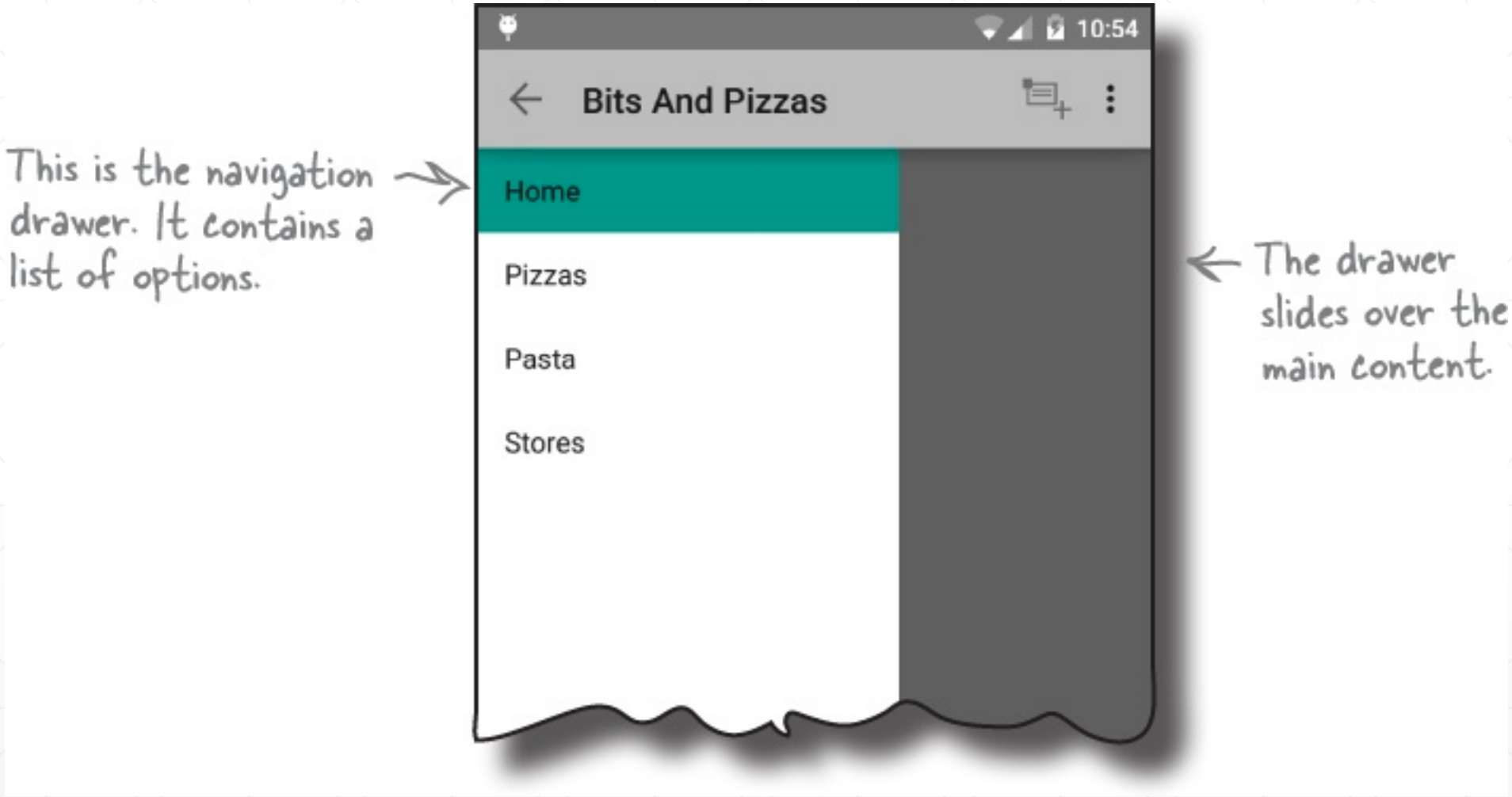
Bullet Points

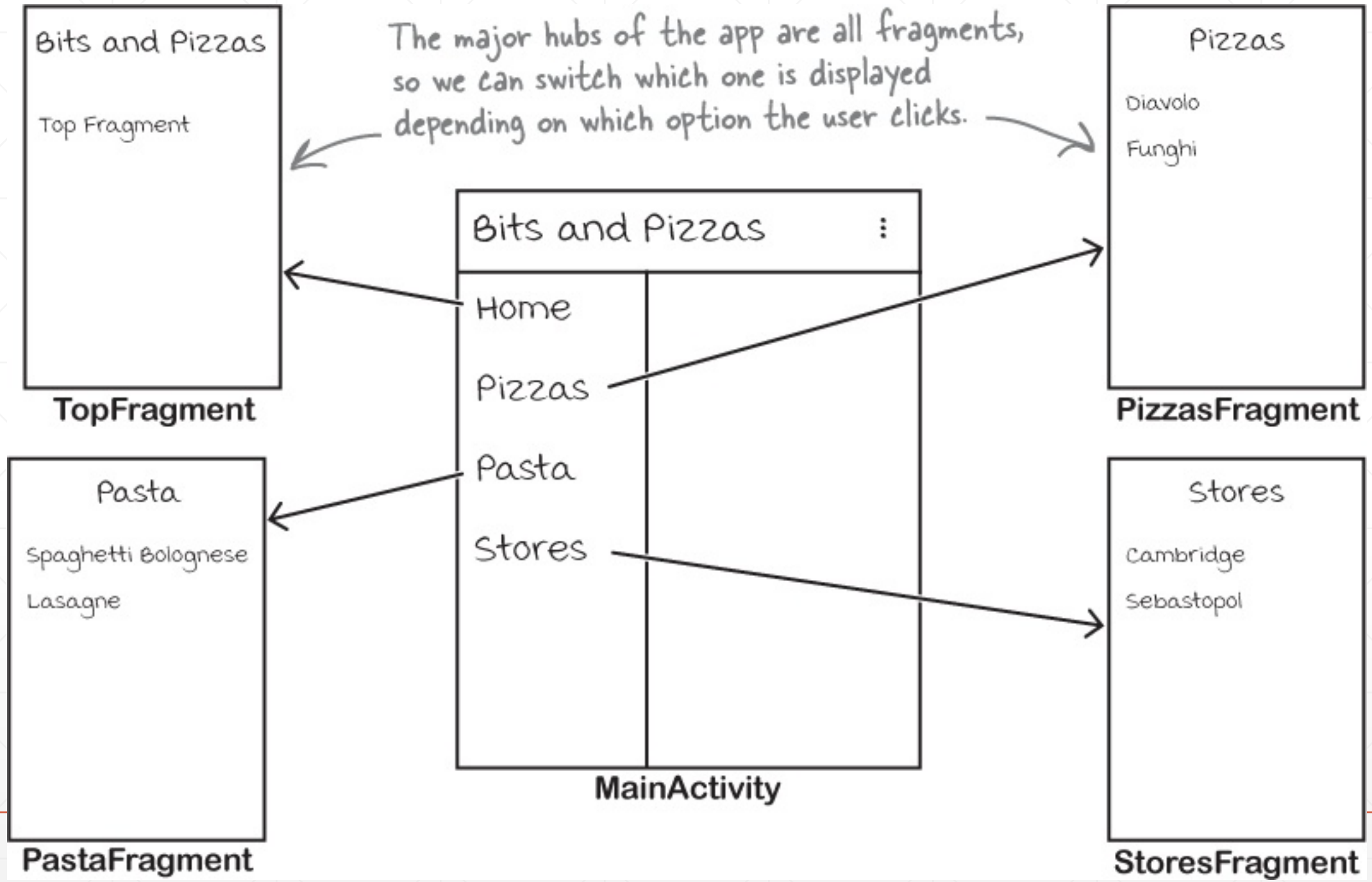
- Use a DrawerLayout to create an activity with a navigation drawer. Use the drawer to navigate to the major hubs of your app.
- If you're using an action bar, use ActionBarDrawerToggle as a DrawerListener. This allows you to respond to the drawer opening and closing, and adds an icon to the action bar for opening and closing the drawer.
- To change action bar items at runtime, call invalidateOptionsMenu() and add the changes in the activity's onPrepareOptionsMenu() method.
- React to changes on the back stack by implementing theFragmentManager.OnBackStackChangedListener().
- The fragment manager's findFragmentByTag() method searches for fragments with a given tag.


Bits and Pizzas
This is the Pizza app top-level activity.
Pizzas
Pasta
Stores
These link to category screens.
Create Order
This takes you to a detail/edit screen where you can create a new order. We moved this to the action bar in Chapter 9.


Bits and Pizzas
Home
Pizzas
Pasta
Stores
This is the navigation drawer. It contains the major hubs of the app.
When you click on an item in the navigation drawer, the content for that option is displayed here.

This is the navigation drawer icon. Click on it or swipe your finger to open the drawer.


This is the navigation drawer. It contains a list of options.
10:54
Bits And Pizzas
Home
Pizzas
Pasta
Stores
The drawer slides over the main content.


Bits and Pizzas
Top Fragment
TopFragment
The major hubs of the app are all fragments, so we can switch which one is displayed depending on which option the user clicks.
Pizzas
Diavolo
Funghi
PizzasFragment
Bits and Pizzas
Home
Pizzas
Pasta
Stores
MainActivity
Pasta
Spaghetti Bolognese
Lasagne
PastaFragment
Stores
Cambridge
Sebastopol
StoresFragment

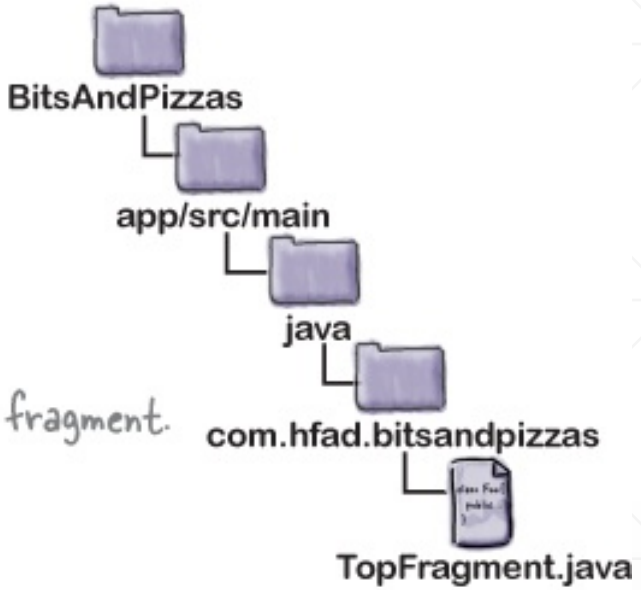
Steps

- Create fragments for the major hubs.
- Create and initialize the navigation drawer.
- Get the ListView to respond to item clicks.
- Add an ActionBarDrawerToggle.

