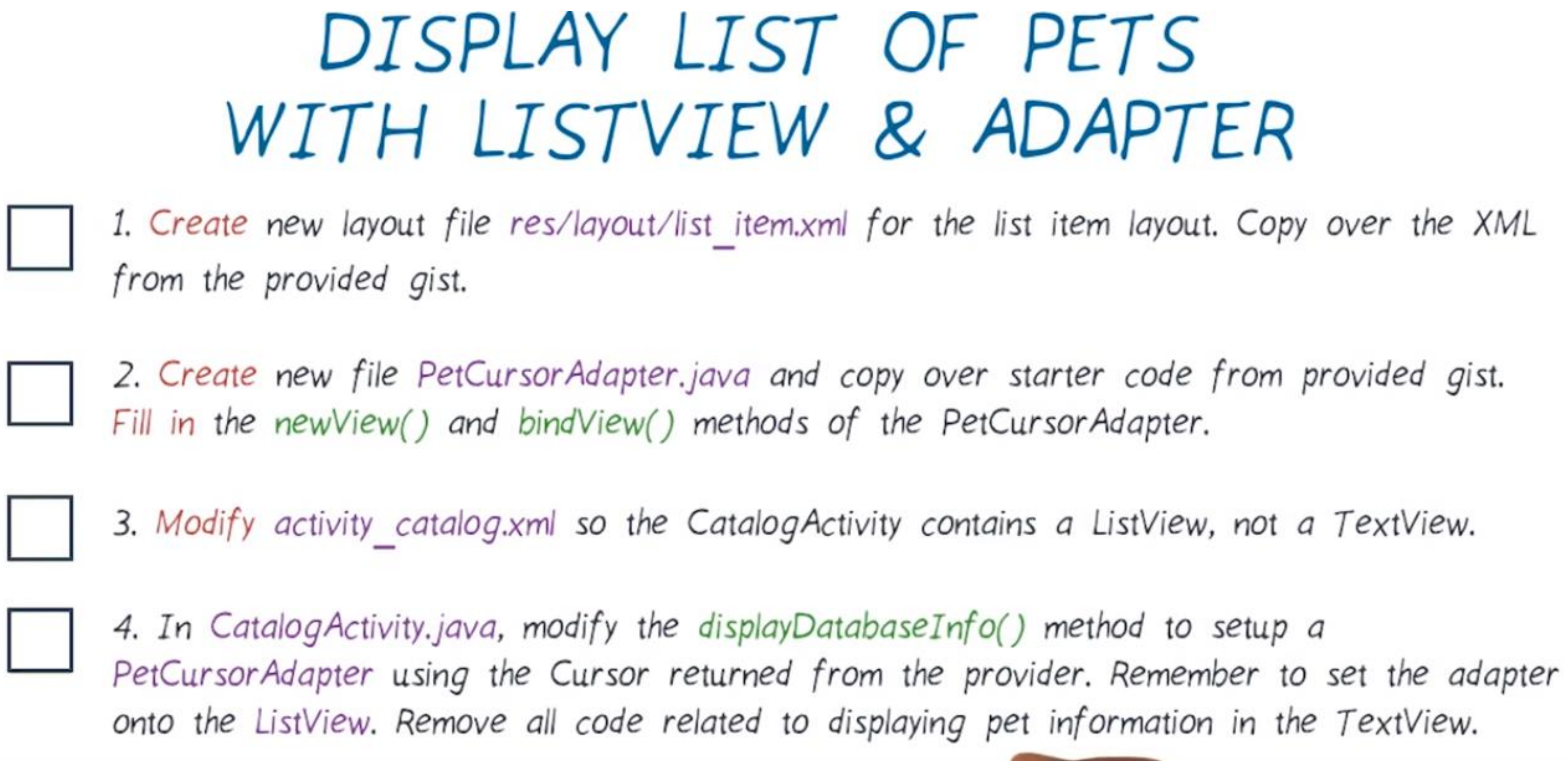
Pets part 3

17 - Create PetCursorAdapter to display list of pets in ListView



DISPLAY LIST OF PETS
WITH LISTVIEW & ADAPTER

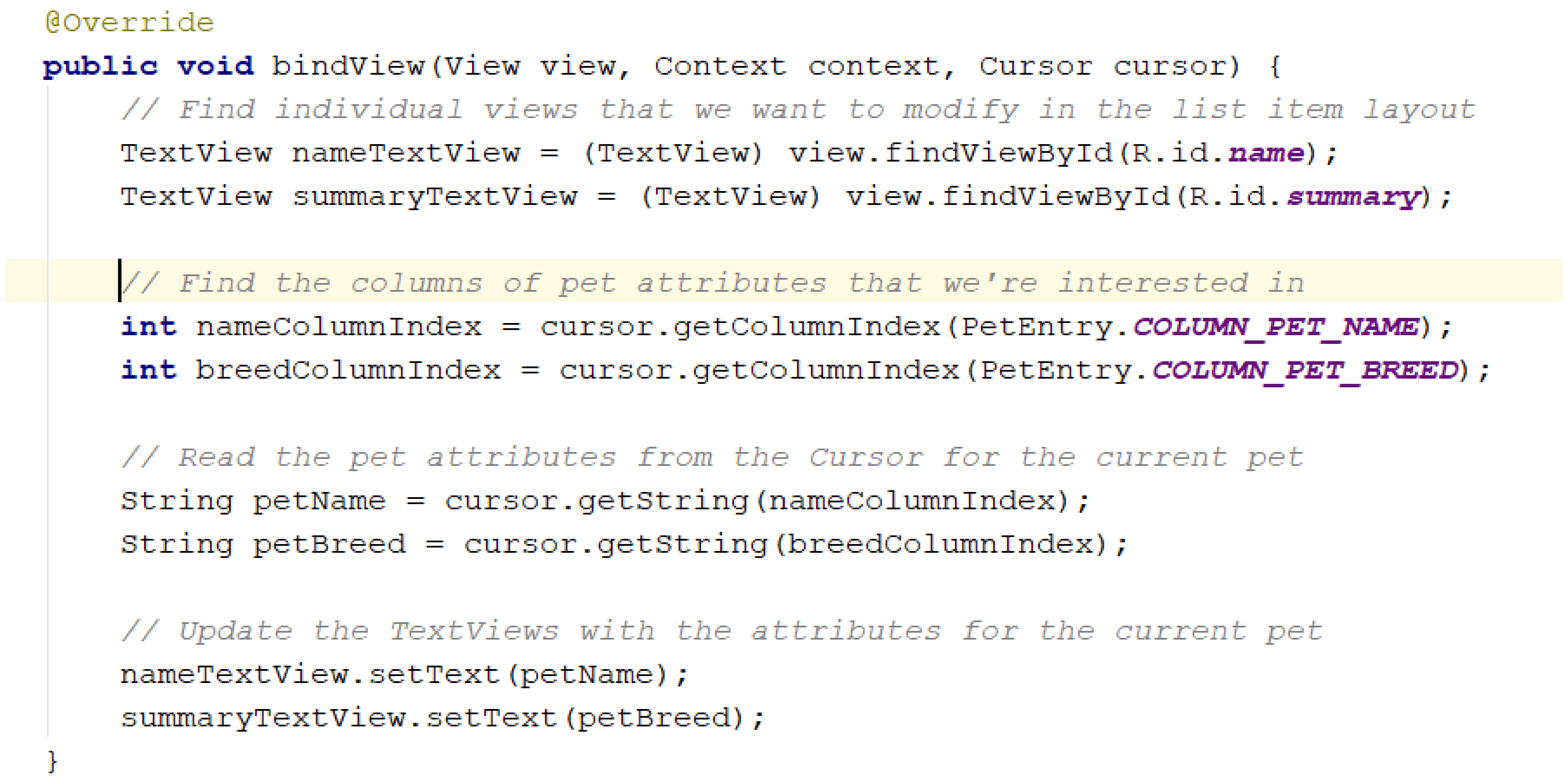
- ☐ 1. Create new layout file res/layout/list_item.xml for the list item layout. Copy over the XML from the provided gist.
- ☐ 2. Create new file PetCursorAdapter.java and copy over starter code from provided gist. Fill in the newView() and bindView() methods of the PetCursorAdapter.
- ☐ 3. Modify activity_catalog.xml so the CatalogActivity contains a ListView, not a TextView.
- ☐ 4. In CatalogActivity.java, modify the displayDatabaseInfo() method to setup a PetCursorAdapter using the Cursor returned from the provider. Remember to set the adapter onto the ListView. Remove all code related to displaying pet information in the TextView.

- Download list_item.xml dan PetCursorAdapter.java from here:
- https://gist.github.com/udacityandroid/5dbc87cd80bc083bac706a4c8f806ba5
- Create list_item.xml, then PASTE the code from GIST
- Create PetCursorAdapter.java, then PASTE the code from GIST

Fill in the newView() method

```
@Override
public View newView(Context context, Cursor cursor, ViewGroup parent) {
    // Inflate a list item view using the layout specified in list_item.xml
    return LayoutInflater.from(context).inflate(R.layout.list_item, parent, false);
}
```

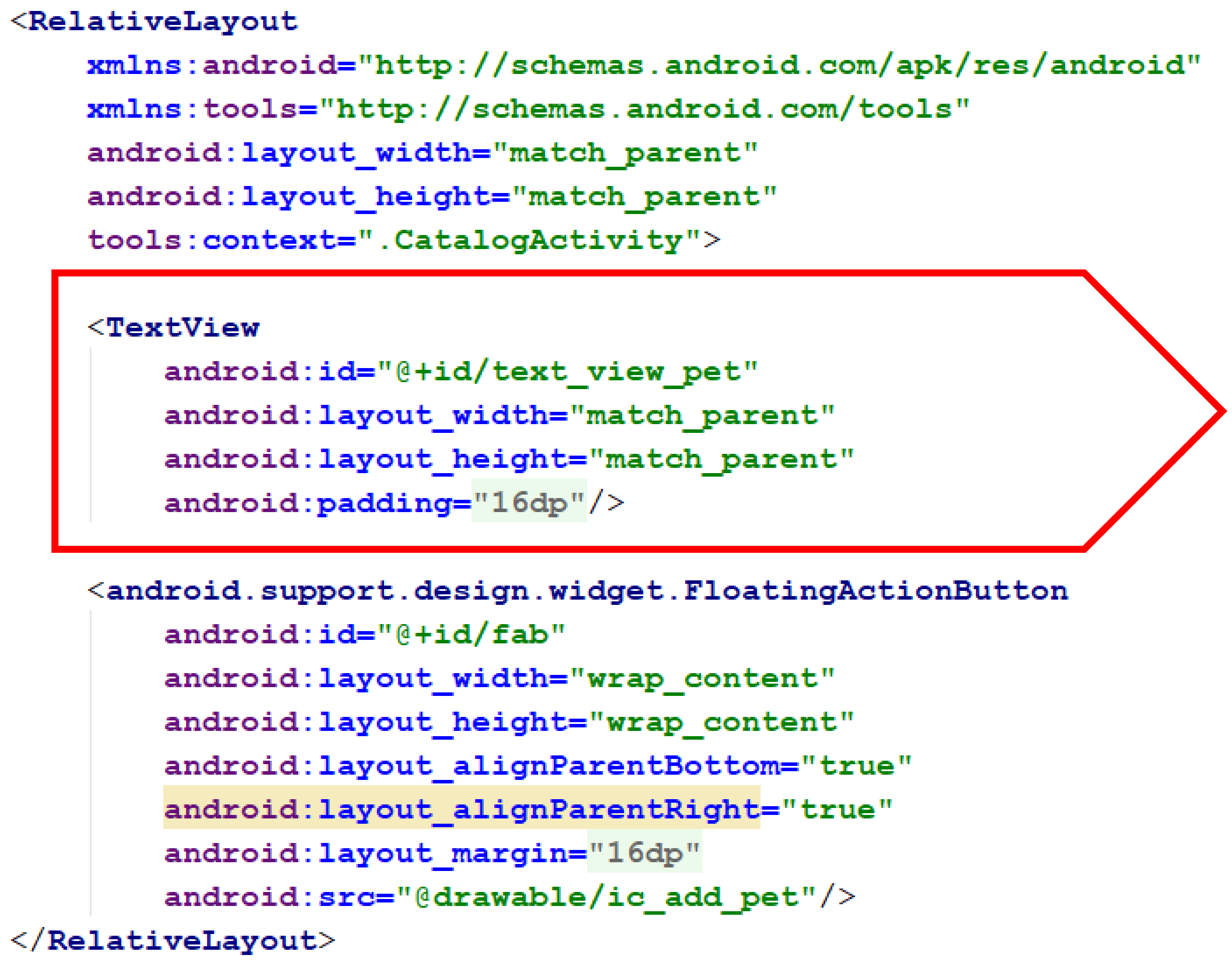
Fill in bindView() method



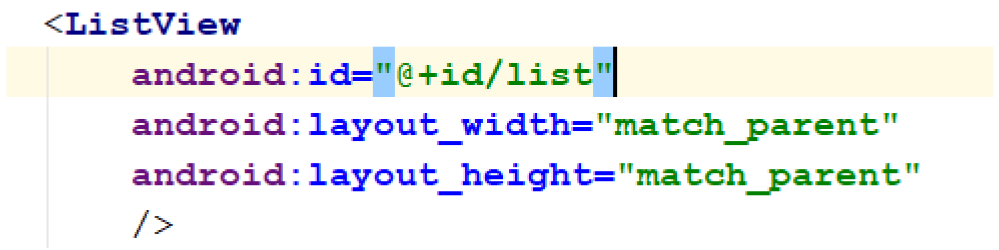
```java
@Override
public void bindView(View view, Context context, Cursor cursor) {
    // Find individual views that we want to modify in the list item layout
    TextView nameTextView = (TextView) view.findViewById(R.id.name);
    TextView summaryTextView = (TextView) view.findViewById(R.id.summary);

    // Find the columns of pet attributes that we're interested in
    int nameColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_NAME);
    int breedColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_BREED);

    // Read the pet attributes from the Cursor for the current pet
    String petName = cursor.getString(nameColumnIndex);
    String petBreed = cursor.getString(breedColumnIndex);

    // Update the TextViews with the attributes for the current pet
    nameTextView.setText(petName);
    summaryTextView.setText(petBreed);
}
```

Modify activity_catalog.xml

Modify TextView to ListView

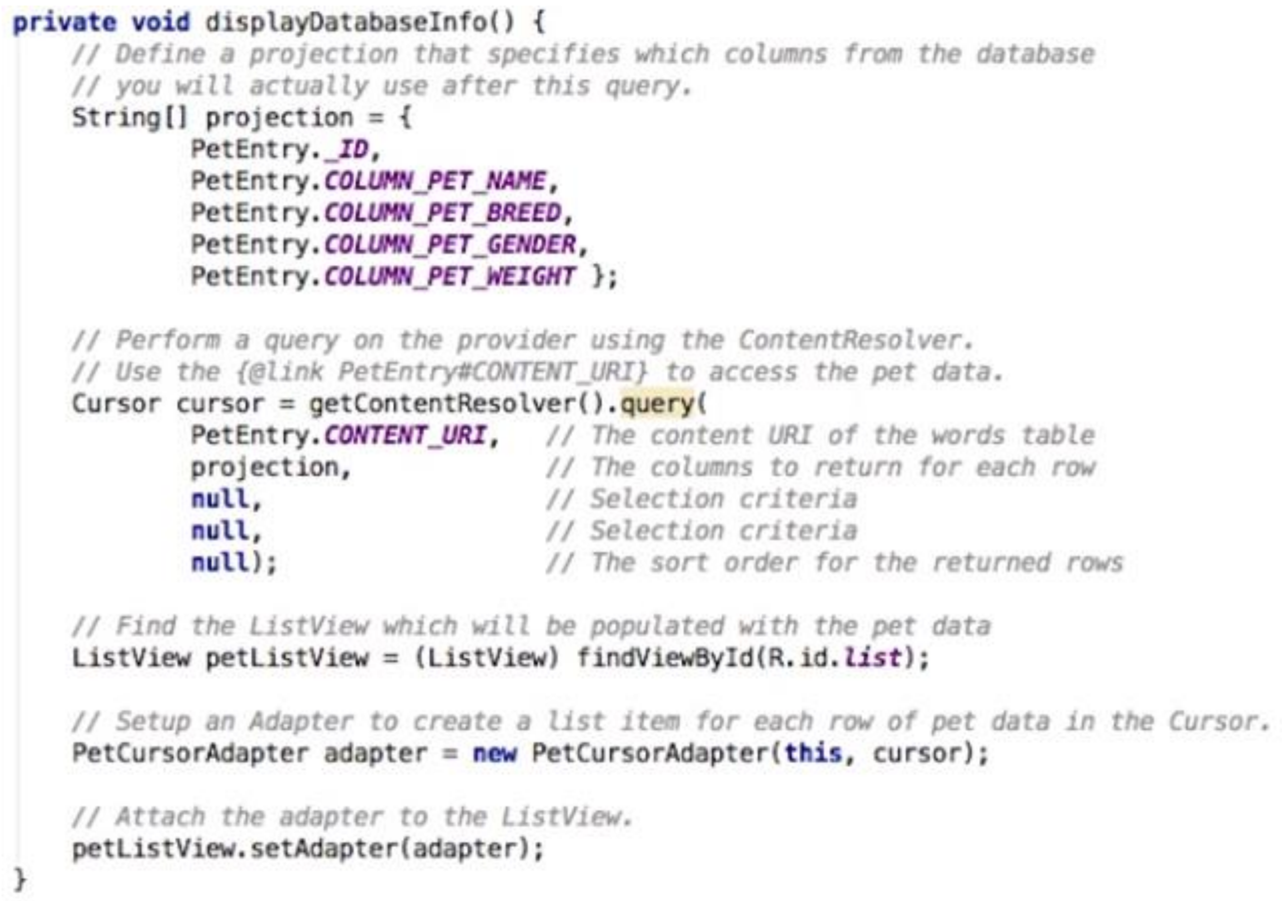


```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".CatalogActivity">

    <TextView
        android:id="@+id/text_view_pet"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="16dp"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_margin="16dp"
        android:src="@drawable/ic_add_pet"/>
</RelativeLayout>
```