Create TopFragment

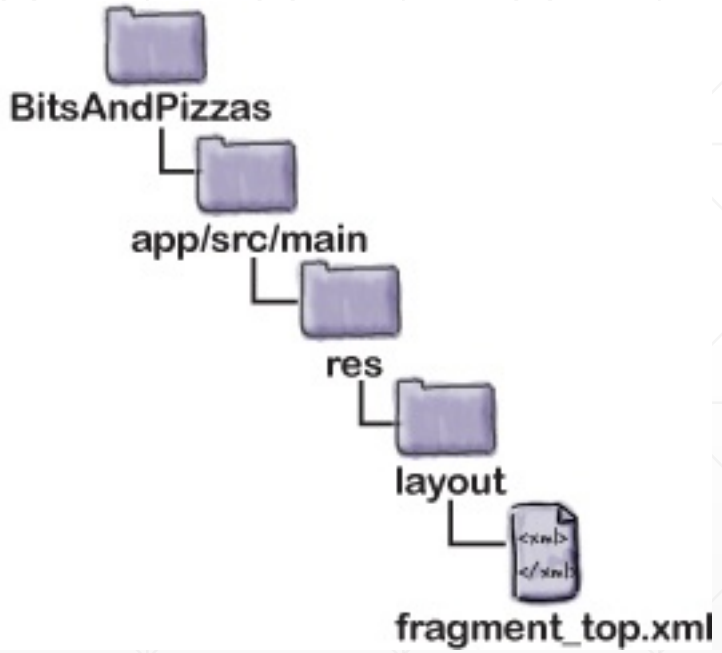
```
package com.hfad.bitsandpizzas;

import android.os.Bundle;
import android.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class TopFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_top, container, false);
    }
}
```

TopFragment.java is a plain fragment.



```
<string name="title_top">Top fragment</string>
```

Add this to strings.xml. We'll use it in the layout so we know when TopFragment is being displayed.




```
<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"
    tools:context=".MainActivity">
    <TextView
        android:text="@string/title_top"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

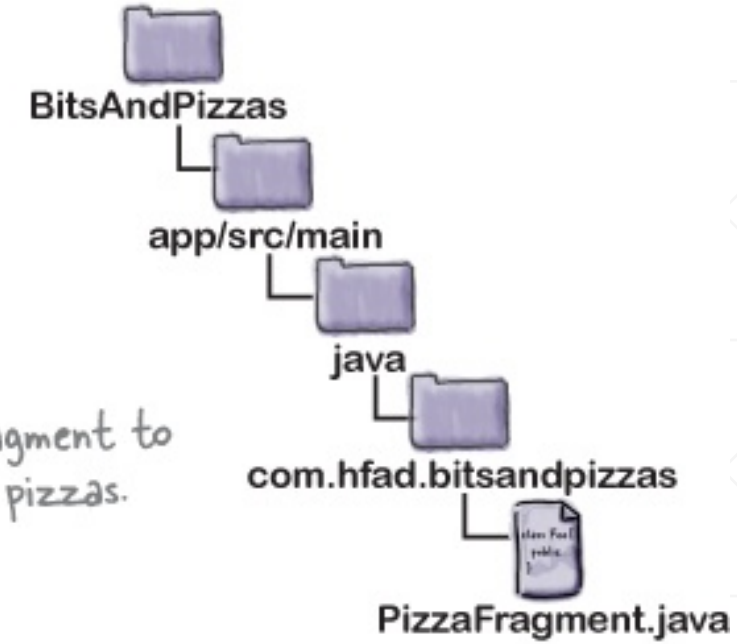
Create PizzaFragment

```
<string-array name="pizzas">
    <item>Diavolo</item>
    <item>Funghi</item>
</string-array>
```

Add the array of pizzas to strings.xml.

```
package com.hfad.bitsandpizzas;

import android.app.ListFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;

public class PizzaFragment extends ListFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                inflater.getContext(),
                android.R.layout.simple_list_item_1,
                getResources().getStringArray(R.array.pizzas));
        setListAdapter(adapter);
        return super.onCreateView(inflater, container, savedInstanceState);
    }
}
```

We'll use a ListFragment to display the list of pizzas.

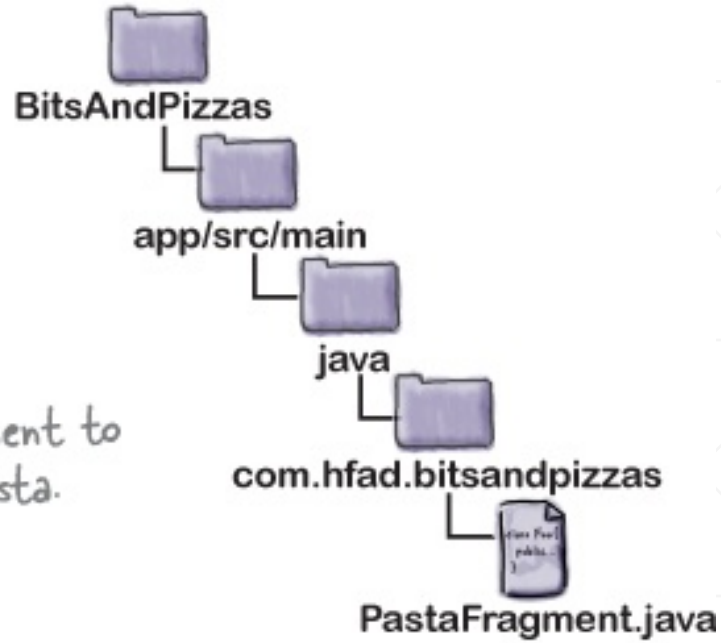


Create PastaFragment

```
<string-array name="pasta">
    <item>Spaghetti Bolognese</item>
    <item>Lasagne</item>
</string-array>
```

Add the array of pasta to strings.xml.

```
package com.hfad.bitsandpizzas;

import android.app.ListFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
```



We'll use a ListFragment to display the list of pasta.

```
public class PastaFragment extends ListFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                inflater.getContext(),
                android.R.layout.simple_list_item_1,
                getResources().getStringArray(R.array.pasta));
        setListAdapter(adapter);
        return super.onCreateView(inflater, container, savedInstanceState);
    }
}
```

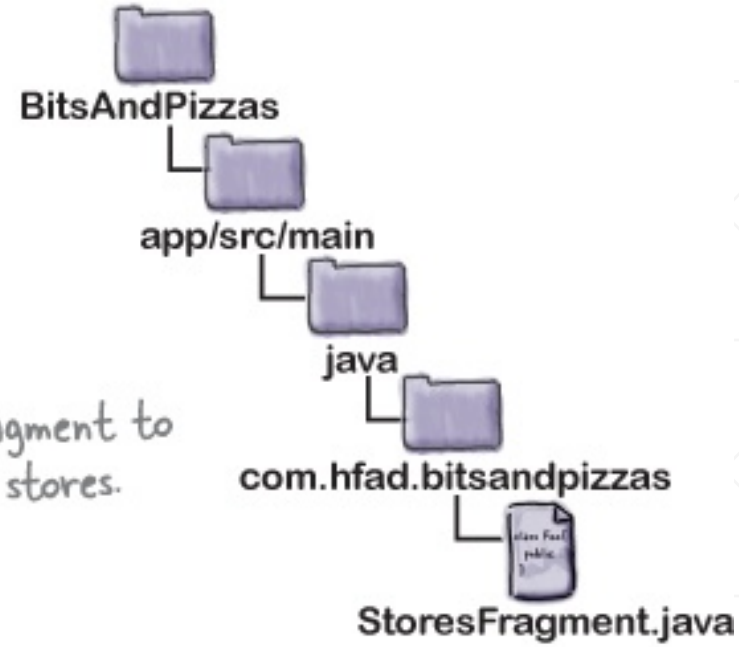
Create StoresFragment

```
<string-array name="stores">
    <item>Cambridge</item>
    <item>Sebastopol</item>
</string-array>
```

Add the array of stores to strings.xml.

```
package com.hfad.bitsandpizzas;

import android.app.ListFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;

public class StoresFragment extends ListFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                inflater.getContext(),
                android.R.layout.simple_list_item_1,
                getResources().getStringArray(R.array.stores));
        setListAdapter(adapter);
        return super.onCreateView(inflater, container, savedInstanceState);
    }
}
```

We'll use a ListFragment to display the list of stores.



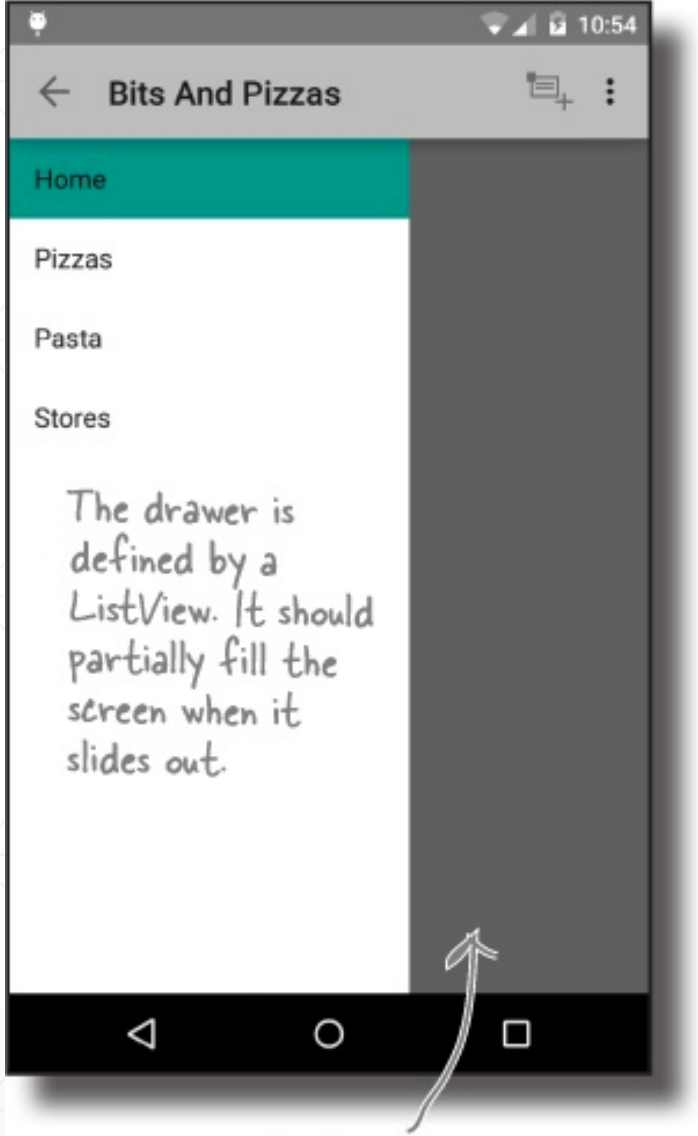
Add a DrawerLayout

The layout uses the DrawerLayout from the v4 support library. The v7 appcompat library includes the v4 support library.

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        ... />

    <ListView
        android:layout_width="240dp"
        android:layout_height="match_parent"
        ... />

</android.support.v4.widget.DrawerLayout>
```

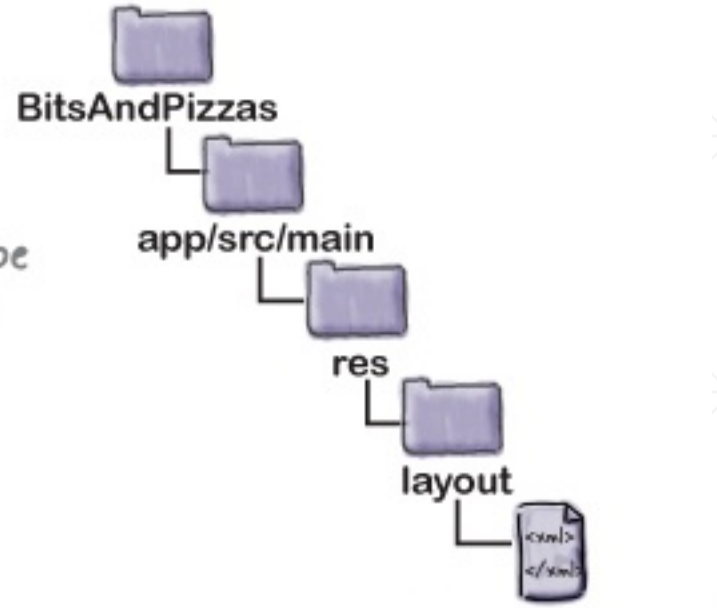
The FrameLayout will be used to display fragments.

The ListView describes the drawer.


10:54
Bits And Pizzas
Home
Pizzas
Pasta
Stores
The drawer is defined by a ListView. It should partially fill the screen when it slides out.
The content goes in a FrameLayout. You want the content to fill the screen. At the moment, it's partially hidden by the drawer.

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <ListView android:id="@+id/drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp"
        android:background="#ffffff"/>

</android.support.v4.widget.DrawerLayout>
```

Fragments will be displayed in the FrameLayout.



The ListView describes the drawer.

The drawer width.

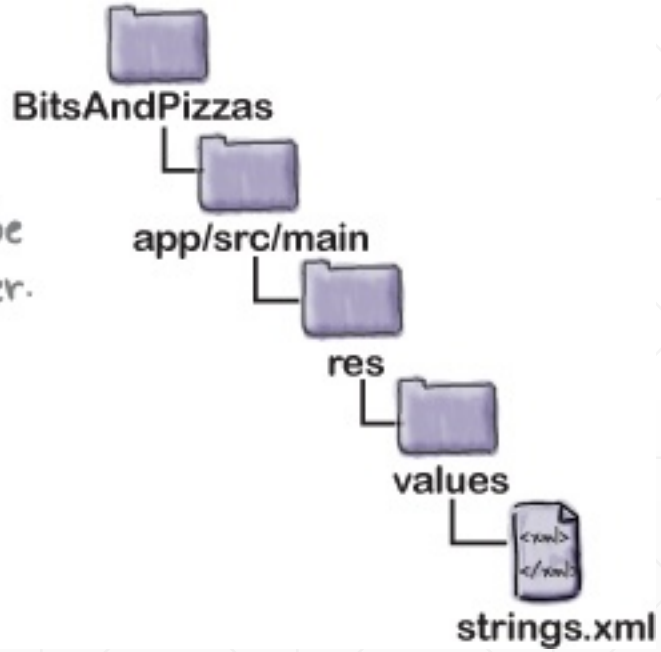
Where to place the drawer.

Select one item at a time.

Switch off the divider lines between items and set the background color of the drawer.

```
<string-array name="titles">
    <item>Home</item>
    <item>Pizzas</item>
    <item>Pasta</item>
    <item>Stores</item>
</string-array>
```

These are the options that will be displayed in the navigation drawer. Add the array to strings.xml.



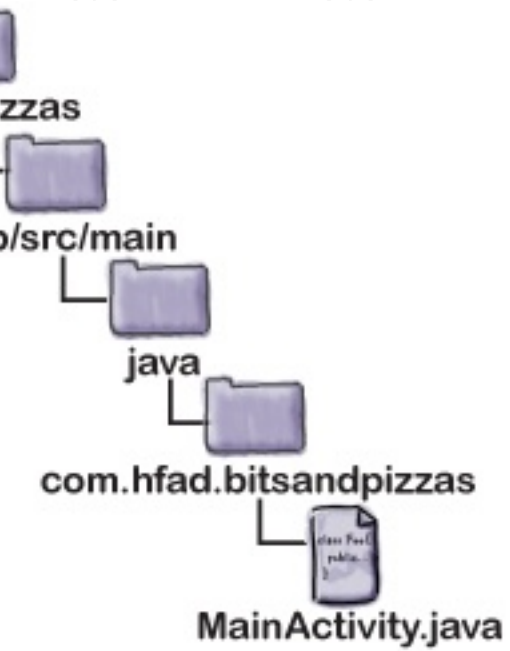
```
...
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends Activity {
    ...
    private String[] titles;
    private ListView drawerList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        titles = getResources().getStringArray(R.array.titles);
        drawerList = (ListView)findViewById(R.id.drawer);
        drawerList.setAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_activated_1, titles));
    }
...
}
```

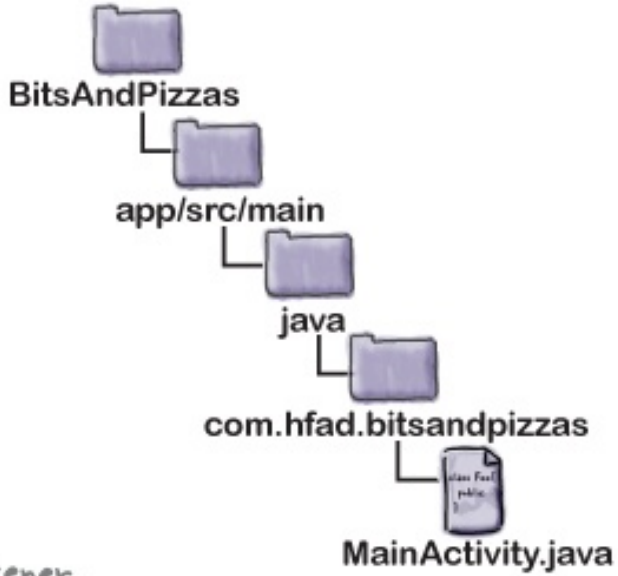
We're using these classes, so we need to import them.

We'll use these in other methods later on, so add them as private class variables.

Use an ArrayAdapter to populate the ListView.

Using simple_list_item_activiated_1 means that the item the user clicks on is highlighted.






```
...
import android.view.View;
import android.widget.AdapterView;

public class MainActivity extends Activity {
    ...

    private class DrawerItemClickListener implements ListView.OnItemClickListener {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id){
            //Code to run when the item gets clicked
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        drawerList.setOnItemClickListener(new DrawerItemClickListener());
    }
};
```

We're using these classes, so we need to import them.

This describes the OnItemClickListener.

When the user clicks on an item in the navigation drawer, the onItemClick() method gets called.

Add a new instance of our OnItemClickListener to the drawer's ListView.

onItemClick()

- The method should do three things:
  - Switch the fragment in the frame layout.
  - Change the title in the action bar to reflect the layout.
  - Close the navigation drawer.

```
...
import android.app.Fragment;
import android.app.FragmentTransaction;

public class MainActivity extends Activity {
    ...
    private class DrawerItemClickListener implements ListView.OnItemClickListener {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            selectItem(position);
        }
    };

    private void selectItem(int position) {
        Fragment fragment;
        switch(position) {
            case 1:
                fragment = new PizzaFragment();
                break;
            case 2:
                fragment = new PastaFragment();
                break;
            case 3:
                fragment = new StoresFragment();
                break;
            default:
                fragment = new TopFragment();
        }
        FragmentTransaction ft = getFragmentManager().beginTransaction();
        ft.replace(R.id.content_frame, fragment);
        ft.addToBackStack(null);
        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
        ft.commit();
    }
}
```

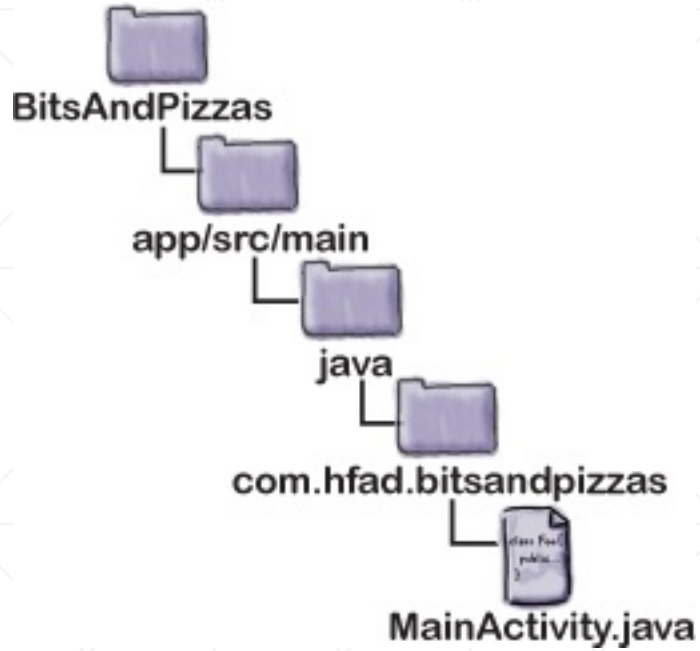
Call the selectItem() method when an item gets clicked.

Check the position of the item that was clicked.

Use the position to create the right type of fragment. The "Pizzas" option is at position 1, for instance, so in this case create a PizzaFragment.

Create a TopFragment by default

Use fragment transaction to replace the fragment that's displayed.



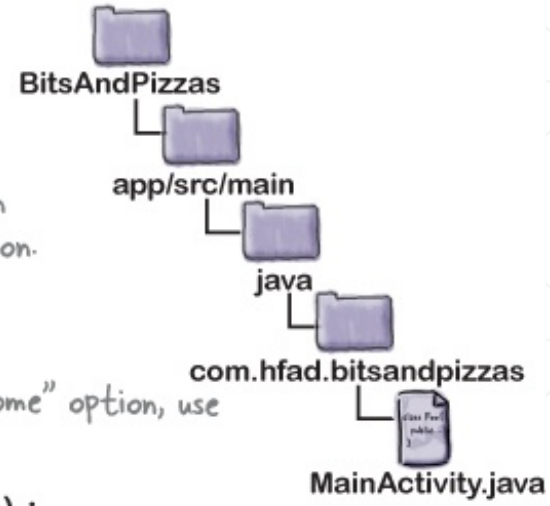
```
private void selectItem(int position) {
    ...
    //Set the action bar title
    setActionBarTitle(position);
}