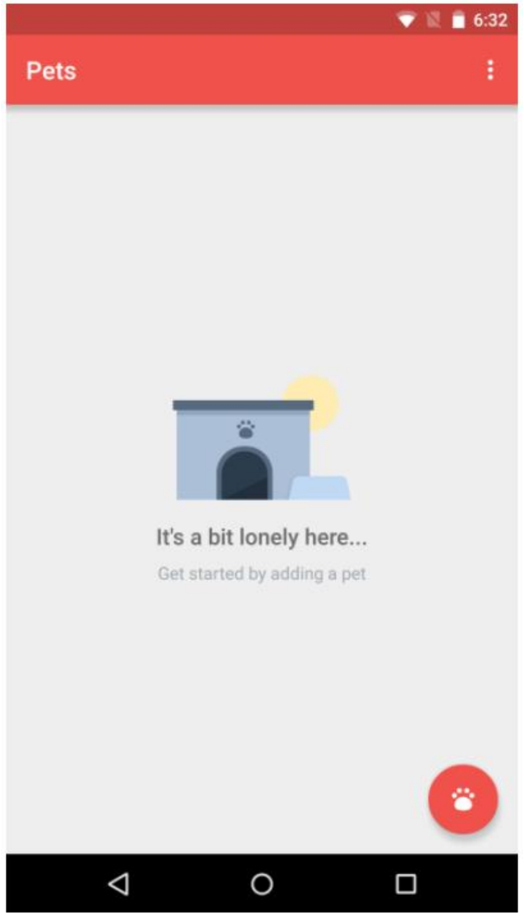
```
<ListView
    android:id="@+id/list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

In CatalogActivity.java
Modify displayDatabaseInfo ()



```java
private void displayDatabaseInfo() {
    // Define a projection that specifies which columns from the database
    // you will actually use after this query.
    String[] projection = {
            PetEntry._ID,
            PetEntry.COLUMN_PET_NAME,
            PetEntry.COLUMN_PET_BREED,
            PetEntry.COLUMN_PET_GENDER,
            PetEntry.COLUMN_PET_WEIGHT };

    // Perform a query on the provider using the ContentResolver.
    // Use the {@link PetEntry#CONTENT_URI} to access the pet data.
    Cursor cursor = getContentResolver().query(
            PetEntry.CONTENT_URI,   // The content URI of the words table
            projection,             // The columns to return for each row
            null,                   // Selection criteria
            null,                   // Selection criteria
            null);                  // The sort order for the returned rows

    // Find the ListView which will be populated with the pet data
    ListView petListView = (ListView) findViewById(R.id.list);

    // Setup an Adapter to create a list item for each row of pet data in the Cursor.
    PetCursorAdapter adapter = new PetCursorAdapter(this, cursor);

    // Attach the adapter to the ListView.
    petListView.setAdapter(adapter);
}
```

18 - Add empty view to the ListView



It's a bit lonely here...

Get started by adding a pet

Add Empty View to activity_catalog.xml (next to <ListView>)

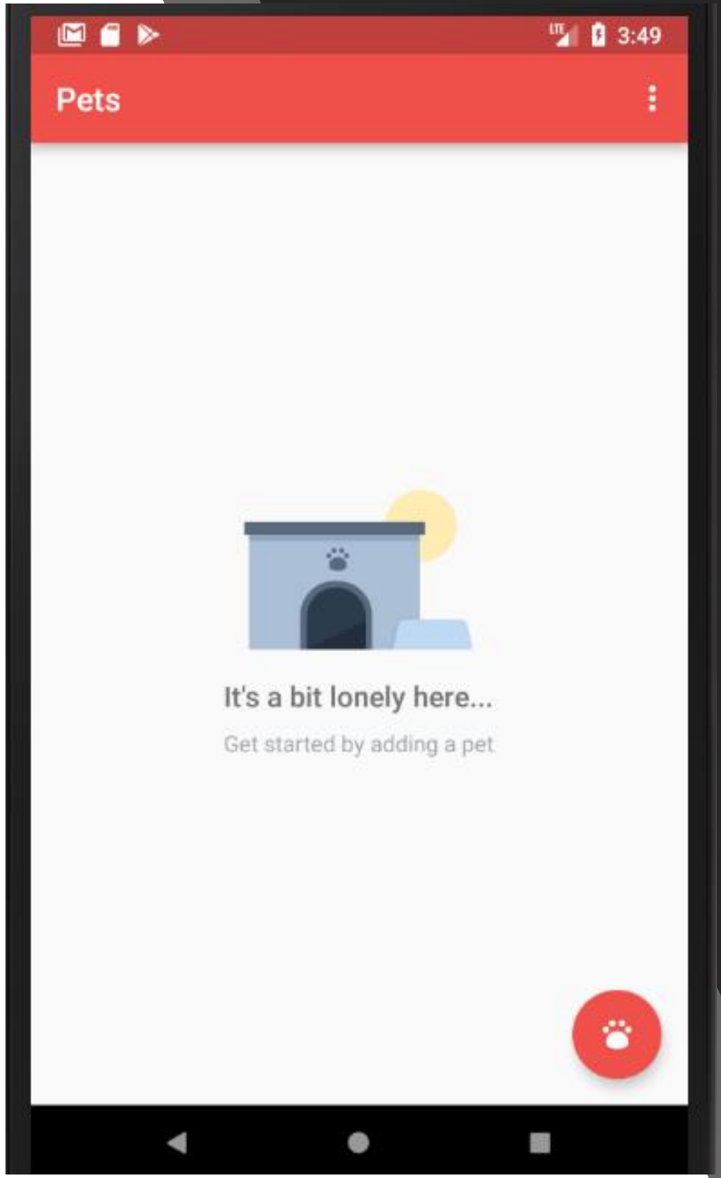
```xml
<ListView
    android:id="@+id/list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

<!-- Empty view for the list -->
<RelativeLayout
    android:id="@+id/empty_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true">

    <ImageView
        android:id="@+id/empty_shelter_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:src="@drawable/ic_empty_shelter"/>

    <TextView
        android:id="@+id/empty_title_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/empty_shelter_image"
        android:layout_centerHorizontal="true"
        android:fontFamily="sans-serif-medium"
        android:paddingTop="16dp"
        android:text="@string/empty_view_title_text"
        android:textAppearance="?android:textAppearanceMedium"/>

    <TextView
        android:id="@+id/empty_subtitle_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/empty_title_text"
        android:layout_centerHorizontal="true"
        android:fontFamily="sans-serif"
        android:paddingTop="8dp"
        android:text="@string/empty_view_subtitle_text"
        android:textAppearance="?android:textAppearanceSmall"
        android:textColor="#A2AAB0"/>
</RelativeLayout>
```

Modify string.xml

```
<!-- Title text for the empty view, which describes the empty dog house image [CHAR LIMIT=50] -->
<string name="empty_view_title_text">It\'s a bit lonely here...</string>

<!-- Subtitle text for the empty view that prompts the user to add a pet [CHAR LIMIT=50] -->
<string name="empty_view_subtitle_text">Get started by adding a pet</string>
```

Modify CatalogActivity.java onClick() method

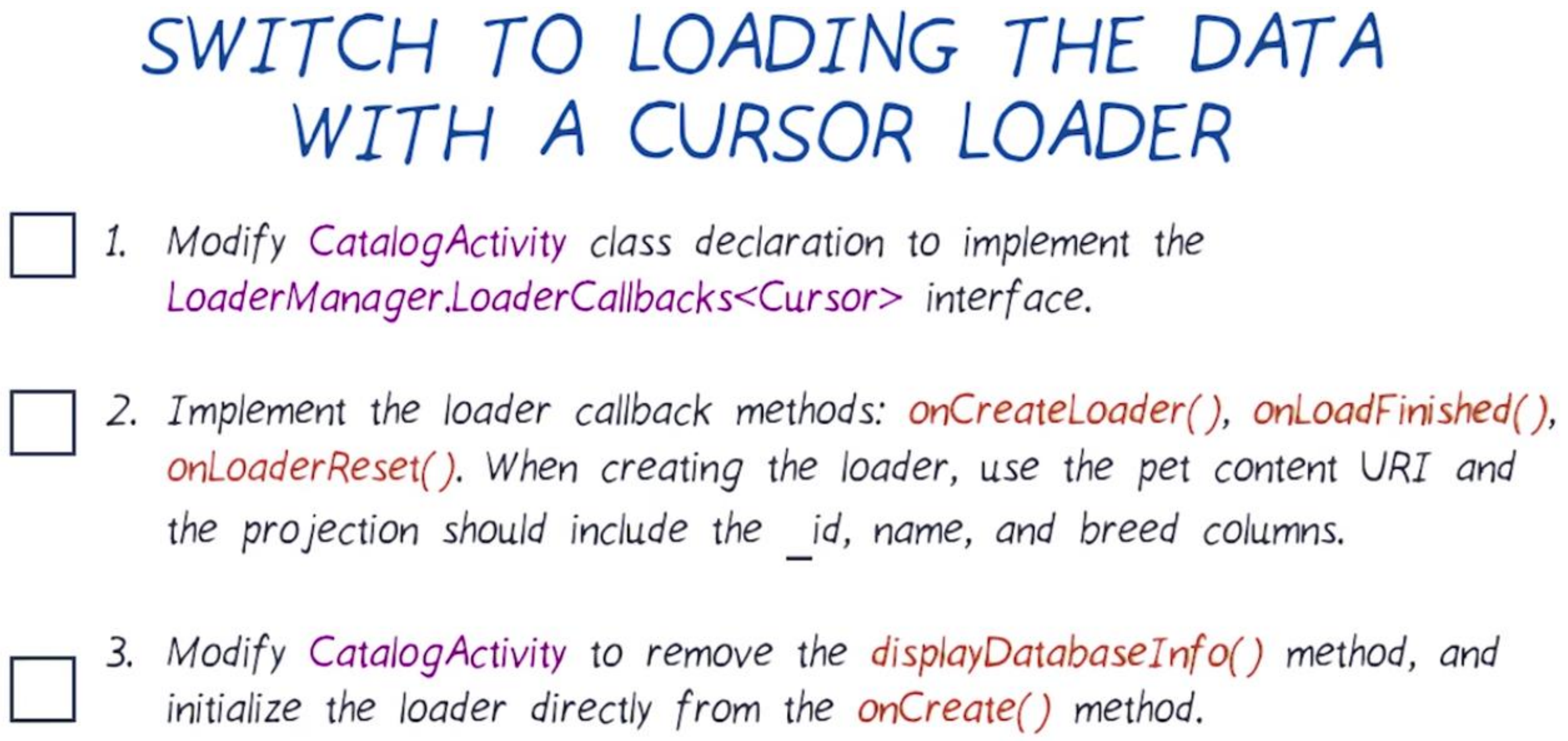
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_catalog);

    // Setup FAB to open EditorActivity
    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(CatalogActivity.this, EditorActivity.class);
            startActivity(intent);
        }
    });

    // Find the ListView which will be populated with the pet data
    ListView petListView = (ListView) findViewById(R.id.list);

    // Find and set empty view on the ListView, so that it only shows when the list has 0 items.
    View emptyView = findViewById(R.id.empty_view);
    petListView.setEmptyView(emptyView);
}
```



Test it !

- Delete the data or uninstall and reinstall the app to start with an empty database. You should now find that the empty view appears on screen!

19 - Switch to CursorLoader (6)



SWITCH TO LOADING THE DATA WITH A CURSOR LOADER

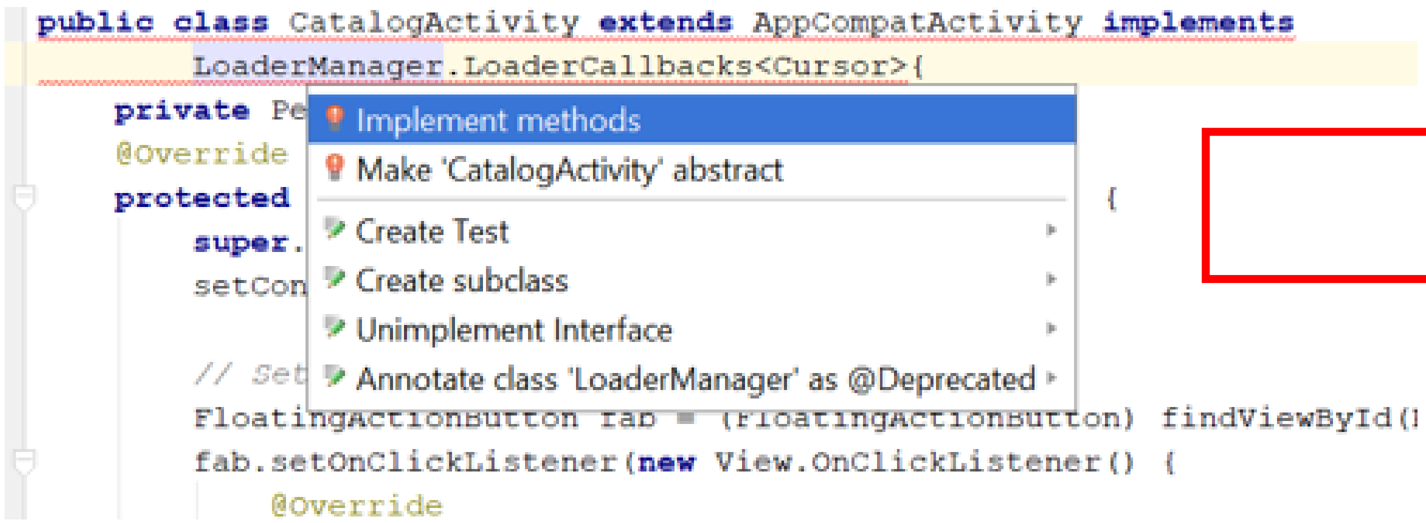
- ☐ 1. Modify CatalogActivity class declaration to implement the LoaderManager.LoaderCallbacks<Cursor> interface.
- ☐ 2. Implement the loader callback methods: onCreateLoader(), onLoadFinished(), onLoaderReset(). When creating the loader, use the pet content URI and the projection should include the _id, name, and breed columns.
- ☐ 3. Modify CatalogActivity to remove the displayDatabaseInfo() method, and initialize the loader directly from the onCreate() method.