private void setActionBarTitle(int position) {
    String title;
    if (position == 0){
        title = getResources().getString(R.string.app_name);
    } else {
        title = titles[position];
    }
    getActionBar().setTitle(title);
}
```

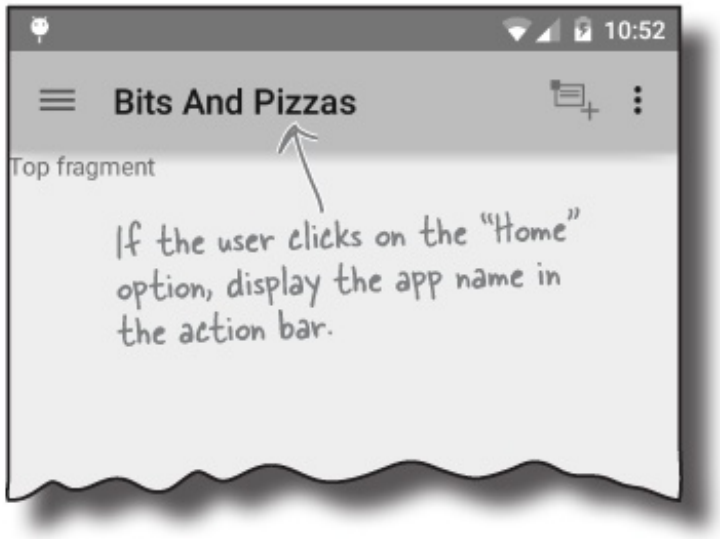
Call the setActionTitle() method, passing it the position of the item that was clicked on.

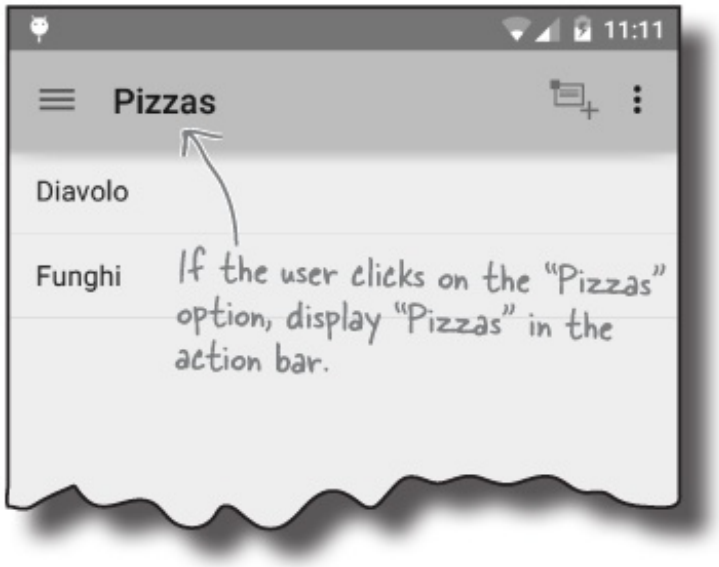
If the user clicks on the "Home" option, use the app name for the title.

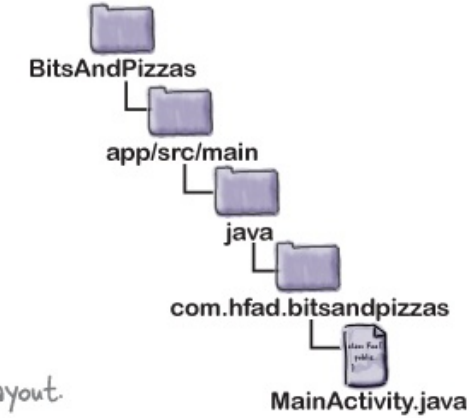
Otherwise, get the String from the titles array for the position that was clicked and wse that

Display the title in the action bar.










```
private void selectItem(int position) {
    ...
    //Close the drawer
    DrawerLayout drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawerLayout.closeDrawer(drawerList);
}
```

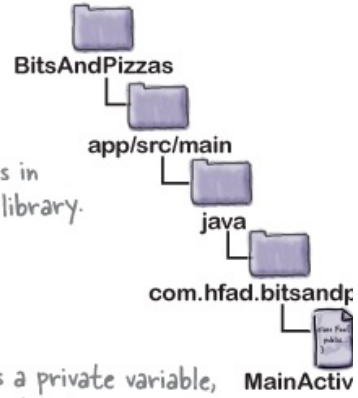
Get a reference to the DrawerLayout.

drawerList is the DrawerLayout's drawer. This tells the DrawerLayout to close the drawerList drawer.

The entire screen is the DrawerLayout. It contains a FrameLayout where the content goes, and a ListView which is used for the drawer.

You need to tell the DrawerLayout to close its ListView drawer.

```
package com.hfad.bitsandpizzas;

...

import android.support.v4.widget.DrawerLayout;

public class MainActivity extends Activity {
    ...
    private DrawerLayout drawerLayout;

    private class DrawerItemClickListener implements ListView.OnItemClickListener {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long
            //Code to run when an item in the navigation drawer gets clicked
            selectItem(position);
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        titles = getResources().getStringArray(R.array.titles);
        drawerList = (ListView)findViewById(R.id.drawer);
        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        //Populate the ListView
        drawerList.setAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_activated_1, titles));
        drawerList.setOnItemClickListener(new DrawerItemClickListener());
        if (savedInstanceState == null) {
            selectItem(0);
        }
    }
}
```



DrawerLayout is in the v4 support library.

Add the DrawerLayout as a private variable, as we'll use it in multiple methods.

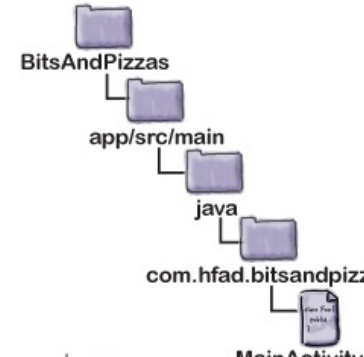
Call the selectItem() method.

Get a reference to DrawerLayout.

If the MainActivity is newly created, use the selectItem() method to display TopFragment.

```
private void selectItem(int position) {
    // update the main content by replacing fragments
    Fragment fragment;
    switch(position) {
        case 1:
            fragment = new PizzaFragment();
            break;
        case 2:
            fragment = new PastaFragment();
            break;
        case 3:
            fragment = new StoresFragment();
            break;
        default:
            fragment = new TopFragment();
    }
    FragmentTransaction ft = getFragmentManager().beginTransaction();
    ft.replace(R.id.content_frame, fragment);
    ft.addToBackStack(null);
    ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
    ft.commit();
    //Set the action bar title
    setActionBarTitle(position);
    //Close drawer
    drawerLayout.closeDrawer(drawerList);
}

private void setActionBarTitle(int position) {
    String title;
    if (position == 0){
        title = getResources().getString(R.string.app_name);
    } else {
        title = titles[position];
    }
    getActionBar().setTitle(title);
}
...
```

Get the right fragment to display.



Display the fragment using a fragment transaction.

Set the action bar title.

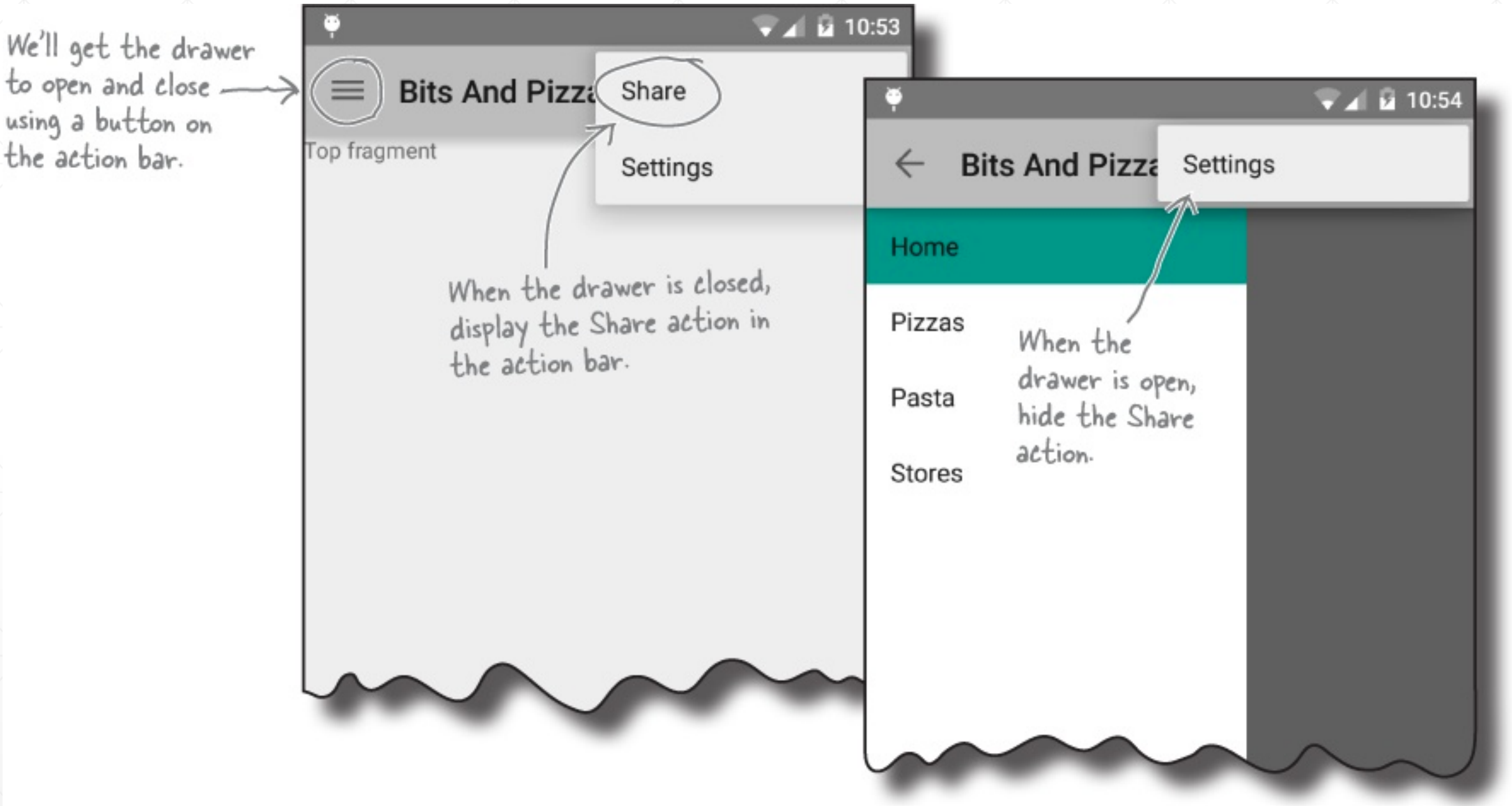
Close the drawer.

If the user clicks on the "Home" option, use the app name for the titl

Otherwise, get the String from the titles array for the position that was clicked and use that

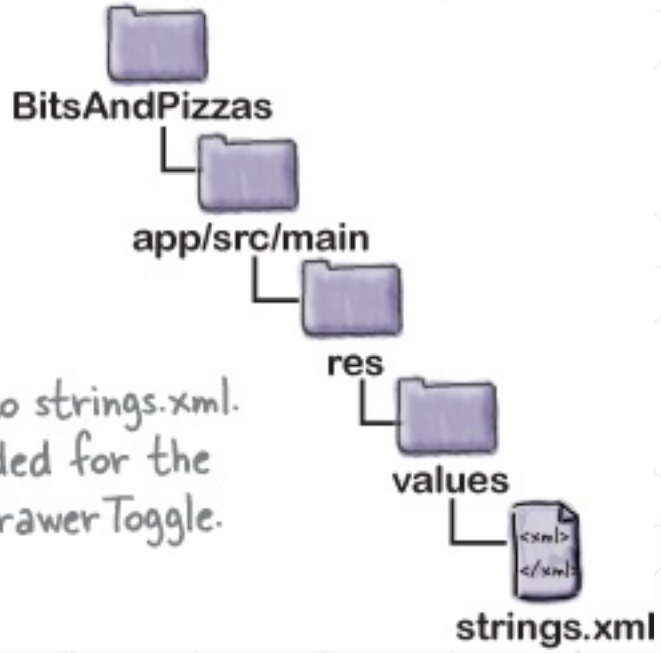
Display the title in the action bar.

We've omitted the onCreateOptionsMenu() and onOptionsItemSelected() methods from our original ManActivity code, as these haven't changed.


We'll get the drawer to open and close using a button on the action bar.
10:53
Bits And Pizza
Share
Settings
Top fragment
When the drawer is closed, display the Share action in the action bar.
10:54
Bits And Pizza
Settings
Home
Pizzas
Pasta
Stores
When the drawer is open, hide the Share action.