Modify CatalogActivity Class Declaration

```
public class CatalogActivity extends AppCompatActivity implements
        LoaderManager.LoaderCallbacks<Cursor> {

    /** Identifier for the pet data loader */
    private static final int PET_LOADER = 0;

    /** Adapter for the ListView */
    PetCursorAdapter mCursorAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

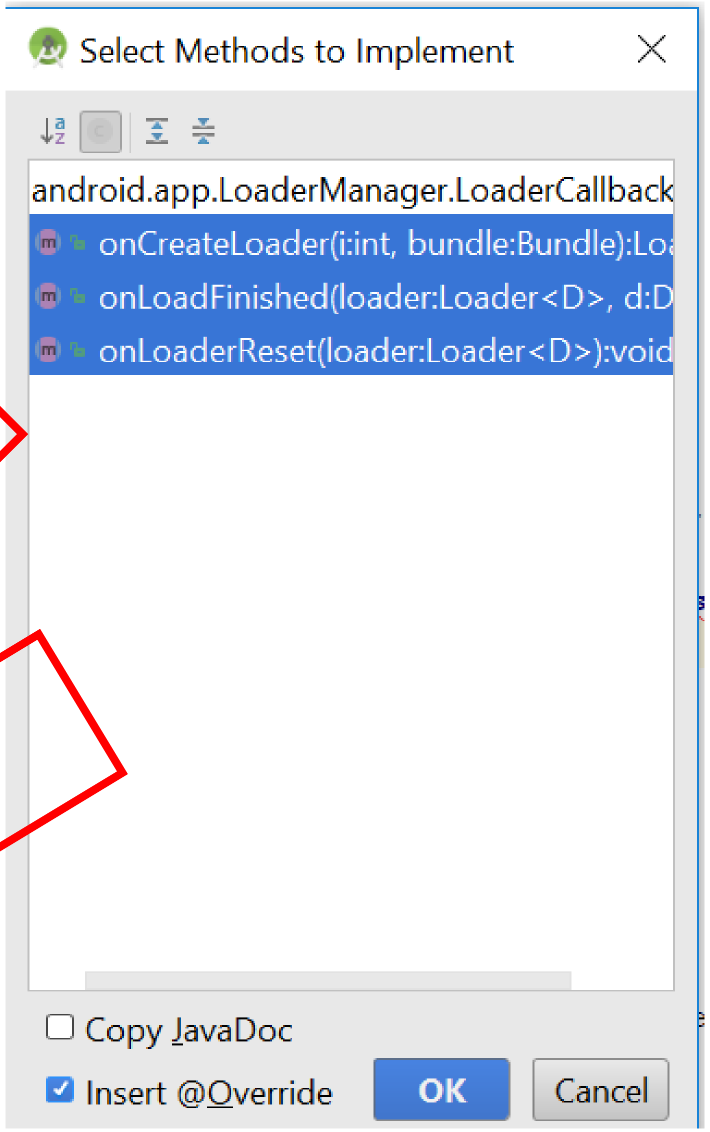
Implement onCreateLoader(), onLoadFinished(), and onLoaderReset

- Click on LoaderManager.LoaderCallbacks<cursor>
- Press Alt + Enter and OK



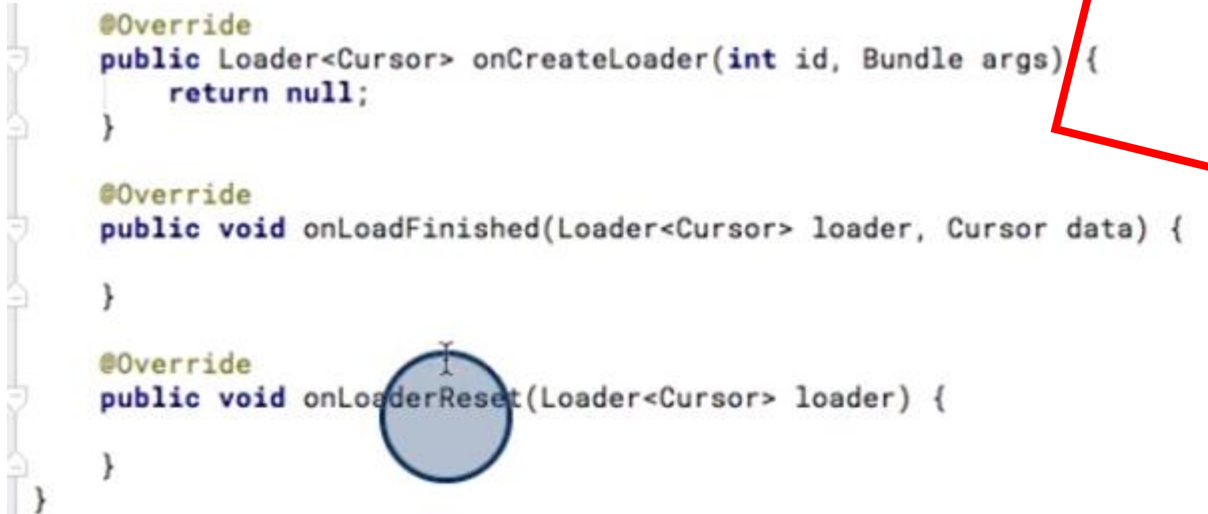
```
public class CatalogActivity extends AppCompatActivity implements
        LoaderManager.LoaderCallbacks<Cursor>{
    private Pe
    @Override
    protected                                              {
        super.
        setCon

        // Set
        FloatingActionButton fab = (FloatingActionButton) findViewById(!
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
```

Implement methods
Make 'CatalogActivity' abstract
Create Test
Create subclass
Unimplement Interface
Annotate class 'LoaderManager' as @Deprecated



```
android.app.LoaderManager.LoaderCallback
    onCreateLoader(i:int, bundle:Bundle):Lo
    onLoadFinished(loader:Loader<D>, d:D
    onLoaderReset(loader:Loader<D>):void
```

☐ Copy JavaDoc
☑ Insert @Override



```
    @Override
    public Loader<Cursor> onCreateLoader(int id, Bundle args) {
        return null;
    }

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor data) {

    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {

    }
}
```

Fill onCreateLoader()

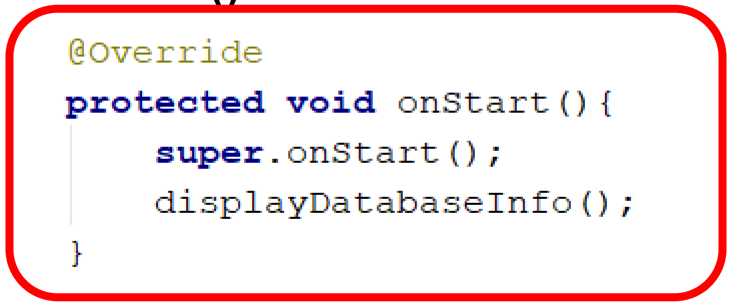
```java
@Override
public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
    // Define a projection that specifies the columns from the table we care about.
    String[] projection = {
            PetEntry._ID,
            PetEntry.COLUMN_PET_NAME,
            PetEntry.COLUMN_PET_BREED };

    // This loader will execute the ContentProvider's query method on a background thread
    return new CursorLoader(this,   // Parent activity context
            PetEntry.CONTENT_URI,   // Provider content URI to query
            projection,             // Columns to include in the resulting Cursor
            null,                   // No selection clause
            null,                   // No selection arguments
            null);                  // Default sort order
}
```

Fill onLoadFinished() and onLoaderReset()

```java
@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
    // Update {@link PetCursorAdapter} with this new cursor containing updated pet data
    mCursorAdapter.swapCursor(data);
}

@Override
public void onLoaderReset(Loader<Cursor> loader) {
    // Callback called when the data needs to be deleted
    mCursorAdapter.swapCursor(null);
}
```

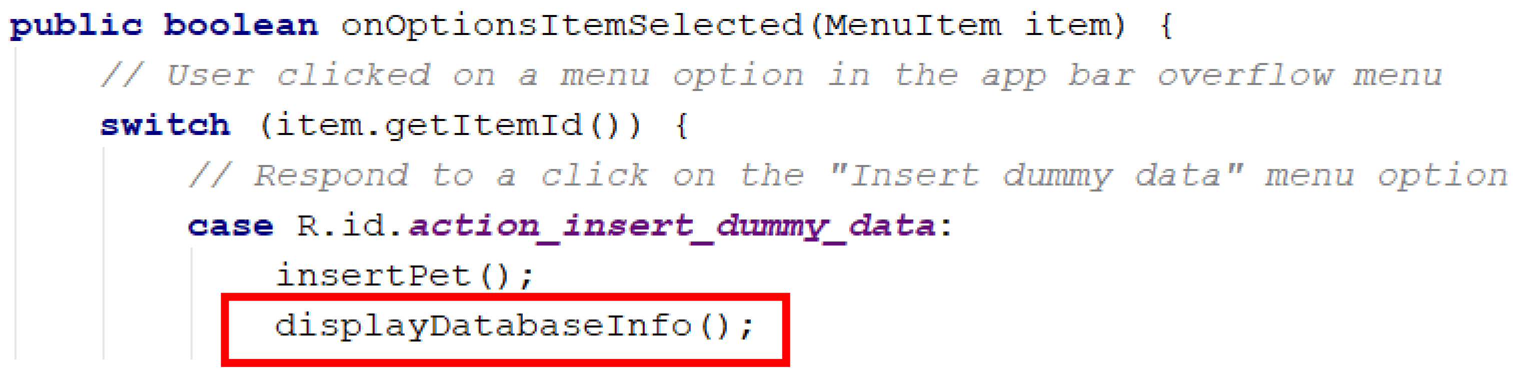
Delete displayDatabaseInfo() method

- Also delete any references to that method
  - void onStart()



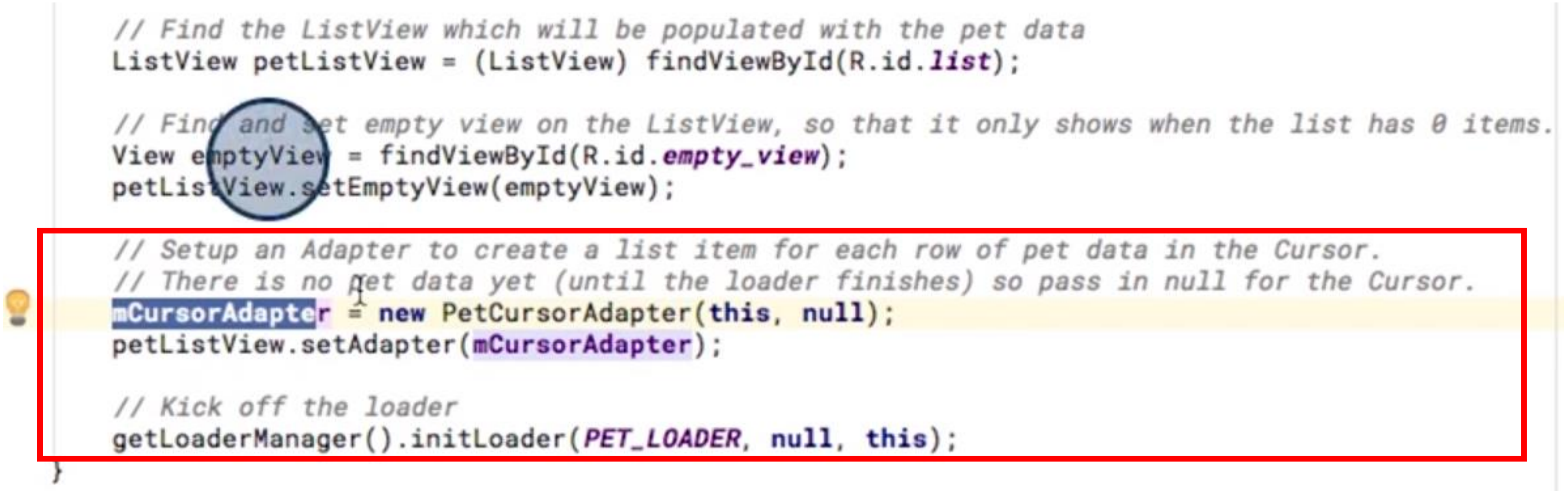
```
@Override
protected void onStart(){
    super.onStart();
    displayDatabaseInfo();
}
```

  - onOptionItemsSelected()

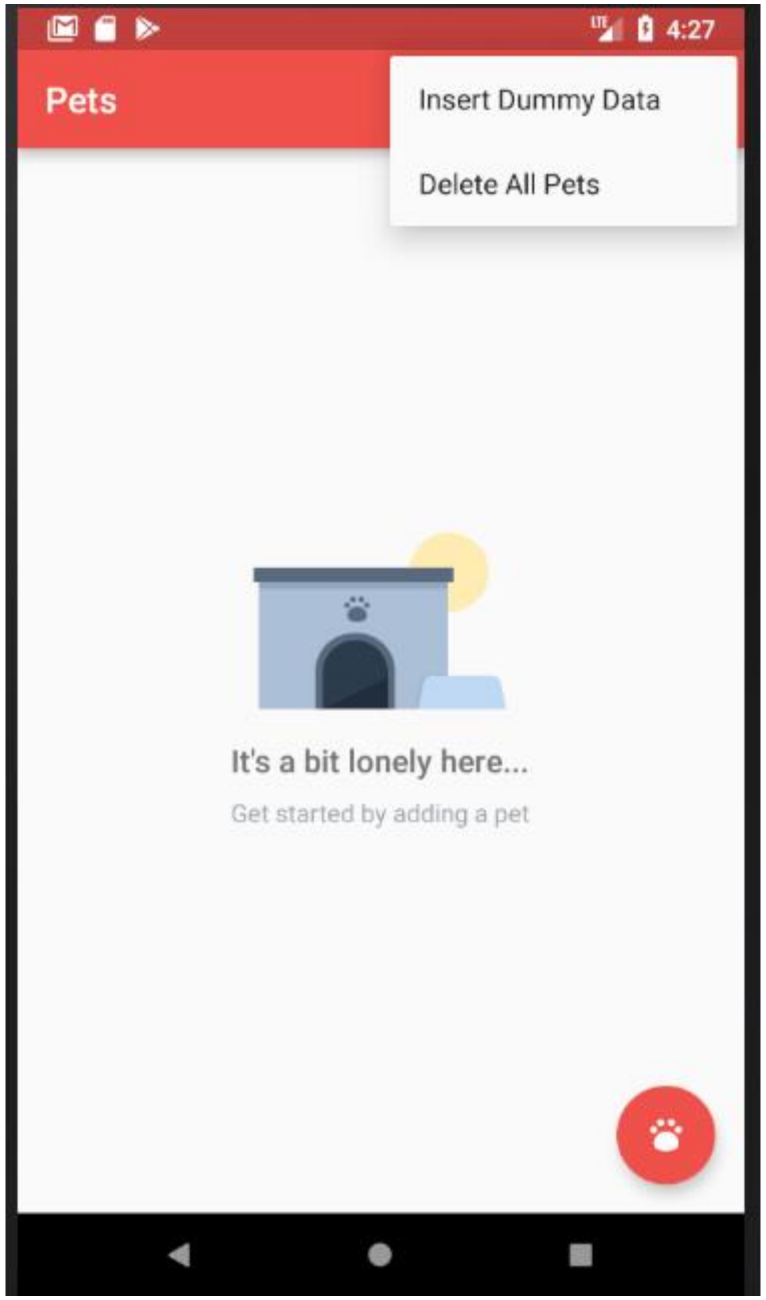


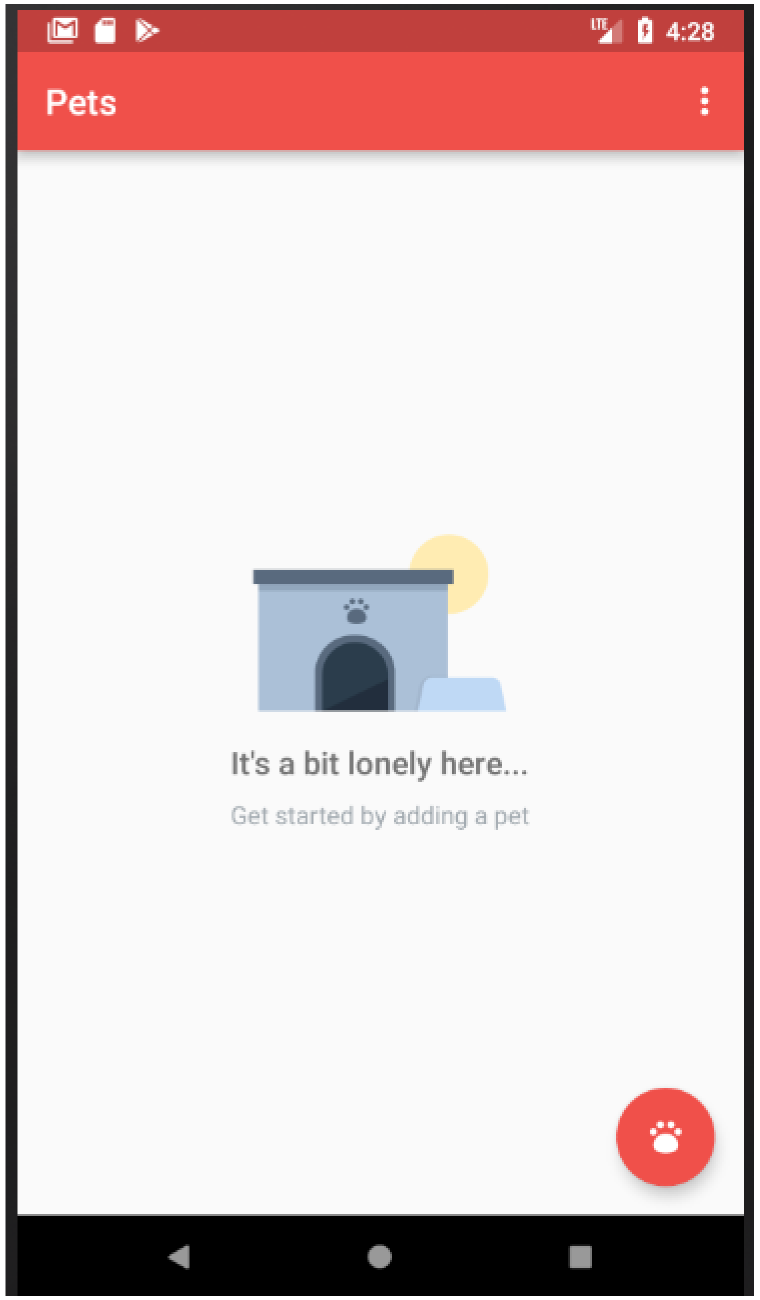
```
public boolean onOptionsItemSelected(MenuItem item) {
    // User clicked on a menu option in the app bar overflow menu
    switch (item.getItemId()) {
        // Respond to a click on the "Insert dummy data" menu option
        case R.id.action_insert_dummy_data:
            insertPet();
            displayDatabaseInfo();
```

CatalogActivity.java – onCreate()