```
<string name="open_drawer">Open drawer</string>
<string name="close_drawer">Close drawer</string>
```

Add these to strings.xml. They're needed for the ActionBarDrawerToggle.

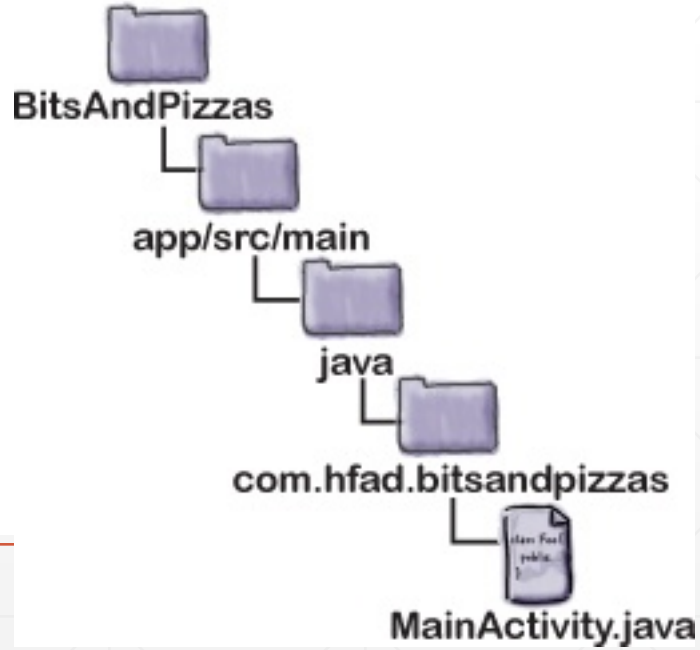


Create the ActionBarDrawerToggle.

```
ActionBarDrawerToggle drawerToggle = new ActionBarDrawerToggle(this, drawerLayout,
                R.string.open_drawer, R.string.close_drawer) {

    //Called when a drawer has settled in a completely closed state
    @Override
    public void onDrawerClosed(View view) {
        super.onDrawerClosed(view);
        //Code to run when the drawer is closed
    }

    //Called when a drawer has settled in a completely open state.
    @Override
    public void onDrawerOpened(View drawerView) {
        super.onDrawerOpened(drawerView);
        //Code to run when the drawer is open
    }
};
```

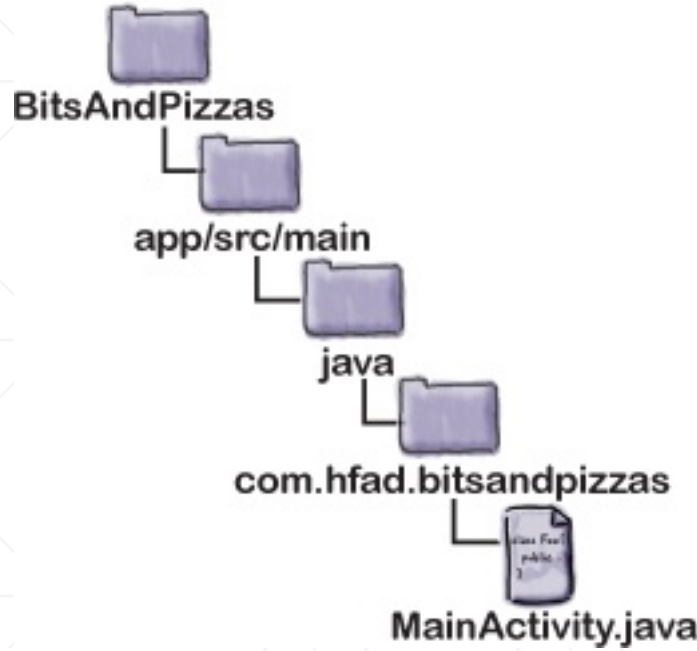
This method gets called when the drawer is closed.

This method gets called when the drawer is open.



```
drawerLayout.setDrawerListener(drawerToggle);
```

Set the ActionBarDrawerToggle as the DrawerLayout's drawer listener.




```
public void onDrawerClosed(View view) {
    super.onDrawerClosed(view);
    invalidateOptionsMenu();
}

public void onDrawerOpened(View drawerView) {
    super.onDrawerOpened(drawerView);
    invalidateOptionsMenu();
}
```

The invalidateOptionsMenu() method tells Android to re-create the menu items. We want to change the visibility of the Share action if the drawer is opened or closed, so we call it in the onDrawerOpened() and onDrawerClosed() methods.

```
//Called whenever we call invalidateOptionsMenu()
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // If the drawer is open, hide action items related to the content view
    boolean drawerOpen = drawerLayout.isDrawerOpen(drawerList);
    menu.findItem(R.id.action_share).setVisible(!drawerOpen);
    return super.onPrepareOptionsMenu(menu);
}
```

The onPrepareOptionsMenu() method gets called whenever invalidateOptionsMenu() gets called.

Set the Share action's visibility to false if the drawer is open, set it to true if it isn't.

```
...
import android.support.v7.app.ActionBarDrawerToggle;

public class MainActivity extends Activity {
    ...
    private ActionBarDrawerToggle drawerToggle;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        //Create the ActionBarDrawerToggle
        drawerToggle = new ActionBarDrawerToggle(this, drawerLayout,
                R.string.open_drawer, R.string.close_drawer) {
            //Called when a drawer has settled in a completely closed state
            public void onDrawerClosed(View view) {
                super.onDrawerClosed(view);
                invalidateOptionsMenu();
            }
            //Called when a drawer has settled in a completely open state.
            public void onDrawerOpened(View drawerView) {
                super.onDrawerOpened(drawerView);
                invalidateOptionsMenu();
            }
        };
        drawerLayout.setDrawerListener(drawerToggle);
    }

    //Called whenever we call invalidateOptionsMenu()
    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        // If the drawer is open, hide action items related to the content view
        boolean drawerOpen = drawerLayout.isDrawerOpen(drawerList);
        menu.findItem(R.id.action_share).setVisible(!drawerOpen);
        return super.onPrepareOptionsMenu(menu);
    }
    ...
}
```

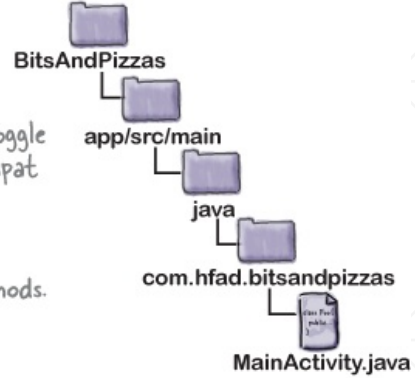
ActionBarDrawerToggle is in the v7 appcompat library.

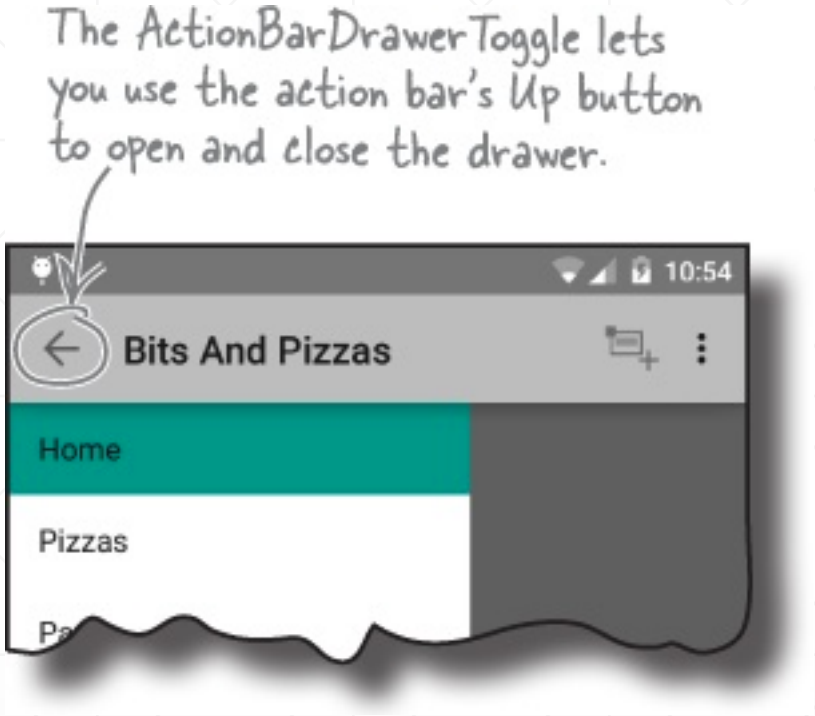
Set this as a private variable, as we'll use it in multiple methods.

Call invalidateOptionsMenu() when the drawer is opened or closed.

Set the DrawerLayout's drawer listener as the ActionBarDrawerToggle.

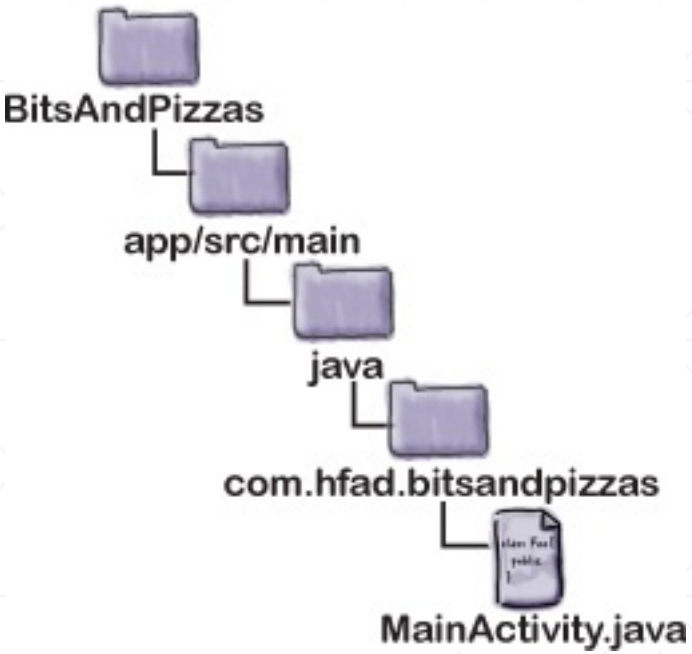
Set the visibility of the Share action when the drawer is opened and closed.




The ActionBarDrawerToggle lets you use the action bar's Up button to open and close the drawer.
10:54
Bits And Pizzas
Home
Pizzas