```
    // Find the ListView which will be populated with the pet data
    ListView petListView = (ListView) findViewById(R.id.list);

    // Find and set empty view on the ListView, so that it only shows when the list has 0 items.
    View emptyView = findViewById(R.id.empty_view);
    petListView.setEmptyView(emptyView);

    // Setup an Adapter to create a list item for each row of pet data in the Cursor.
    // There is no pet data yet (until the loader finishes) so pass in null for the Cursor.
    mCursorAdapter = new PetCursorAdapter(this, null);
    petListView.setAdapter(mCursorAdapter);

    // Kick off the loader
    getLoaderManager().initLoader(PET_LOADER, null, this);
}
```





Run the app.

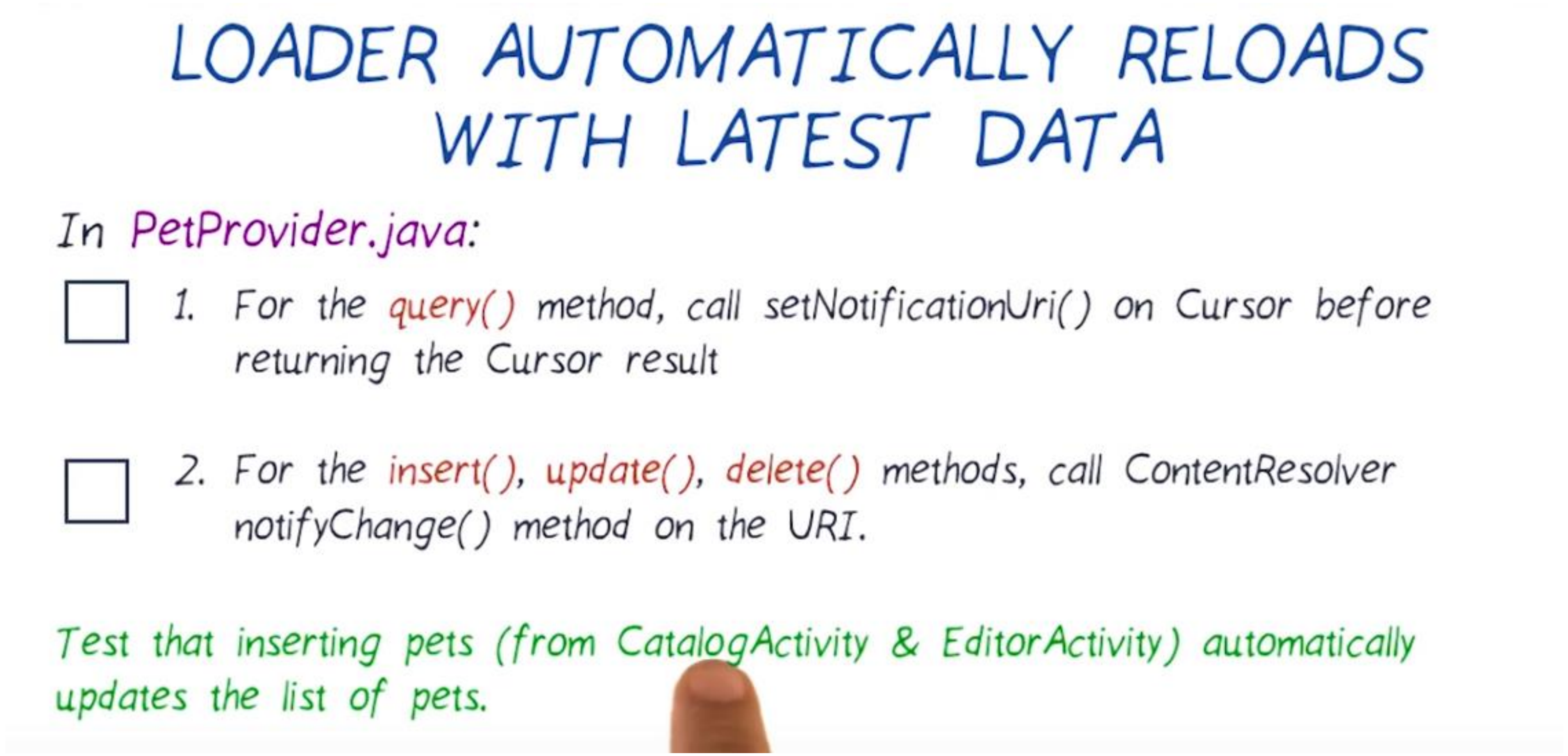
Try to insert Dummy Data

The Data will not shown automatically

You must to quit the app and restart it

Let's fix this on the next slide

20 - Modify ContentProvider so Loader refreshes data automatically



LOADER AUTOMATICALLY RELOADS WITH LATEST DATA

In PetProvider.java:

- ☐ 1. For the query() method, call setNotificationUri() on Cursor before returning the Cursor result
- ☐ 2. For the insert(), update(), delete() methods, call ContentResolver notifyChange() method on the URI.

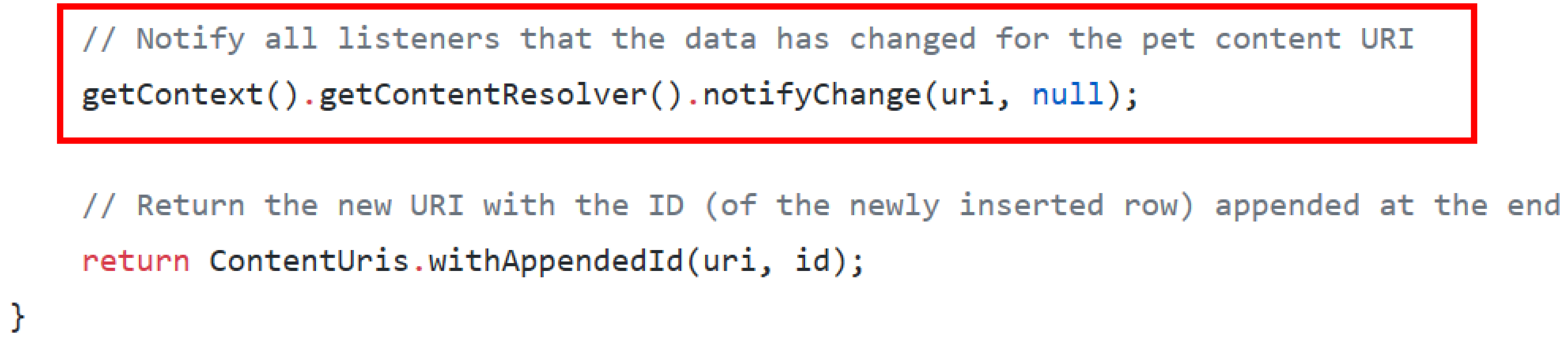
Test that inserting pets (from CatalogActivity & EditorActivity) automatically updates the list of pets.

PetProvider.java – **public Cursor query()** method

- Add this code before return cursor;

```
        default:
            throw new IllegalArgumentException("Cannot query unknown URI " + uri);
    }

    // Set notification URI on the Cursor,
    // so we know what content URI the Cursor was created for.
    // If the data at this URI changes, then we know we need to update the Cursor.
    cursor.setNotificationUri(getContext().getContentResolver(), uri);

    // Return the cursor
    return cursor;
}
```

PetProvider.java – Uri insertPet()



```
    // Notify all listeners that the data has changed for the pet content URI
    getContext().getContentResolver().notifyChange(uri, null);

    // Return the new URI with the ID (of the newly inserted row) appended at the end
    return ContentUris.withAppendedId(uri, id);
}
```

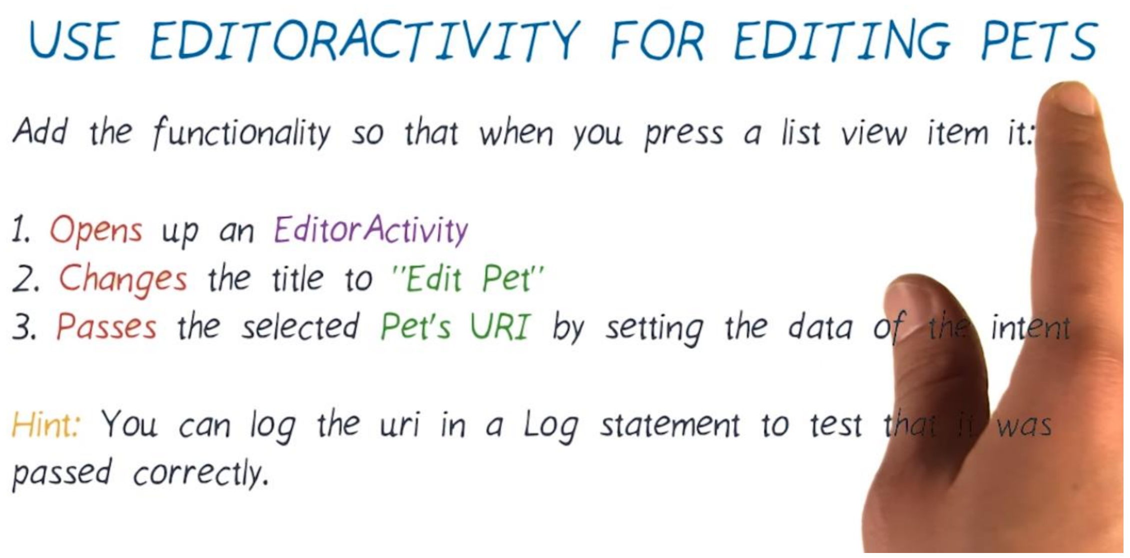
Run the app, try to insert dummy data or insert in editor activity.
The data will automatically refreshed

PetProvider.java – Uri updatePet()

```java
        // Otherwise, get writeable database to update the data
        SQLiteDatabase database = mDbHelper.getWritableDatabase();

        // Perform the update on the database and get the number of rows affected
        int rowsUpdated = database.update(PetEntry.TABLE_NAME, values, selection, selectionArgs);

        // If 1 or more rows were updated, then notify all listeners that the data at the
        // given URI has changed
        if (rowsUpdated != 0) {
            getContext().getContentResolver().notifyChange(uri, null);
        }

        // Return the number of rows updated
        return rowsUpdated;
    }
```

PetProvider.java – Uri delete()

```
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    // Get writeable database
    SQLiteDatabase database = mDbHelper.getWritableDatabase();

    // Track the number of rows that were deleted
    int rowsDeleted;

    final int match = sUriMatcher.match(uri);
    switch (match) {
        case PETS:
            // Delete all rows that match the selection and selection args
            rowsDeleted = database.delete(PetEntry.TABLE_NAME, selection, selectionArgs);
            break;
            case PET_ID:
                // Delete a single row given by the ID in the URI
                selection = PetEntry._ID + "=?";
                selectionArgs = new String[] { String.valueOf(ContentUris.parseId(uri)) };
                rowsDeleted = database.delete(PetEntry.TABLE_NAME, selection, selectionArgs);
                break;
            default:
                throw new IllegalArgumentException("Deletion is not supported for " + uri);
        }

        // If 1 or more rows were deleted, then notify all listeners that the data at the
        // given URI has changed
        if (rowsDeleted != 0) {
            getContext().getContentResolver().notifyChange(uri, null);
        }

        // Return the number of rows deleted
        return rowsDeleted;
    }
```

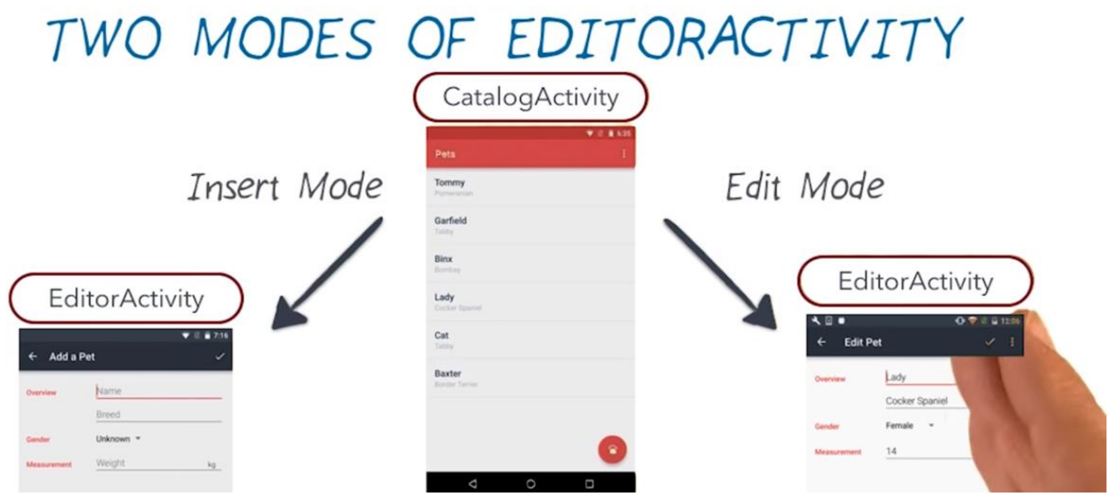


USE EDITORACTIVITY FOR EDITING PETS

Add the functionality so that when you press a list view item it:

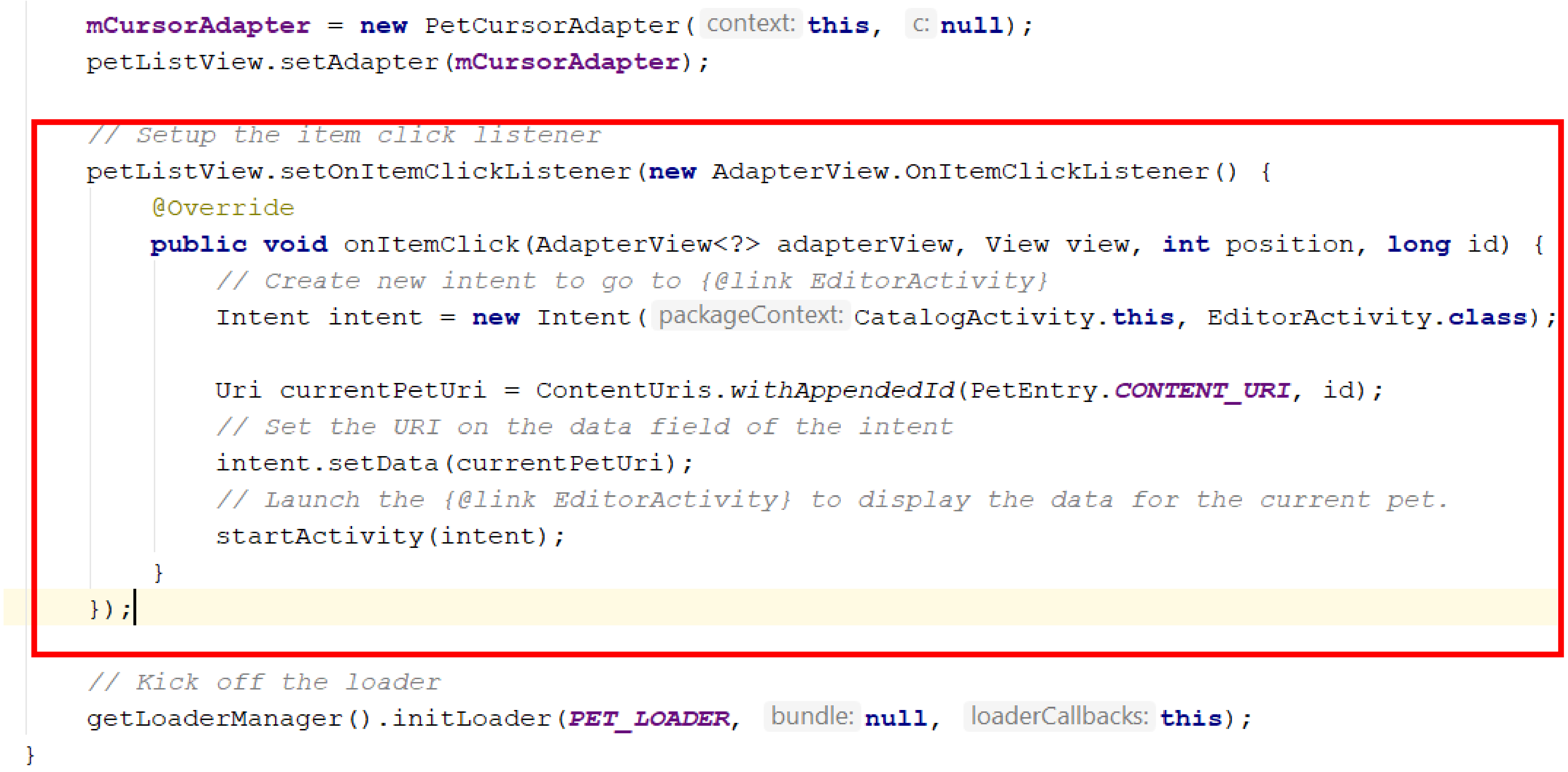
1. Opens up an EditorActivity
2. Changes the title to "Edit Pet"
3. Passes the selected Pet's URI by setting the data of the intent

Hint: You can log the uri in a Log statement to test that it was passed correctly.



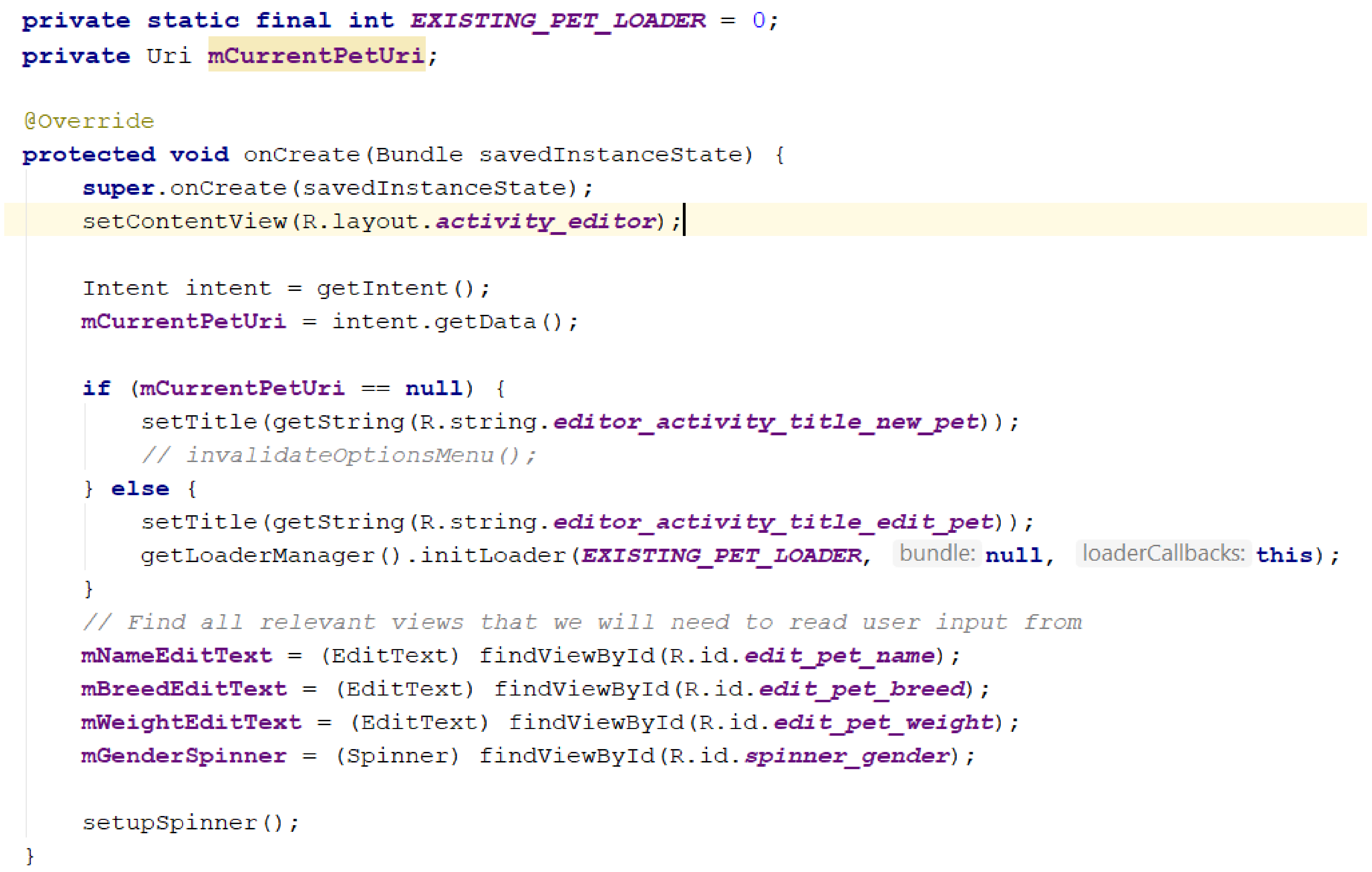
21 - Clicking on list item opens EditorActivity

CatalogActivity – onCreate()



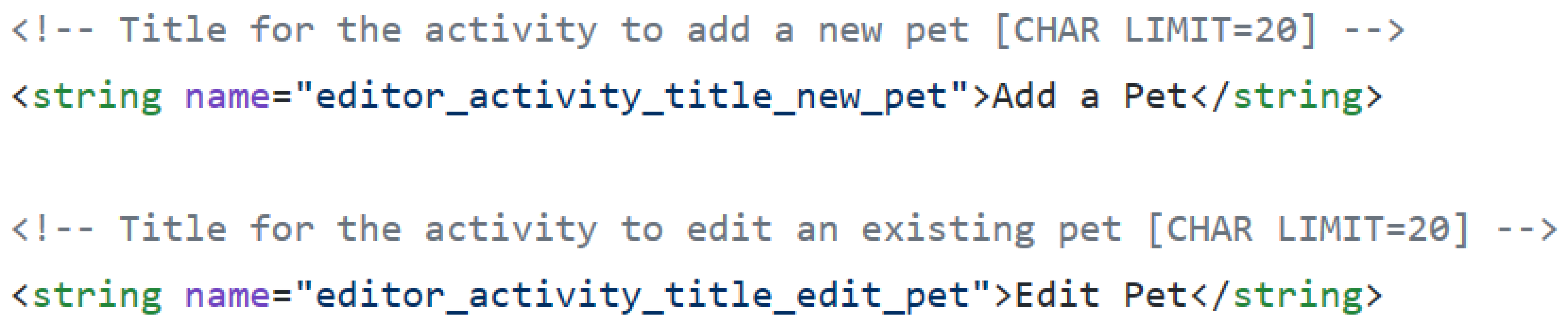
```
    mCursorAdapter = new PetCursorAdapter( context: this,  c: null);
    petListView.setAdapter(mCursorAdapter);

    // Setup the item click listener
    petListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {
            // Create new intent to go to {@link EditorActivity}
            Intent intent = new Intent( packageContext: CatalogActivity.this, EditorActivity.class);

            Uri currentPetUri = ContentUris.withAppendedId(PetEntry.CONTENT_URI, id);
            // Set the URI on the data field of the intent
            intent.setData(currentPetUri);
            // Launch the {@link EditorActivity} to display the data for the current pet.
            startActivity(intent);
        }
    });

    // Kick off the loader
    getLoaderManager().initLoader(PET_LOADER,  bundle: null,  loaderCallbacks: this);
}
```

EditorActivity.java – onCreate()



```java
private static final int EXISTING_PET_LOADER = 0;
private Uri mCurrentPetUri;

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

    Intent intent = getIntent();
    mCurrentPetUri = intent.getData();

    if (mCurrentPetUri == null) {
        setTitle(getString(R.string.editor_activity_title_new_pet));
        // invalidateOptionsMenu();
    } else {
        setTitle(getString(R.string.editor_activity_title_edit_pet));
        getLoaderManager().initLoader(EXISTING_PET_LOADER, bundle: null, loaderCallbacks: this);
    }
    // Find all relevant views that we will need to read user input from
    mNameEditText = (EditText) findViewById(R.id.edit_pet_name);
    mBreedEditText = (EditText) findViewById(R.id.edit_pet_breed);
    mWeightEditText = (EditText) findViewById(R.id.edit_pet_weight);
    mGenderSpinner = (Spinner) findViewById(R.id.spinner_gender);

    setupSpinner();
}
```

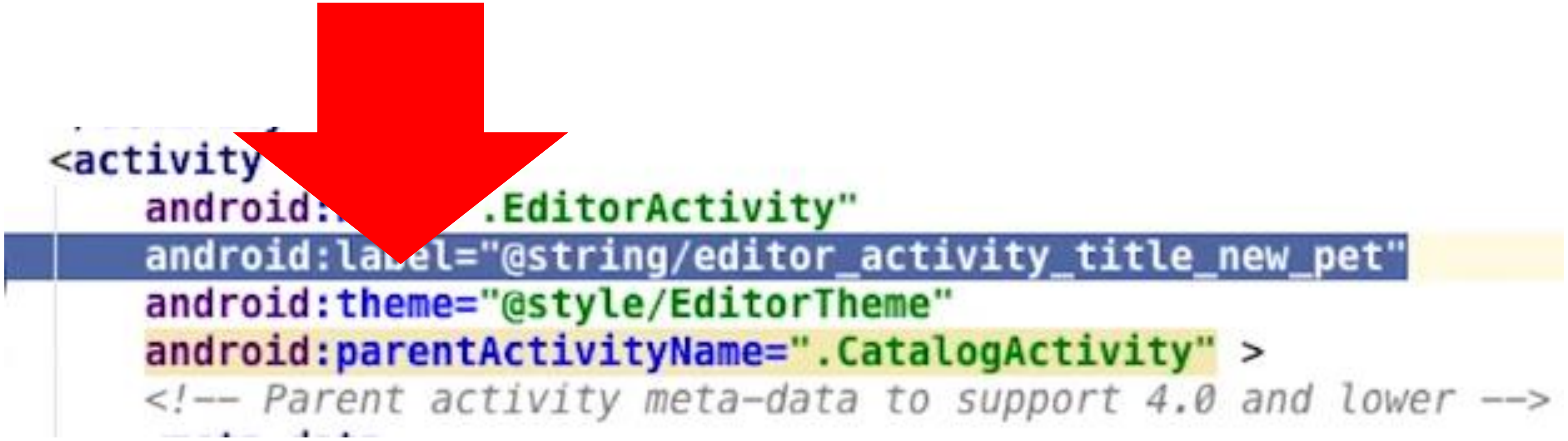
Add this to string.xml



```
<!-- Title for the activity to add a new pet [CHAR LIMIT=20] -->
<string name="editor_activity_title_new_pet">Add a Pet</string>

<!-- Title for the activity to edit an existing pet [CHAR LIMIT=20] -->
<string name="editor_activity_title_edit_pet">Edit Pet</string>
```

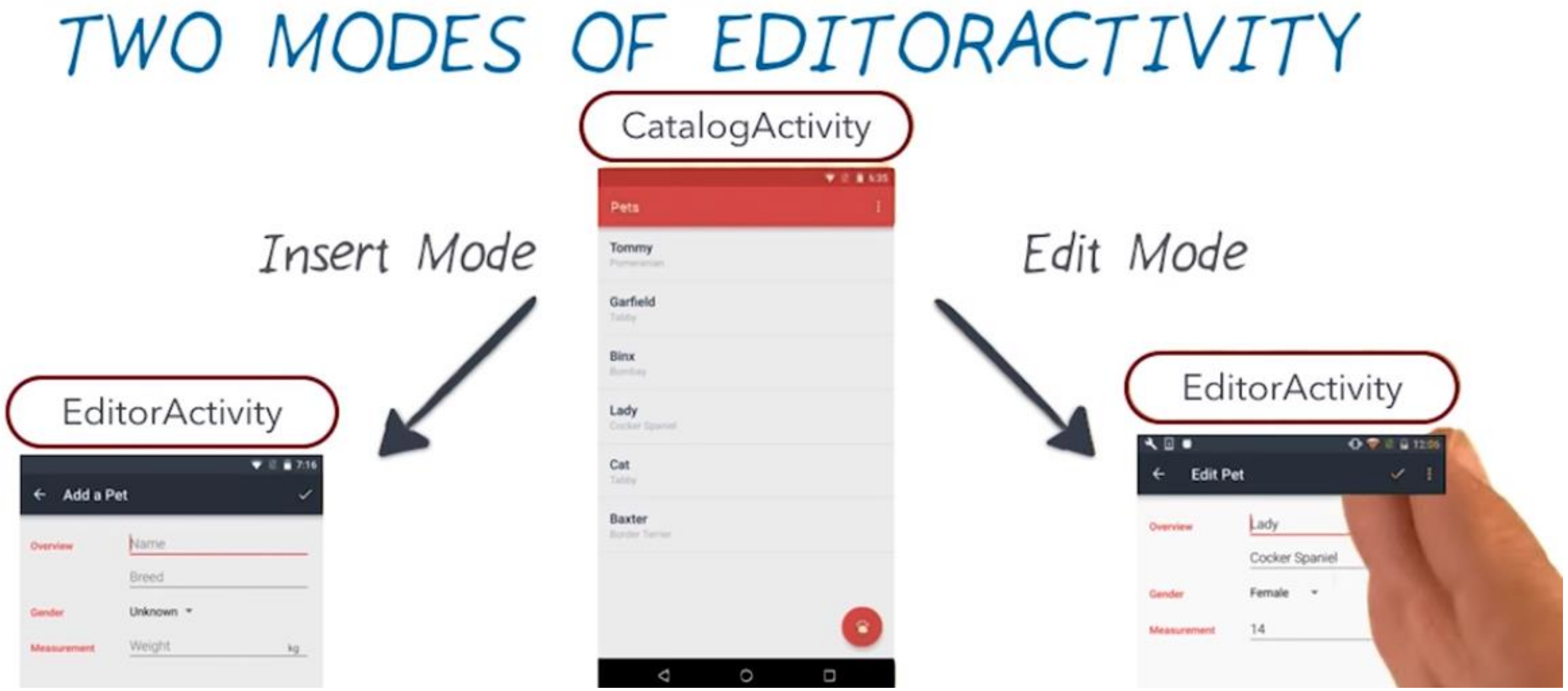
Modify Android Manifest

- Delete android:label for EditorActivity

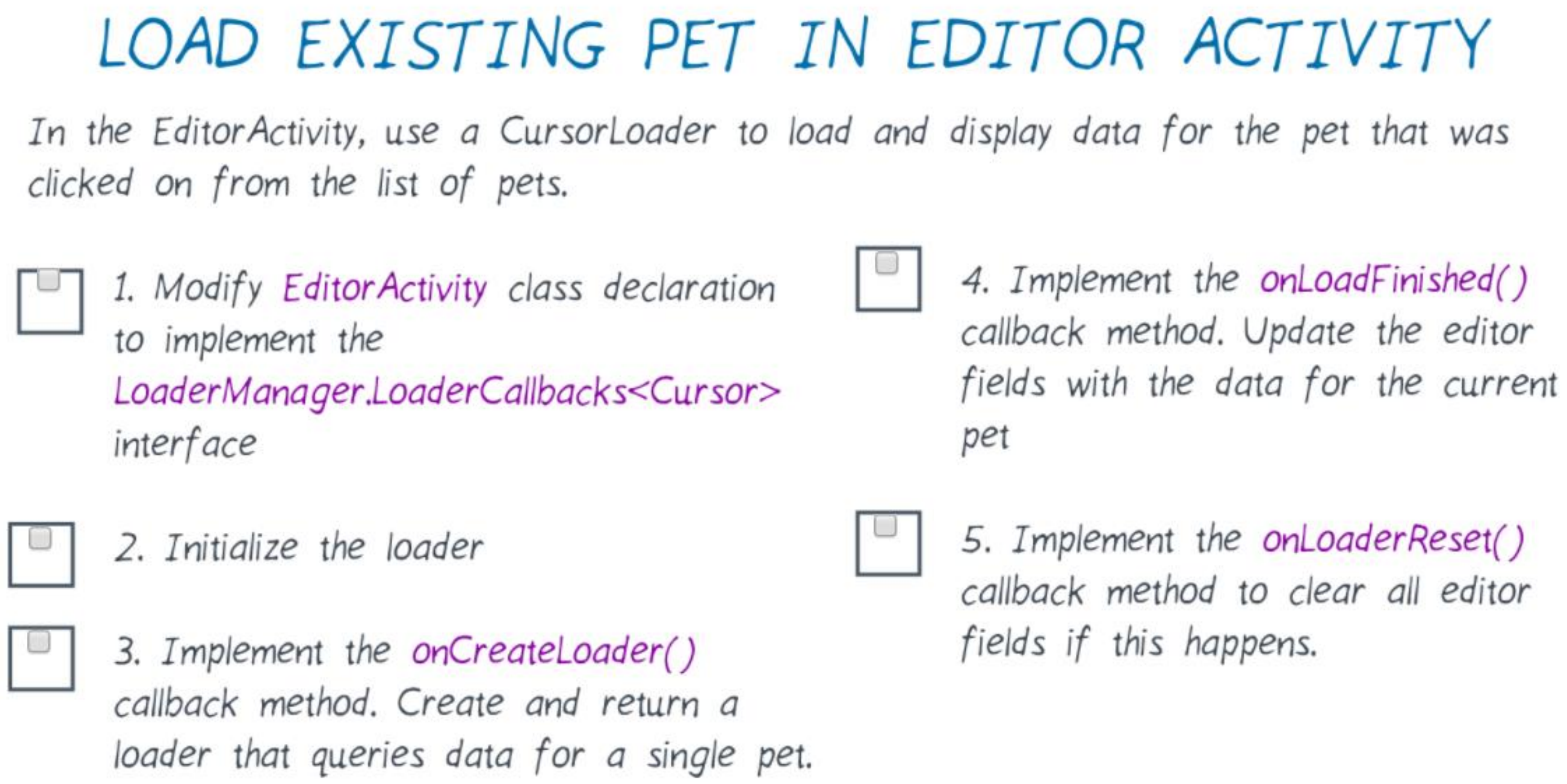