```
getActionBar().setDisplayHomeAsUpEnabled(true);
getActionBar().setHomeButtonEnabled(true);
```

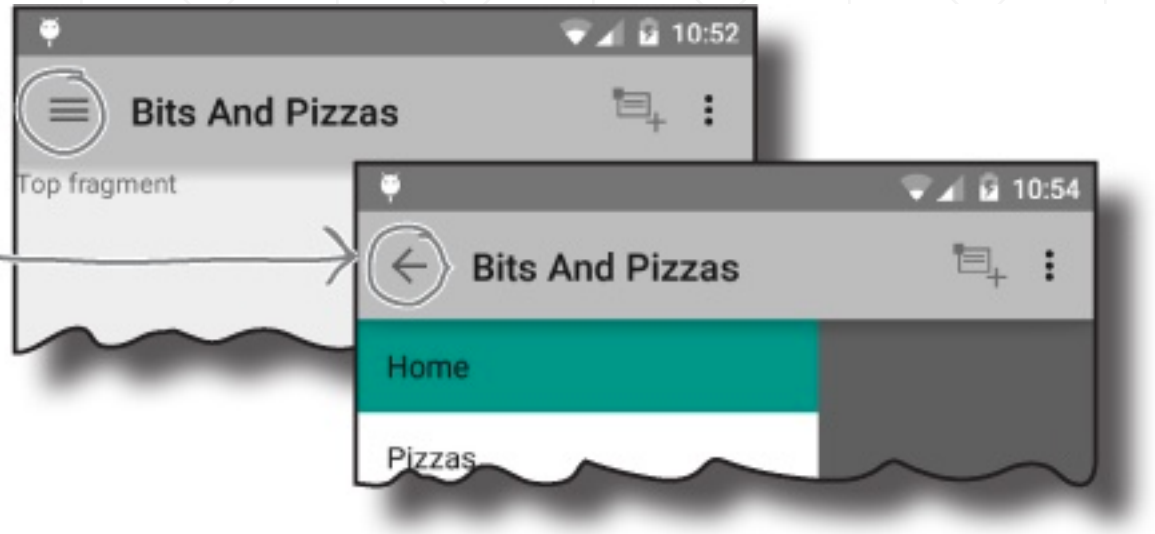
Enable the Up button so you can use it for the drawer.

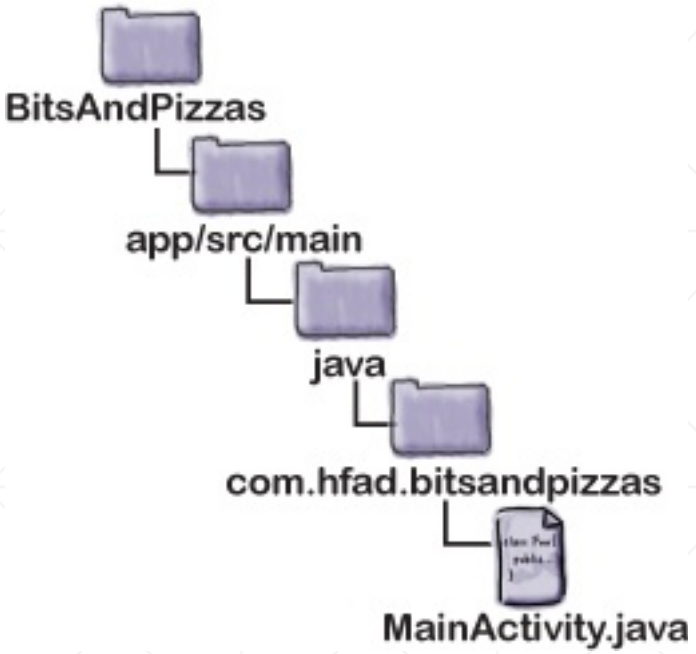



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (drawerToggle.onOptionsItemSelected(item)) {
        return true;
    }
    //Code to handle the rest of the action items
    ...
    }
}
```

You need to add these lines of code to the onOptionsItemSelected() method so that the ActionBarDrawerToggle can handle being clicked.

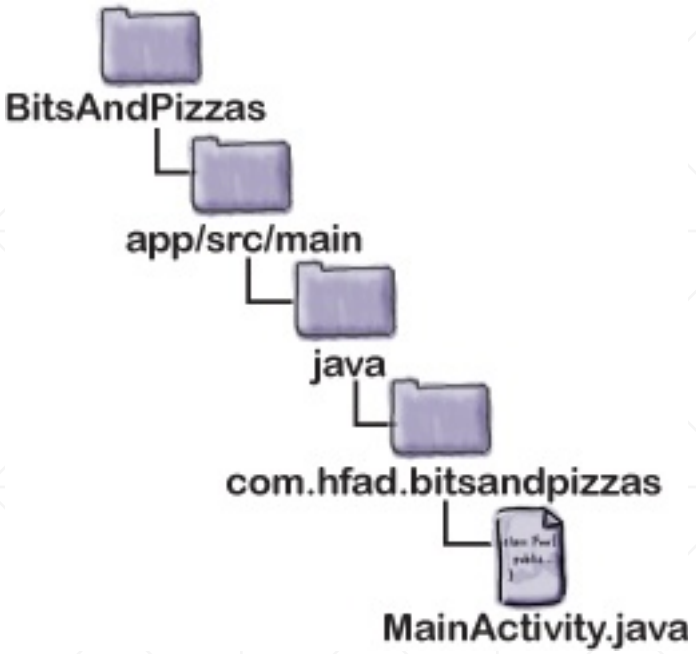
Syncing the state means that the drawer icon appears one way when the drawer is closed, and another way when the drawer is open.





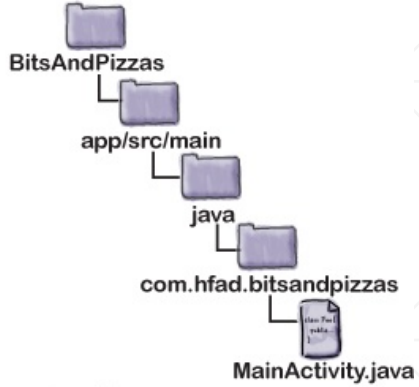

```
@Override
protected void onPostCreate(Bundle savedInstanceState) {
    super.onPostCreate(savedInstanceState);
    // Sync the toggle state after onRestoreInstanceState has occurred.
    drawerToggle.syncState();
}
```

You need to add this method to your activity so that the state of the ActionBarDrawerToggle is in sync with the state of the drawer.




```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    drawerToggle.onConfigurationChanged(newConfig);
}
```

You need to add this method to your activity so that any configuration changes get passed to the ActionBarDrawerToggle.




```
...
import android.content.res.Configuration;
```

Import this class as it's used by the onConfigurationChanged() method.

```
public class MainActivity extends Activity {
    ...
    private ActionBarDrawerToggle drawerToggle;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        getActionBar().setDisplayHomeAsUpEnabled(true);
        getActionBar().setHomeButtonEnabled(true);
    }
```

Enable the Up icon so it can be used by the ActionBarDrawerToggle.

```
    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        drawerToggle.syncState();
    }
```

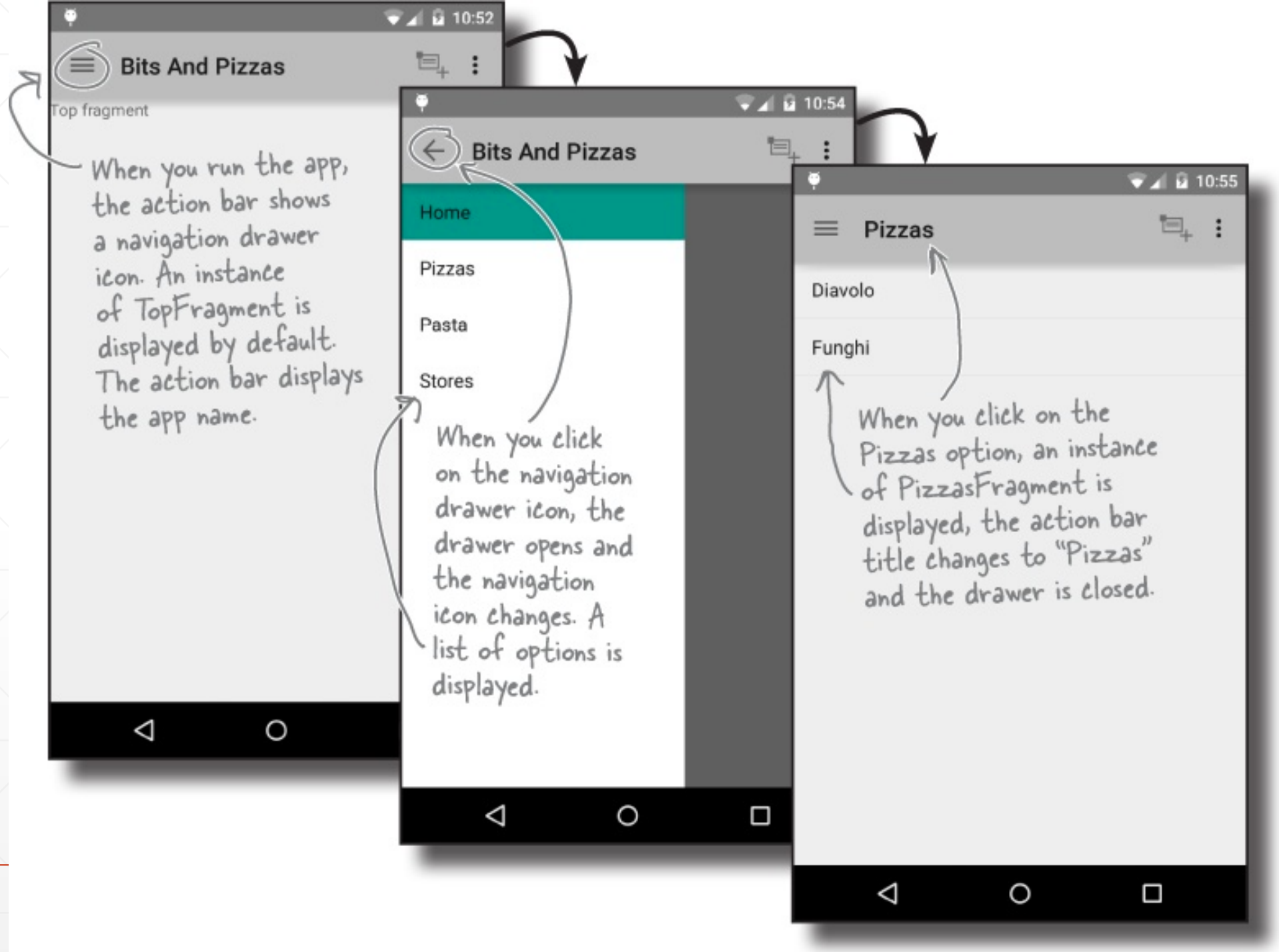
Sync the state of the ActionBarDrawerToggle with the state of the drawer.

```
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        drawerToggle.onConfigurationChanged(newConfig);
    }
```