```
<activity
    android:        .EditorActivity"
    android:label="@string/editor_activity_title_new_pet"
    android:theme="@style/EditorTheme"
    android:parentActivityName=".CatalogActivity" >
    <!-- Parent activity meta-data to support 4.0 and lower -->
```

Run the App


TWO MODES OF EDITORACTIVITY
CatalogActivity
Insert Mode
Edit Mode
EditorActivity
EditorActivity
Pets
Tommy
Pomeranian
Garfield
Tabby
Binx
Bombay
Lady
Cocker Spaniel
Cat
Tabby
Baxter
Border Terrier
Add a Pet
Overview
Name
Breed
Gender
Unknown
Measurement
Weight
kg
Edit Pet
Overview
Lady
Cocker Spaniel
Gender
Female
Measurement
14

22 - Load existing pet data from database (10)



LOAD EXISTING PET IN EDITOR ACTIVITY

In the EditorActivity, use a CursorLoader to load and display data for the pet that was clicked on from the list of pets.

- [ ] 1. Modify EditorActivity class declaration to implement the LoaderManager.LoaderCallbacks<Cursor> interface
- [ ] 2. Initialize the loader
- [ ] 3. Implement the onCreateLoader() callback method. Create and return a loader that queries data for a single pet.
- [ ] 4. Implement the onLoadFinished() callback method. Update the editor fields with the data for the current pet
- [ ] 5. Implement the onLoaderReset() callback method to clear all editor fields if this happens.

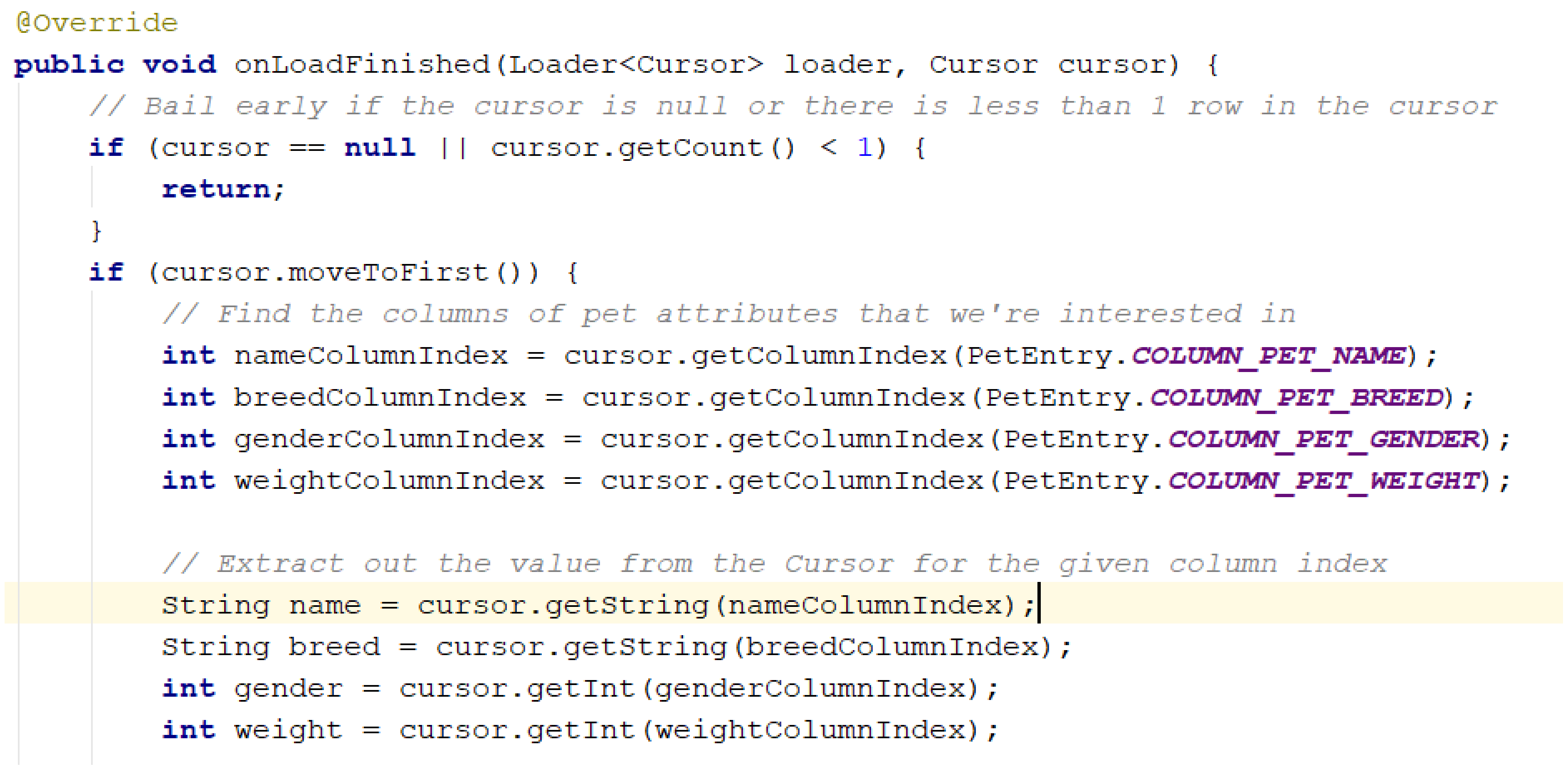
EditorActivity.java

```java
public class EditorActivity extends AppCompatActivity implements
        LoaderManager.LoaderCallbacks<Cursor> {

    /** Identifier for the pet data loader */
    private static final int EXISTING_PET_LOADER = 0;

    /** Content URI for the existing pet (null if it's a new pet) */
    private Uri mCurrentPetUri;
```

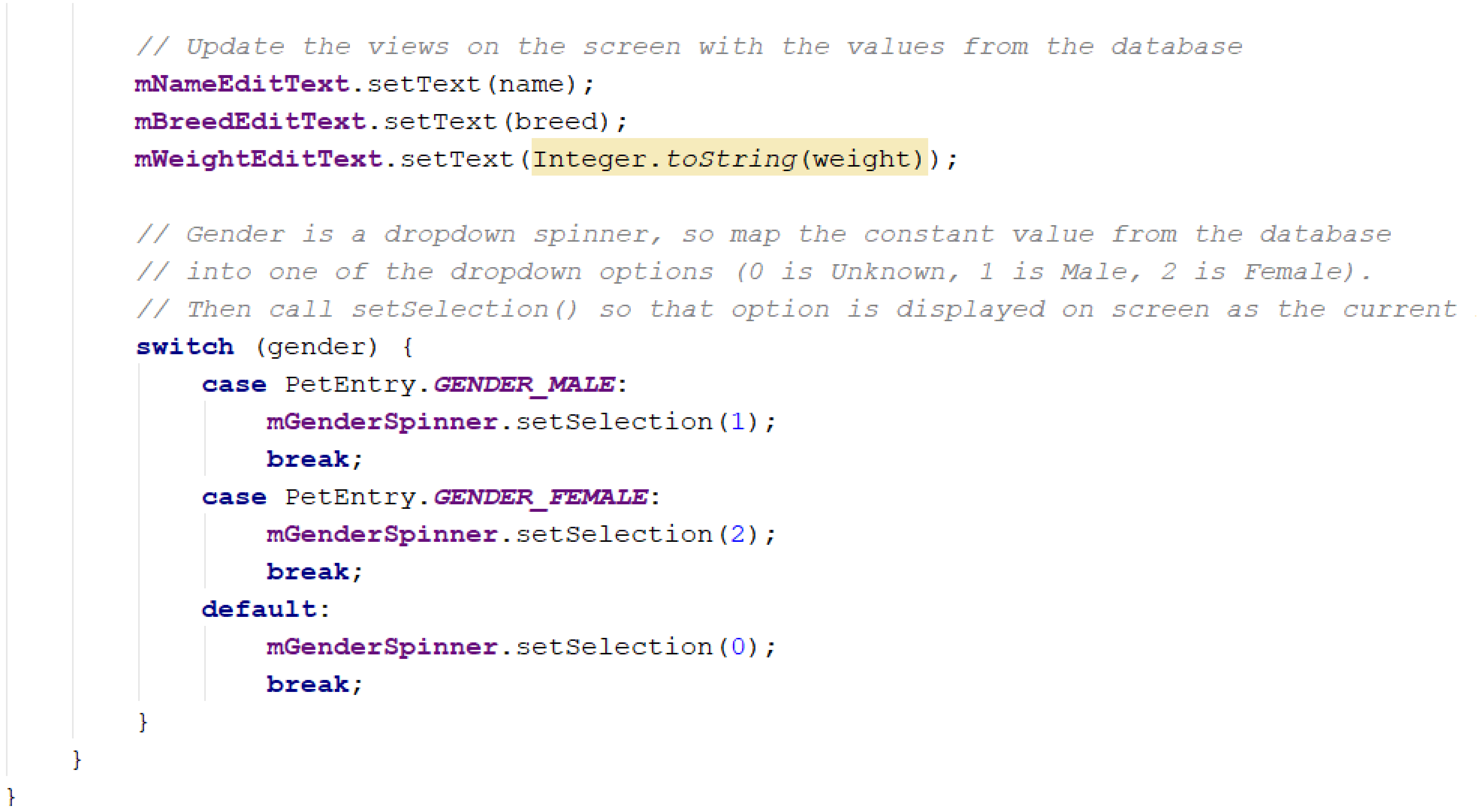
EditorActivity—onCreateLoader()

```java
    @Override
public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
    // Since the editor shows all pet attributes, define a projection that contains
    // all columns from the pet table
    String[] projection = {
            PetEntry._ID,
            PetEntry.COLUMN_PET_NAME,
            PetEntry.COLUMN_PET_BREED,
            PetEntry.COLUMN_PET_GENDER,
            PetEntry.COLUMN_PET_WEIGHT };

    // This loader will execute the ContentProvider's query method on a background thread
    return new CursorLoader(this,   // Parent activity context
            mCurrentPetUri,         // Query the content URI for the current pet
            projection,             // Columns to include in the resulting Cursor
            null,                   // No selection clause
            null,                   // No selection arguments
            null);                  // Default sort order
}
```

EditorActivity—onLoadFinished()

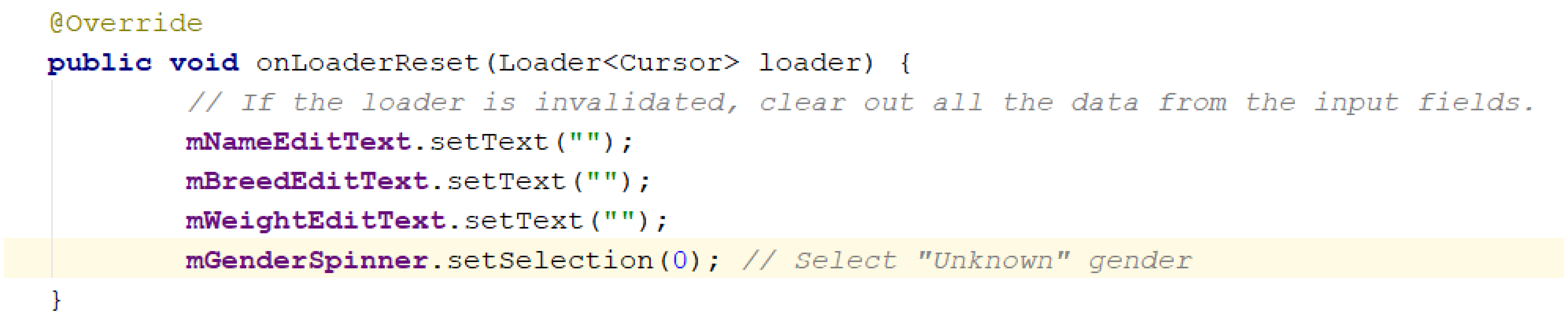


```
@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
    // Bail early if the cursor is null or there is less than 1 row in the cursor
    if (cursor == null || cursor.getCount() < 1) {
        return;
    }
    if (cursor.moveToFirst()) {
        // Find the columns of pet attributes that we're interested in
        int nameColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_NAME);
        int breedColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_BREED);
        int genderColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_GENDER);
        int weightColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_WEIGHT);

        // Extract out the value from the Cursor for the given column index
        String name = cursor.getString(nameColumnIndex);
        String breed = cursor.getString(breedColumnIndex);
        int gender = cursor.getInt(genderColumnIndex);
        int weight = cursor.getInt(weightColumnIndex);
```



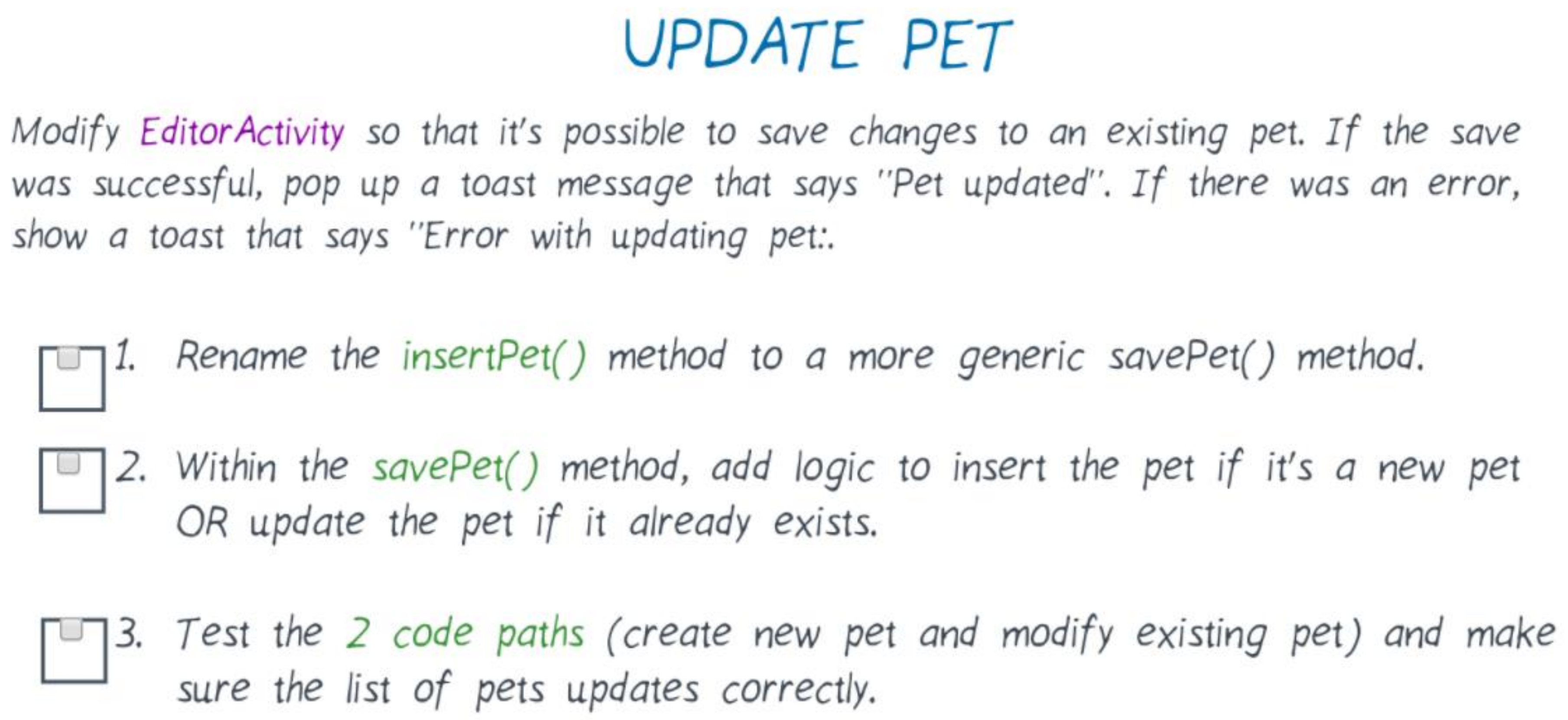
```
        // Update the views on the screen with the values from the database
        mNameEditText.setText(name);
        mBreedEditText.setText(breed);
        mWeightEditText.setText(Integer.toString(weight));

        // Gender is a dropdown spinner, so map the constant value from the database
        // into one of the dropdown options (0 is Unknown, 1 is Male, 2 is Female).
        // Then call setSelection() so that option is displayed on screen as the current
        switch (gender) {
            case PetEntry.GENDER_MALE:
                mGenderSpinner.setSelection(1);
                break;
            case PetEntry.GENDER_FEMALE:
                mGenderSpinner.setSelection(2);
                break;
            default:
                mGenderSpinner.setSelection(0);
                break;
        }
    }
}
```

EditorActivity—onLoaderReset()



```java
@Override
public void onLoaderReset(Loader<Cursor> loader) {
        // If the loader is invalidated, clear out all the data from the input fields.
        mNameEditText.setText("");
        mBreedEditText.setText("");
        mWeightEditText.setText("");
        mGenderSpinner.setSelection(0); // Select "Unknown" gender
}
```

23 - Save changes to existing pet if it already exist



UPDATE PET

Modify EditorActivity so that it's possible to save changes to an existing pet. If the save was successful, pop up a toast message that says "Pet updated". If there was an error, show a toast that says "Error with updating pet:.

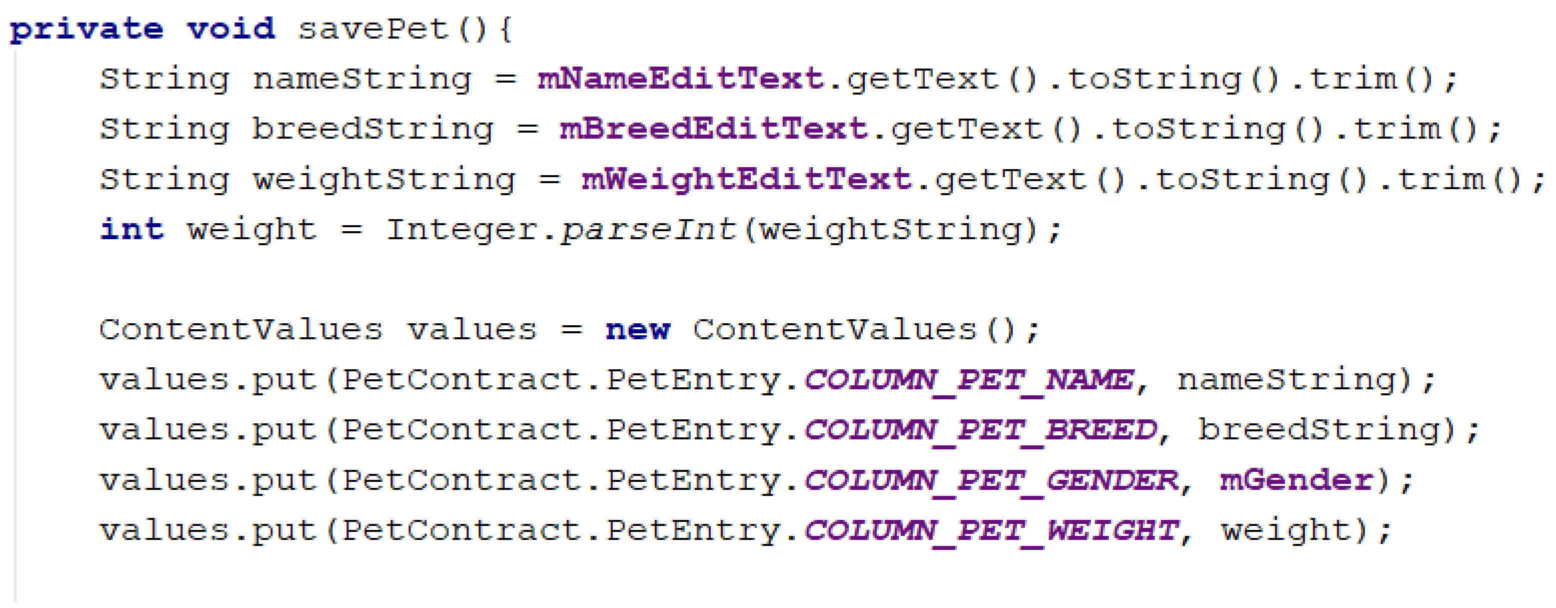
☐ 1. Rename the insertPet() method to a more generic savePet() method.

☐ 2. Within the savePet() method, add logic to insert the pet if it's a new pet OR update the pet if it already exists.

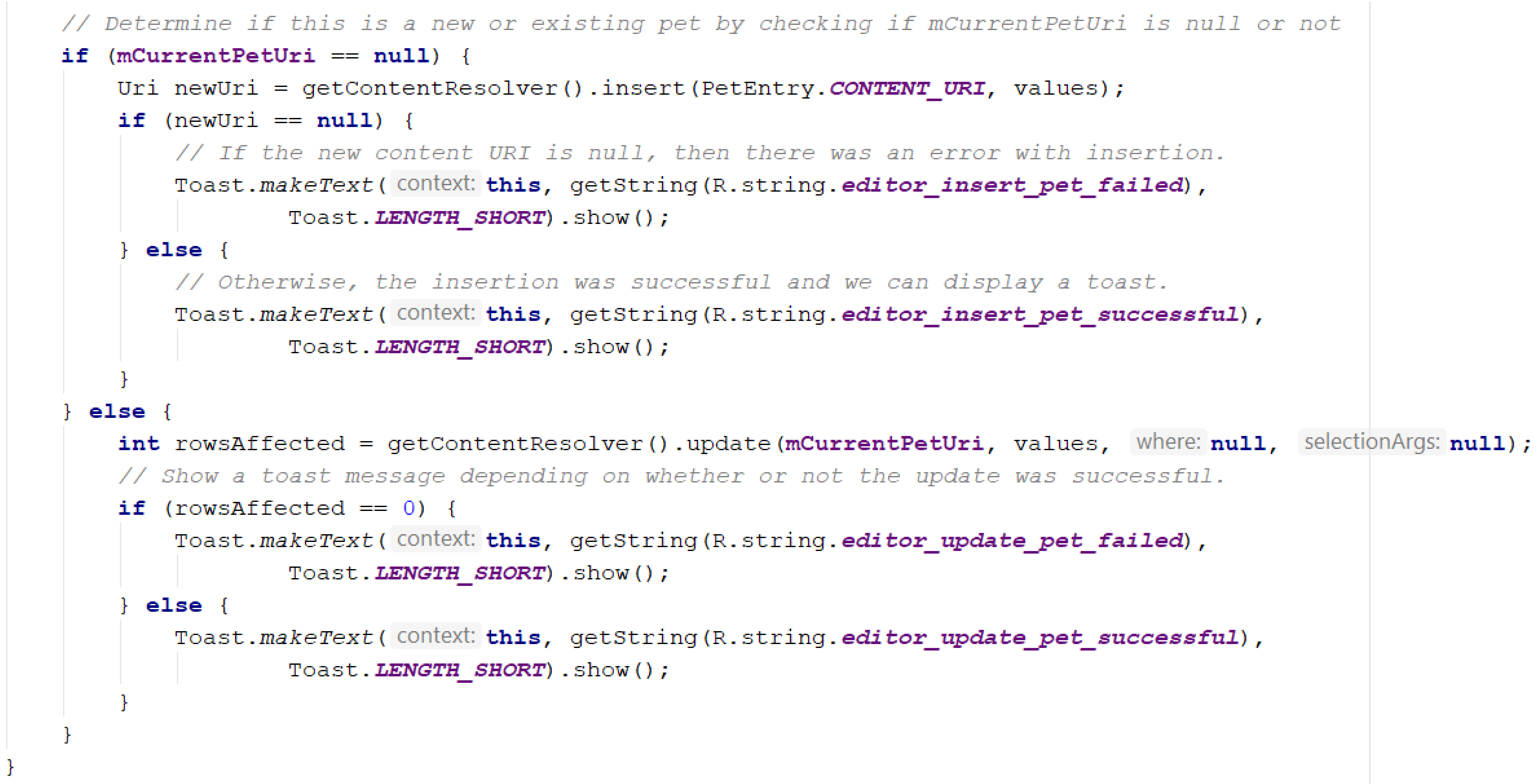
☐ 3. Test the 2 code paths (create new pet and modify existing pet) and make sure the list of pets updates correctly.

EditorActivity – insertPet -- savePet

- Rename insertPet() to savePet()

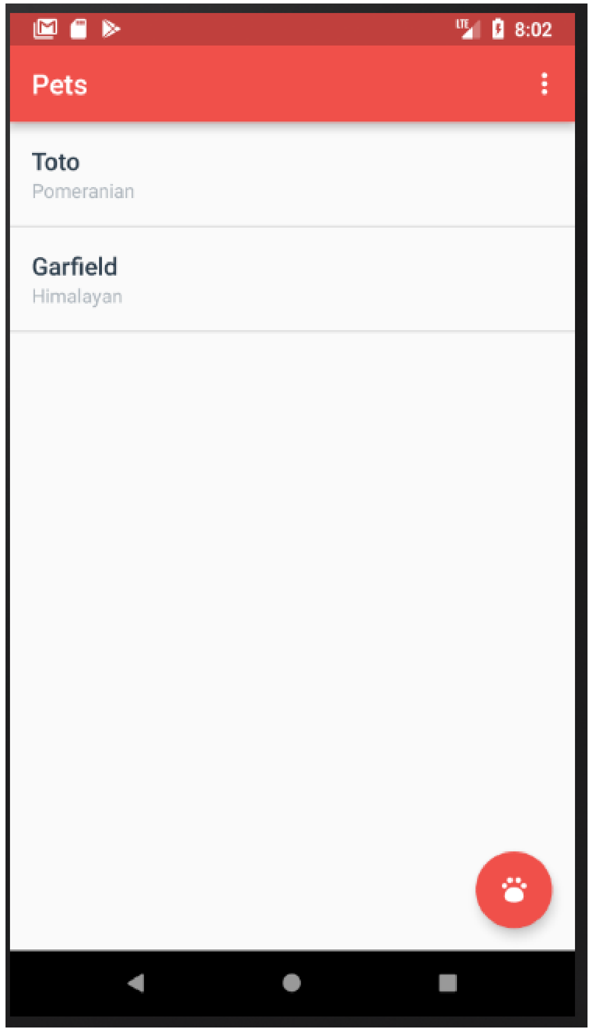


```java
private void savePet(){
    String nameString = mNameEditText.getText().toString().trim();
    String breedString = mBreedEditText.getText().toString().trim();
    String weightString = mWeightEditText.getText().toString().trim();
    int weight = Integer.parseInt(weightString);

    ContentValues values = new ContentValues();
    values.put(PetContract.PetEntry.COLUMN_PET_NAME, nameString);
    values.put(PetContract.PetEntry.COLUMN_PET_BREED, breedString);
    values.put(PetContract.PetEntry.COLUMN_PET_GENDER, mGender);
    values.put(PetContract.PetEntry.COLUMN_PET_WEIGHT, weight);
```



```
    // Determine if this is a new or existing pet by checking if mCurrentPetUri is null or not
    if (mCurrentPetUri == null) {
        Uri newUri = getContentResolver().insert(PetEntry.CONTENT_URI, values);
        if (newUri == null) {
            // If the new content URI is null, then there was an error with insertion.
            Toast.makeText( context: this, getString(R.string.editor_insert_pet_failed),
                    Toast.LENGTH_SHORT).show();
        } else {
            // Otherwise, the insertion was successful and we can display a toast.
            Toast.makeText( context: this, getString(R.string.editor_insert_pet_successful),
                    Toast.LENGTH_SHORT).show();
        }
    } else {
        int rowsAffected = getContentResolver().update(mCurrentPetUri, values,  where: null,  selectionArgs: null);
        // Show a toast message depending on whether or not the update was successful.
        if (rowsAffected == 0) {
            Toast.makeText( context: this, getString(R.string.editor_update_pet_failed),
                    Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText( context: this, getString(R.string.editor_update_pet_successful),
                    Toast.LENGTH_SHORT).show();
        }
    }
}
```

Test App

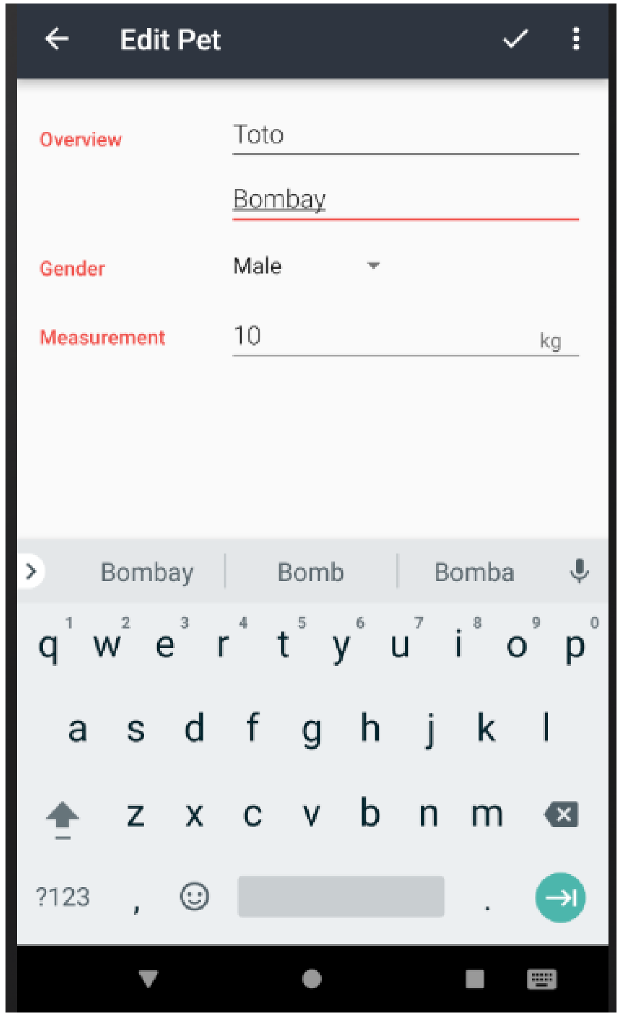


8:02

Pets

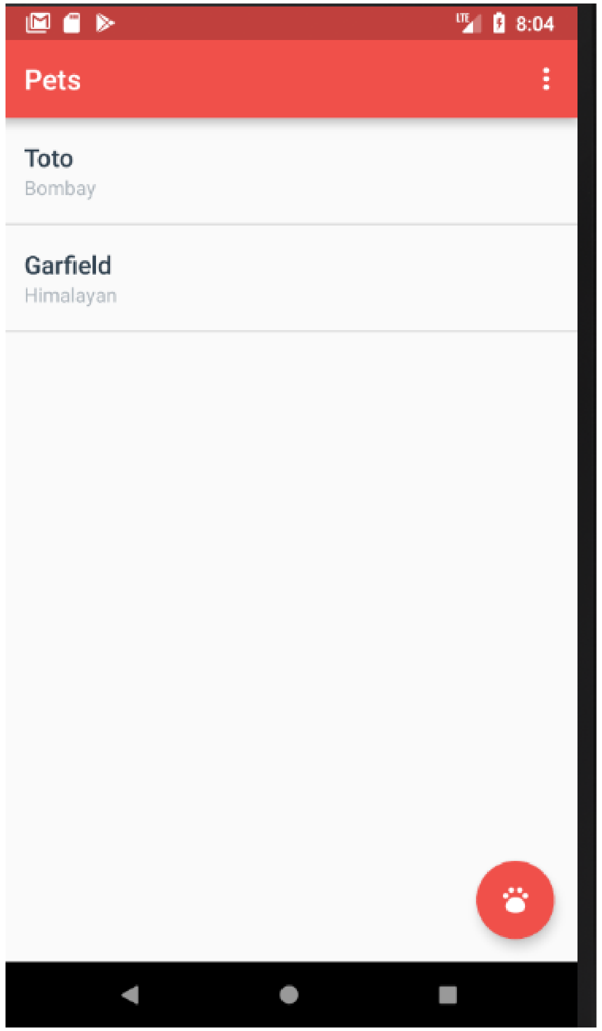
Toto
Pomeranian

Garfield
Himalayan



Edit Pet

Overview	Toto
	Bombay
Gender	Male
Measurement	10 kg



8:04

Pets

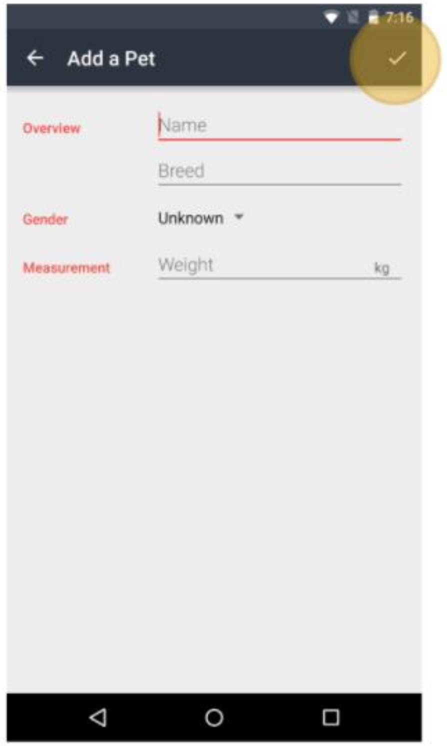
Toto
Bombay

Garfield
Himalayan

24 - Prevent crash with blank editor (12)

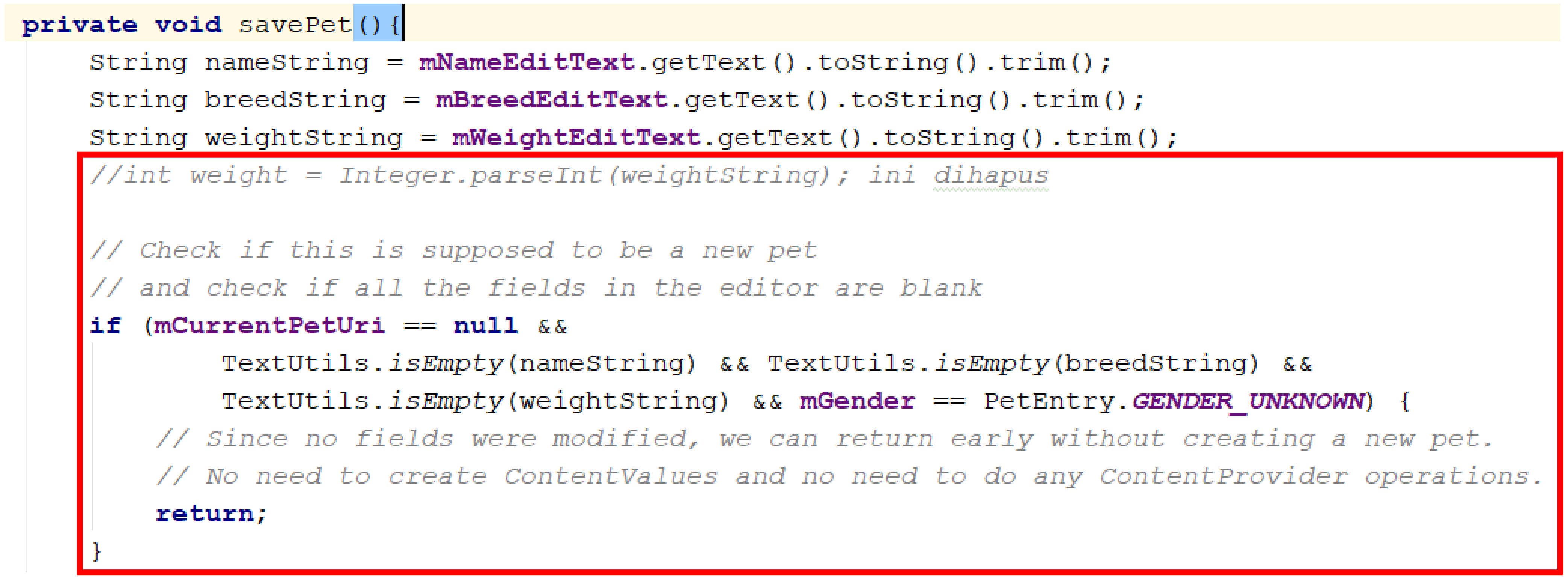
PREVENT CRASH WITH BLANK EDITOR

- [ ] *Modify the editor so that when all the fields are empty, and the "Save" button is clicked, don't create a new pet. Just finish the activity.*

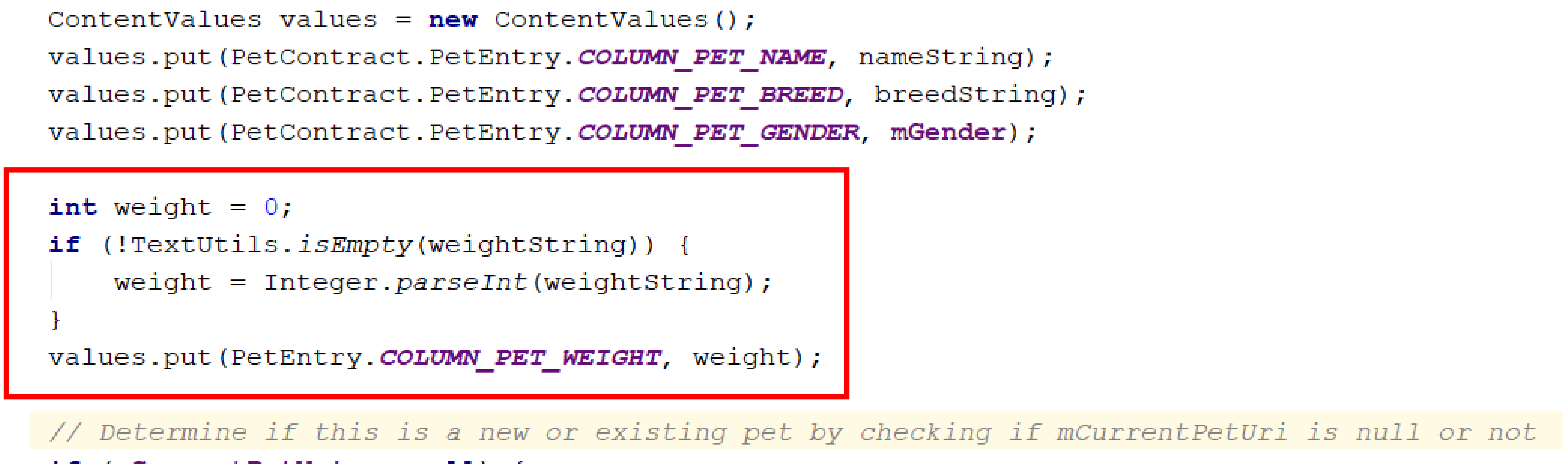


Overview — Name
Breed
Gender — Unknown
Measurement — Weight kg

EditorActivity – savePet()