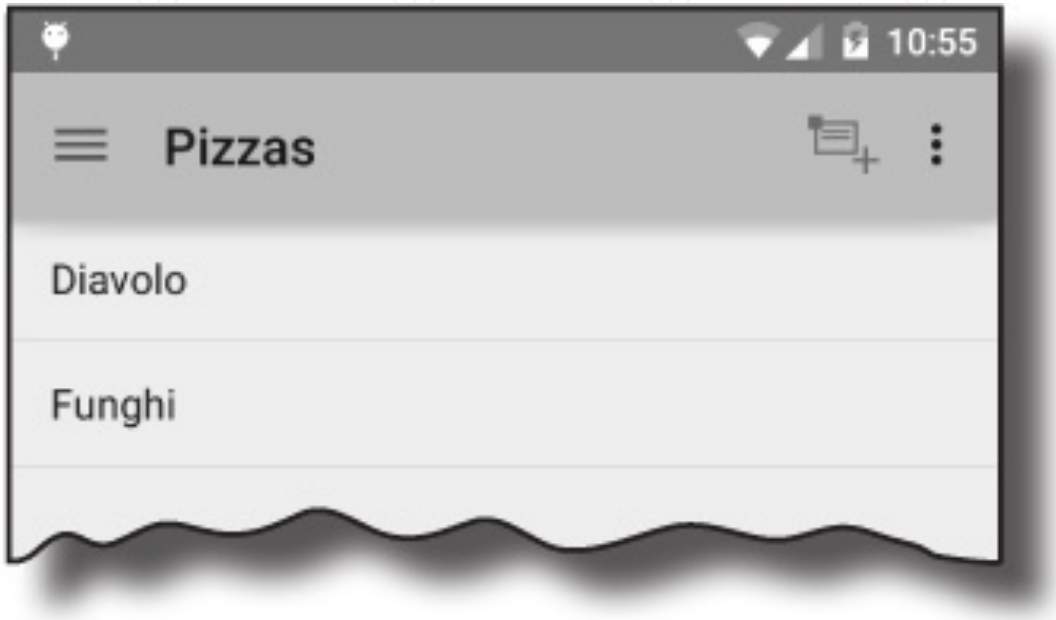
Pass any configuration changes to the ActionBarDrawerToggle.

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (drawerToggle.onOptionsItemSelected(item)) {
            return true;
        }
        //Code to handle the rest of the action items
        switch (item.getItemId()) {
        ...
    }
    ...
}
```

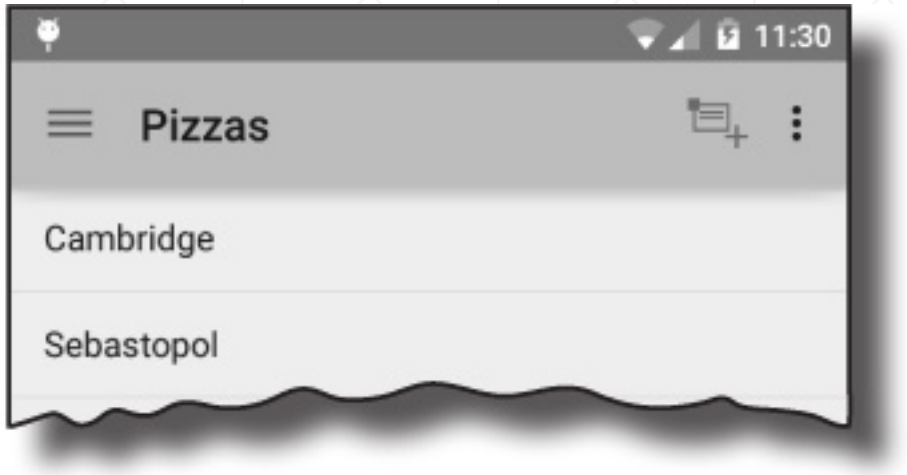
Let the ActionBarDrawerToggle handle being clicked.


10:52
Bits And Pizzas
Top fragment
When you run the app, the action bar shows a navigation drawer icon. An instance of TopFragment is displayed by default. The action bar displays the app name.
10:54
Bits And Pizzas
Home
Pizzas
Pasta
Stores
When you click on the navigation drawer icon, the drawer opens and the navigation icon changes. A list of options is displayed.
10:55
Pizzas
Diavolo
Funghi
When you click on the Pizzas option, an instance of PizzasFragment is displayed, the action bar title changes to "Pizzas" and the drawer is closed.

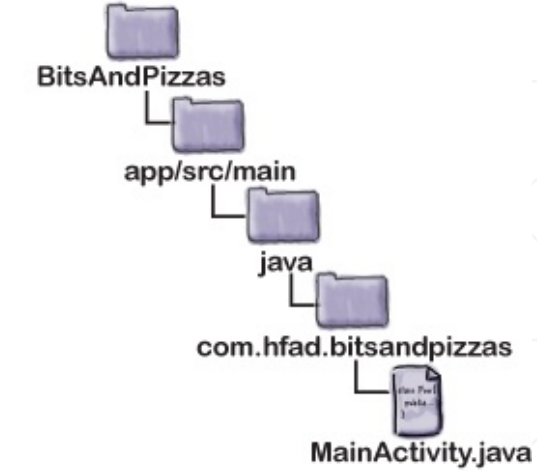
When you click on items in the navigation drawer, the title gets updated correctly.



The action bar title stays the same when we click on the Back button. In this case, it says "Pizzas" when a list of stores is displayed



The action bar title is reset when you rotate the device.




```
...
public class MainActivity extends Activity {
    ...
    private int currentPosition = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        //Display the correct fragment.
        if (savedInstanceState != null) {
            currentPosition = savedInstanceState.getInt("position");
            setActionBarTitle(currentPosition);
        } else {
            selectItem(0);
        }
        ...
    }

    private void selectItem(int position) {
        currentPosition = position;
        ...
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt("position", currentPosition);
    }
    ...
}
```

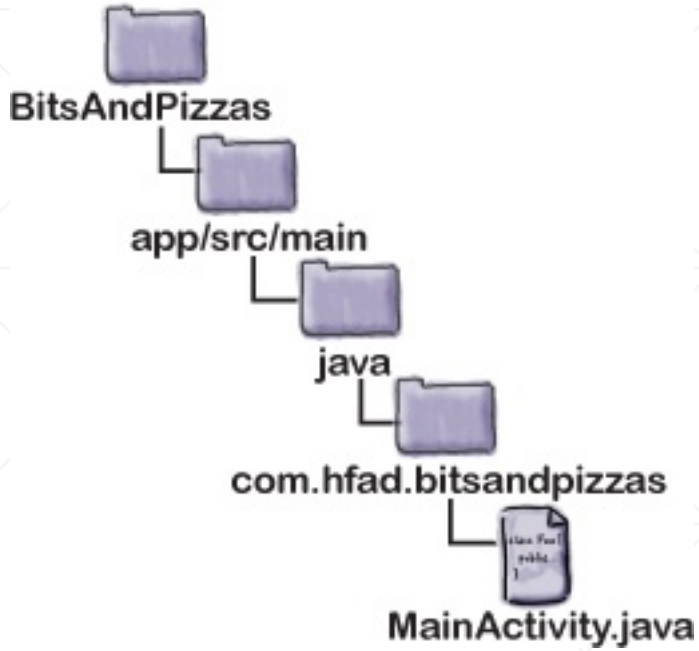
Set currentPosition to 0 by default.

If the activity has been destroyed and re-created, set the value of currentPosition from the activity's previous state and use it to set the action bar title.

If the activity's newly created, display TopFragment.

Update currentPosition when an item is selected.

Save the state of currentPosition if the activity's going to be destroyed.

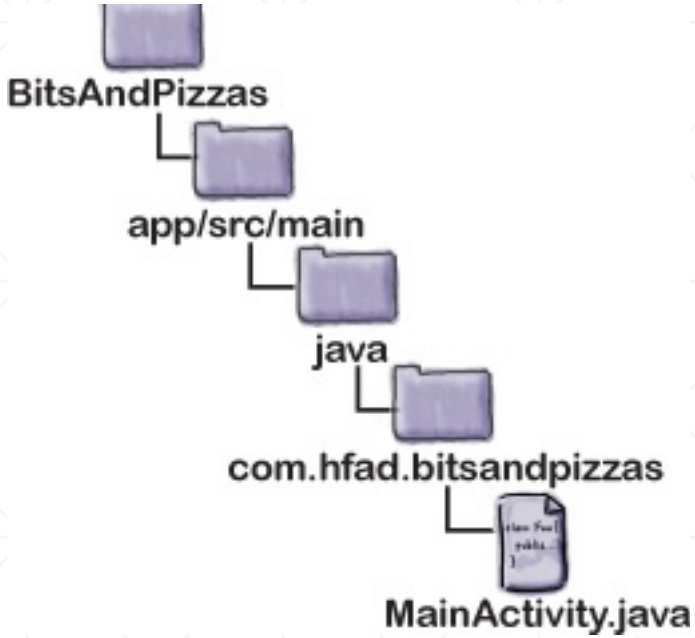



```
getFragmentManager().addOnBackStackChangedListener(
    new FragmentManager.OnBackStackChangedListener() {
        public void onBackStackChanged() {
            //Code to run when the back stack changes
        }
    }
);
```

You add a new FragmentManager.OnBackStackChangedListener, implementing its onBackStackChanged() method. This method is called whenever the back stack changes.

onBackStackChanged()

- Update the currentPosition variable so that it reflects the position in the list view of the currently displayed fragment.
- Call the setActionBarTitle() method, passing it the value of currentPosition.
- Make sure that the right option in the navigation drawer's list view is highlighted by calling itssetItemChecked() method.




```
FragmentTransaction ft = getFragmentManager().beginTransaction();
ft.replace(R.id.content_frame, fragment);
ft.addToBackStack(null);
ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
ft.commit();
```

```
FragmentTransaction ft = getFragmentManager().beginTransaction();
ft.replace(R.id.content_frame, fragment, "visible_fragment");
ft.addToBackStack(null);
ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
ft.commit();
```

This adds a tag of "visible_fragment" to the fragment as it's added to the back stack.

Find a fragment with a tag of "visible_fragment".

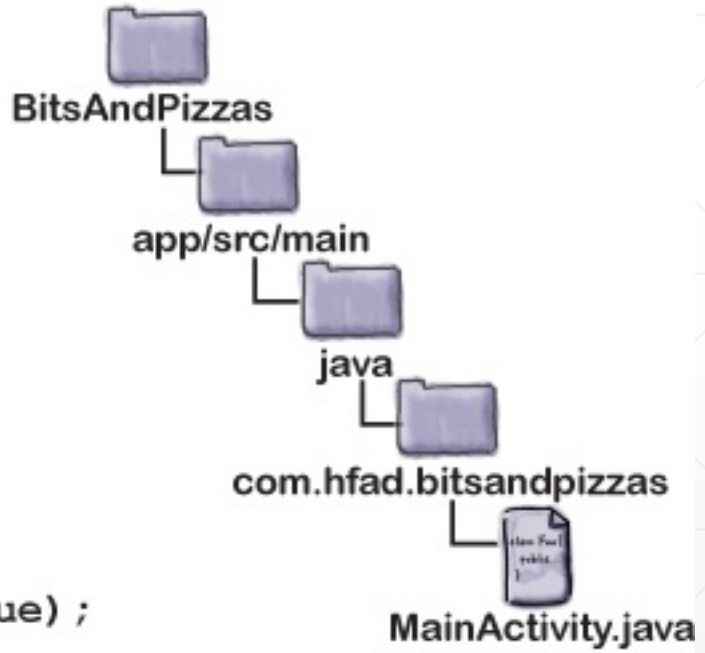
```
FragmentManager fragMan = getFragmentManager();
Fragment fragment = fragMan.findFragmentByTag("visible_fragment");
```

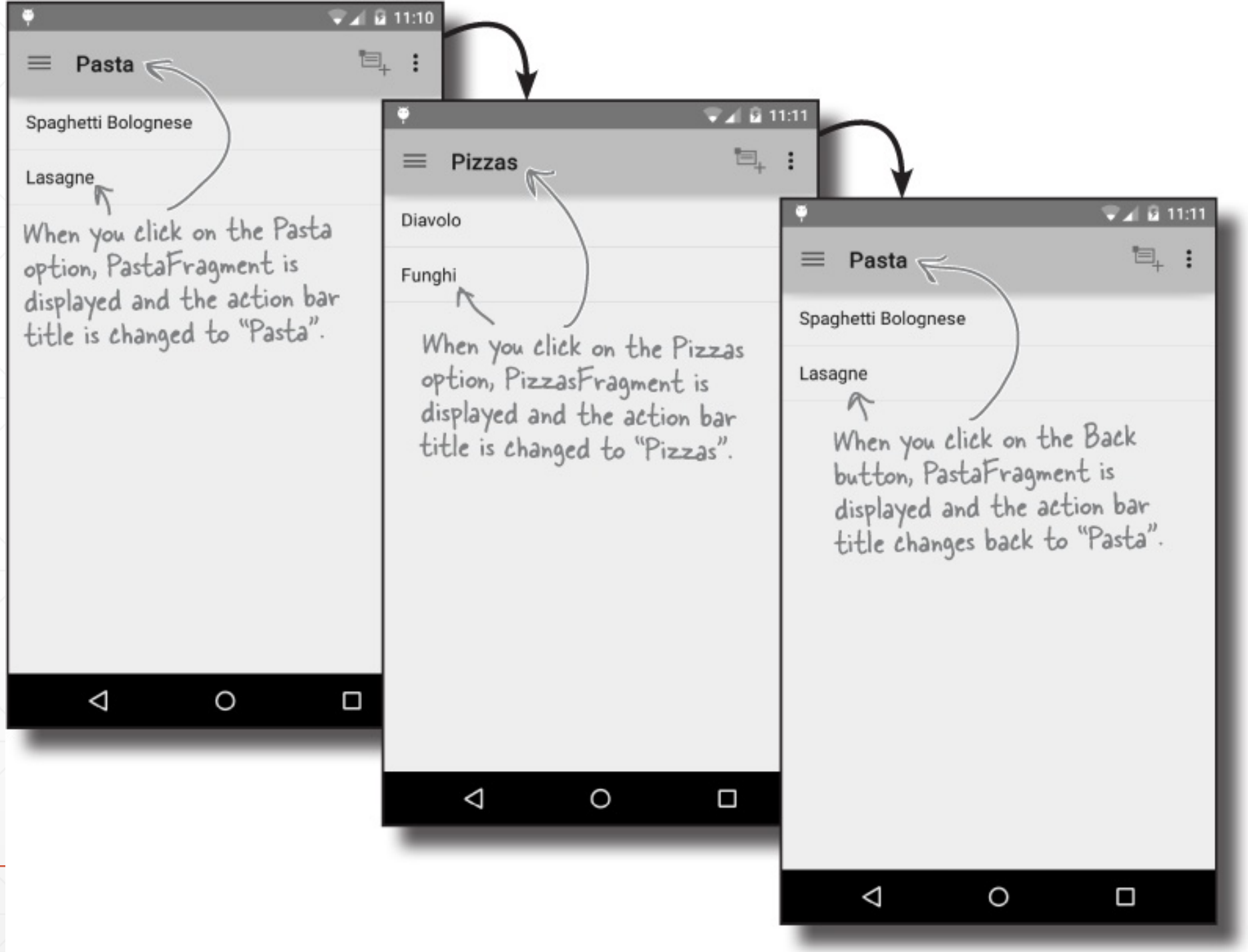
```
getFragmentManager().addOnBackStackChangedListener(
    new FragmentManager.OnBackStackChangedListener() {
        public void onBackStackChanged() {
            FragmentManager fragMan = getFragmentManager();
            Fragment fragment = fragMan.findFragmentByTag("visible_fragment");
            if (fragment instanceof TopFragment) {
                currentPosition = 0;
            }
            if (fragment instanceof PizzaFragment) {
                currentPosition = 1;
            }
            if (fragment instanceof PastaFragment) {
                currentPosition = 2;
            }
            if (fragment instanceof StoresFragment) {
                currentPosition = 3;
            }
            setActionBarTitle(currentPosition);
            drawerList.setItemChecked(currentPosition, true);
        }
    }
);
```

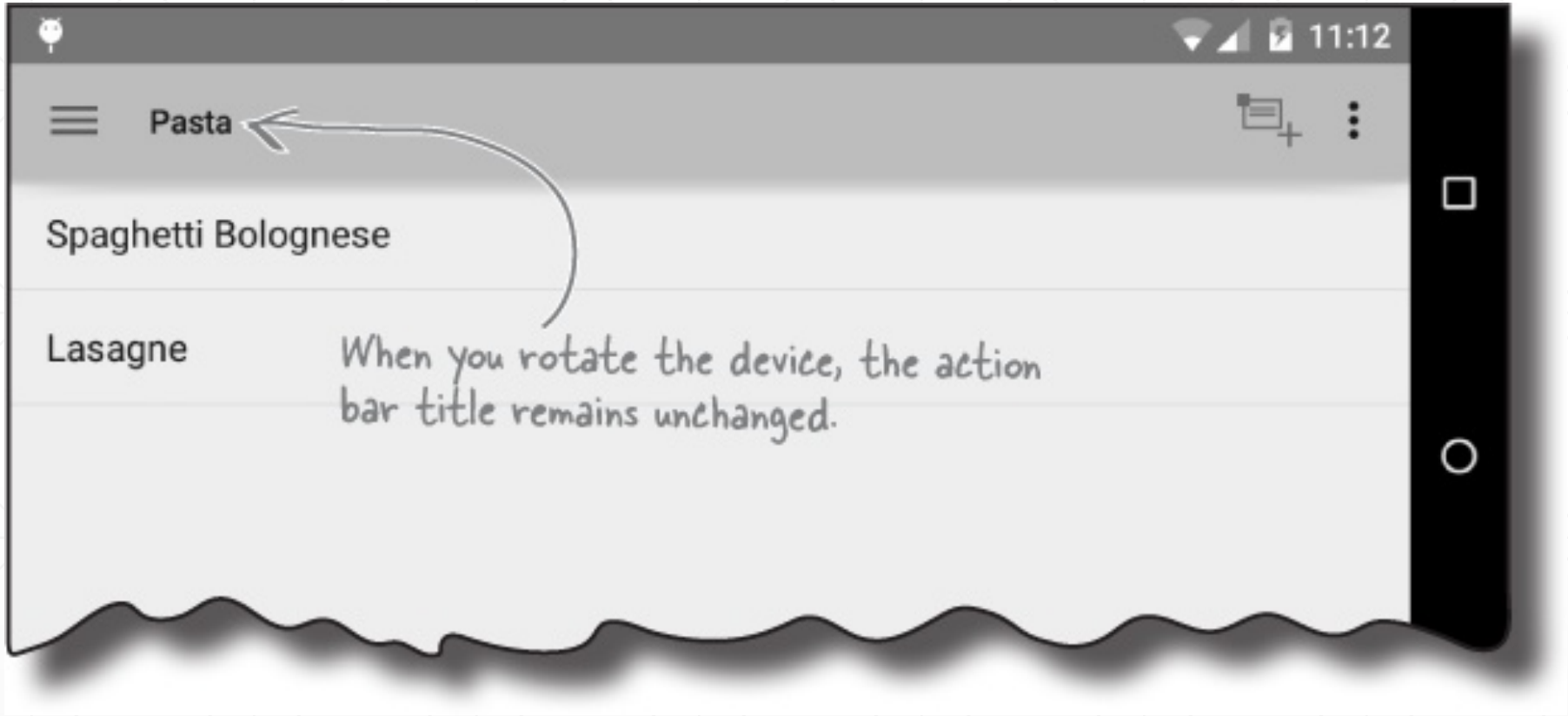
This gets the fragment currently attached to the activity.

Check what type of fragment it is, and set currentPosition accordingly.

Set the action bar title and highlight the correct item in the drawer ListView.




11:10
Pasta
Spaghetti Bolognese
Lasagne
When you click on the Pasta option, PastaFragment is displayed and the action bar title is changed to "Pasta".
11:11
Pizzas
Diavolo
Funghi
When you click on the Pizzas option, PizzasFragment is displayed and the action bar title is changed to "Pizzas".
11:11
Pasta
Spaghetti Bolognese
Lasagne
When you click on the Back button, PastaFragment is displayed and the action bar title changes back to "Pasta".


11:12
Pasta
Spaghetti Bolognese
Lasagne
When you rotate the device, the action bar title remains unchanged.

Bullet Points

- Use a DrawerLayout to create an activity with a navigation drawer. Use the drawer to navigate to the major hubs of your app.
- If you're using an action bar, use ActionBarDrawerToggle as a DrawerListener. This allows you to respond to the drawer opening and closing, and adds an icon to the action bar for opening and closing the drawer.
- To change action bar items at runtime, call invalidateOptionsMenu() and add the changes in the activity's onPrepareOptionsMenu() method.
- React to changes on the back stack by implementing theFragmentManager.OnBackStackChangedListener().
- The fragment manager's findFragmentByTag() method searches for fragments with a given tag.