```java
private void savePet(){
    String nameString = mNameEditText.getText().toString().trim();
    String breedString = mBreedEditText.getText().toString().trim();
    String weightString = mWeightEditText.getText().toString().trim();
    //int weight = Integer.parseInt(weightString); ini dihapus

    // Check if this is supposed to be a new pet
    // and check if all the fields in the editor are blank
    if (mCurrentPetUri == null &&
            TextUtils.isEmpty(nameString) && TextUtils.isEmpty(breedString) &&
            TextUtils.isEmpty(weightString) && mGender == PetEntry.GENDER_UNKNOWN) {
        // Since no fields were modified, we can return early without creating a new pet.
        // No need to create ContentValues and no need to do any ContentProvider operations.
        return;
    }
```



```
ContentValues values = new ContentValues();
values.put(PetContract.PetEntry.COLUMN_PET_NAME, nameString);
values.put(PetContract.PetEntry.COLUMN_PET_BREED, breedString);
values.put(PetContract.PetEntry.COLUMN_PET_GENDER, mGender);

int weight = 0;
if (!TextUtils.isEmpty(weightString)) {
    weight = Integer.parseInt(weightString);
}
values.put(PetEntry.COLUMN_PET_WEIGHT, weight);
// Determine if this is a new or existing pet by checking if mCurrentPetUri is null or not
```

25 - Warn user about losing unsaved changes (13

EditorActivity.java

```
private int mGender = PetEntry.GENDER_UNKNOWN;

/** Boolean flag that keeps track of whether the pet has been edited (true) or not (false) */
private boolean mPetHasChanged = false;

/**
 * OnTouchListener that listens for any user touches on a View, implying that they are modifying
 * the view, and we change the mPetHasChanged boolean to true.
 */
private View.OnTouchListener mTouchListener = new View.OnTouchListener() {
    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        mPetHasChanged = true;
        return false;
    }
};
```

EditorActivity – onCreate()

```
    mNameEditText.setOnTouchListener(mTouchListener);
    mBreedEditText.setOnTouchListener(mTouchListener);
    mWeightEditText.setOnTouchListener(mTouchListener);
    mGenderSpinner.setOnTouchListener(mTouchListener);

    setupSpinner();
}
```

Make a method for creating a “Discard changes” dialog (put after onLoaderReset()

```
private void showUnsavedChangesDialog(
        DialogInterface.OnClickListener discardButtonClickListener) {
    // Create an AlertDialog.Builder and set the message, and click listeners
    // for the postivie and negative buttons on the dialog.
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage(R.string.unsaved_changes_dialog_msg);
    builder.setPositiveButton(R.string.discard, discardButtonClickListener);
    builder.setNegativeButton(R.string.keep_editing, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            // User clicked the "Keep editing" button, so dismiss the dialog
            // and continue editing the pet.
            if (dialog != null) {
                dialog.dismiss();
            }
        }
    });

    // Create and show the AlertDialog
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

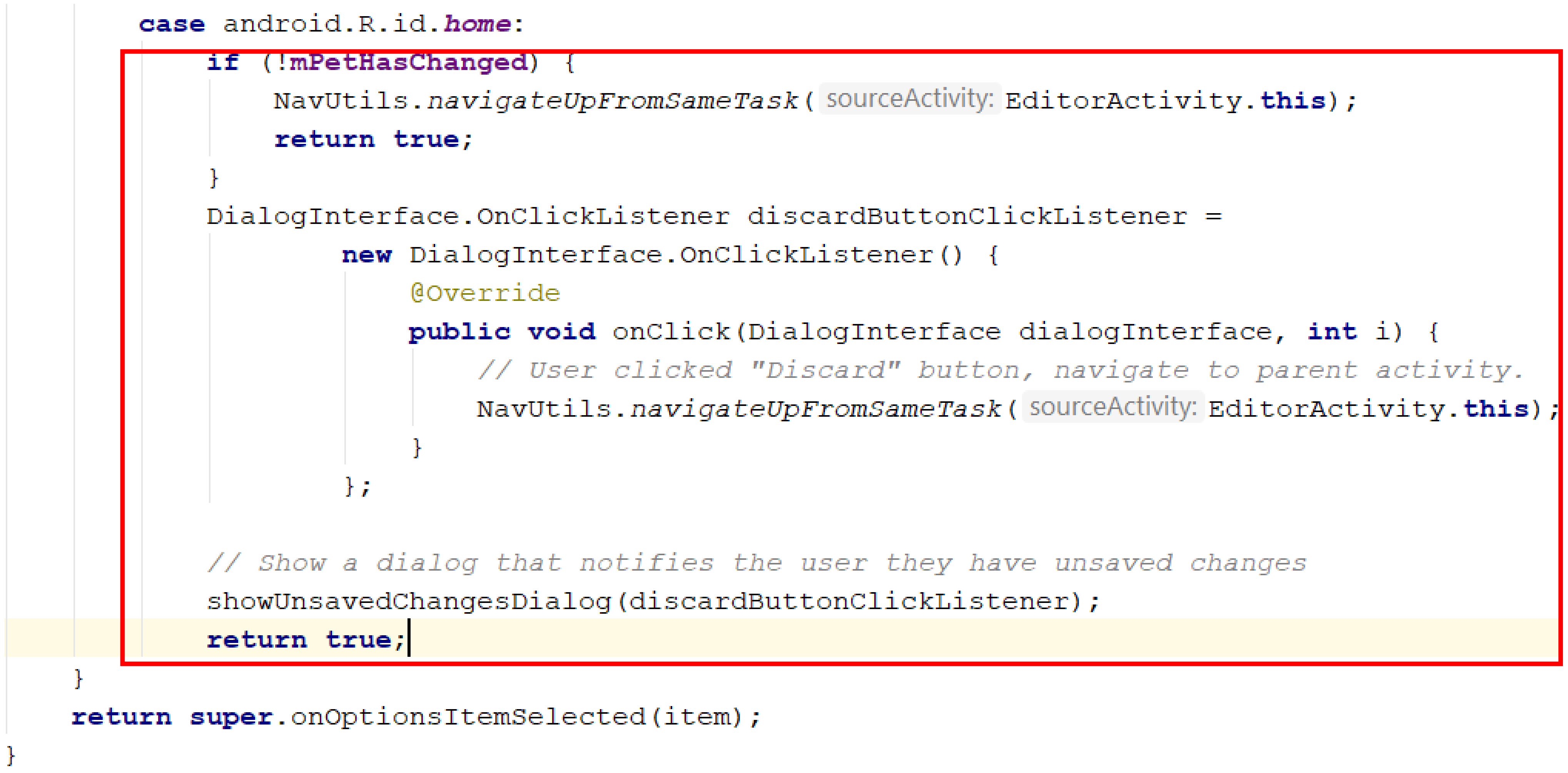
Add to string.xml

```xml
<!-- Dialog message when user is leaving editor but hasn't saved changes [CHAR LIMIT=NONE] -->
<string name="unsaved_changes_dialog_msg">Discard your changes and quit editing?</string>

<!-- Dialog button text for the option to discard a user's changes [CHAR LIMIT=20] -->
<string name="discard">Discard</string>

<!-- Dialog button text for the option to keep editing the current pet [CHAR LIMIT=20] -->
<string name="keep_editing">Keep Editing</string>
```

Create onBackPressed() method

```java
@Override
public void onBackPressed() {
    // If the pet hasn't changed, continue with handling back button press
    if (!mPetHasChanged) {
        super.onBackPressed();
        return;
    }

    // Otherwise if there are unsaved changes, setup a dialog to warn the user.
    // Create a click listener to handle the user confirming that changes should be discarded.
    DialogInterface.OnClickListener discardButtonClickListener =
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    // User clicked "Discard" button, close the current activity.
                    finish();
                }
            };

    // Show dialog that there are unsaved changes
    showUnsavedChangesDialog(discardButtonClickListener);
}
```

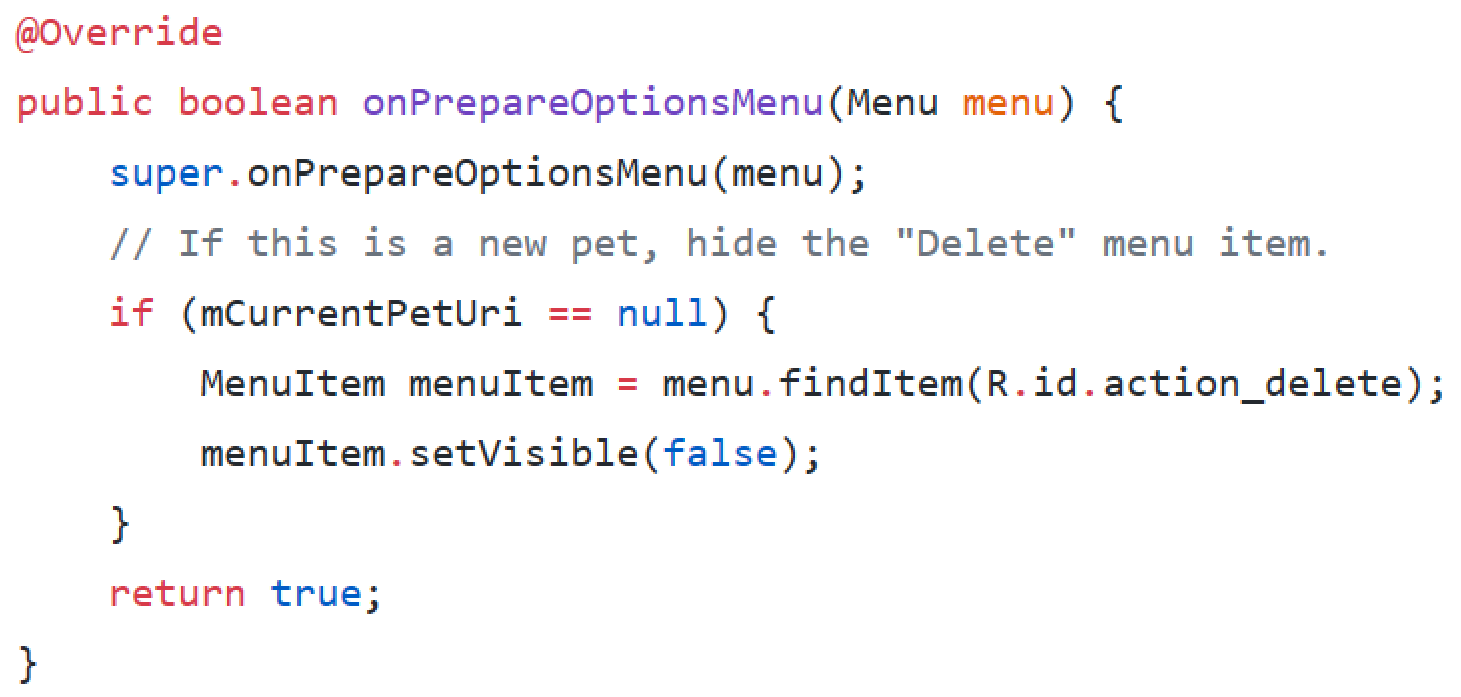
onOptionItemSelected()



```java
            case android.R.id.home:
                if (!mPetHasChanged) {
                    NavUtils.navigateUpFromSameTask( sourceActivity: EditorActivity.this);
                    return true;
                }
                DialogInterface.OnClickListener discardButtonClickListener =
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                // User clicked "Discard" button, navigate to parent activity.
                                NavUtils.navigateUpFromSameTask( sourceActivity: EditorActivity.this);
                            }
                        };

                // Show a dialog that notifies the user they have unsaved changes
                showUnsavedChangesDialog(discardButtonClickListener);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

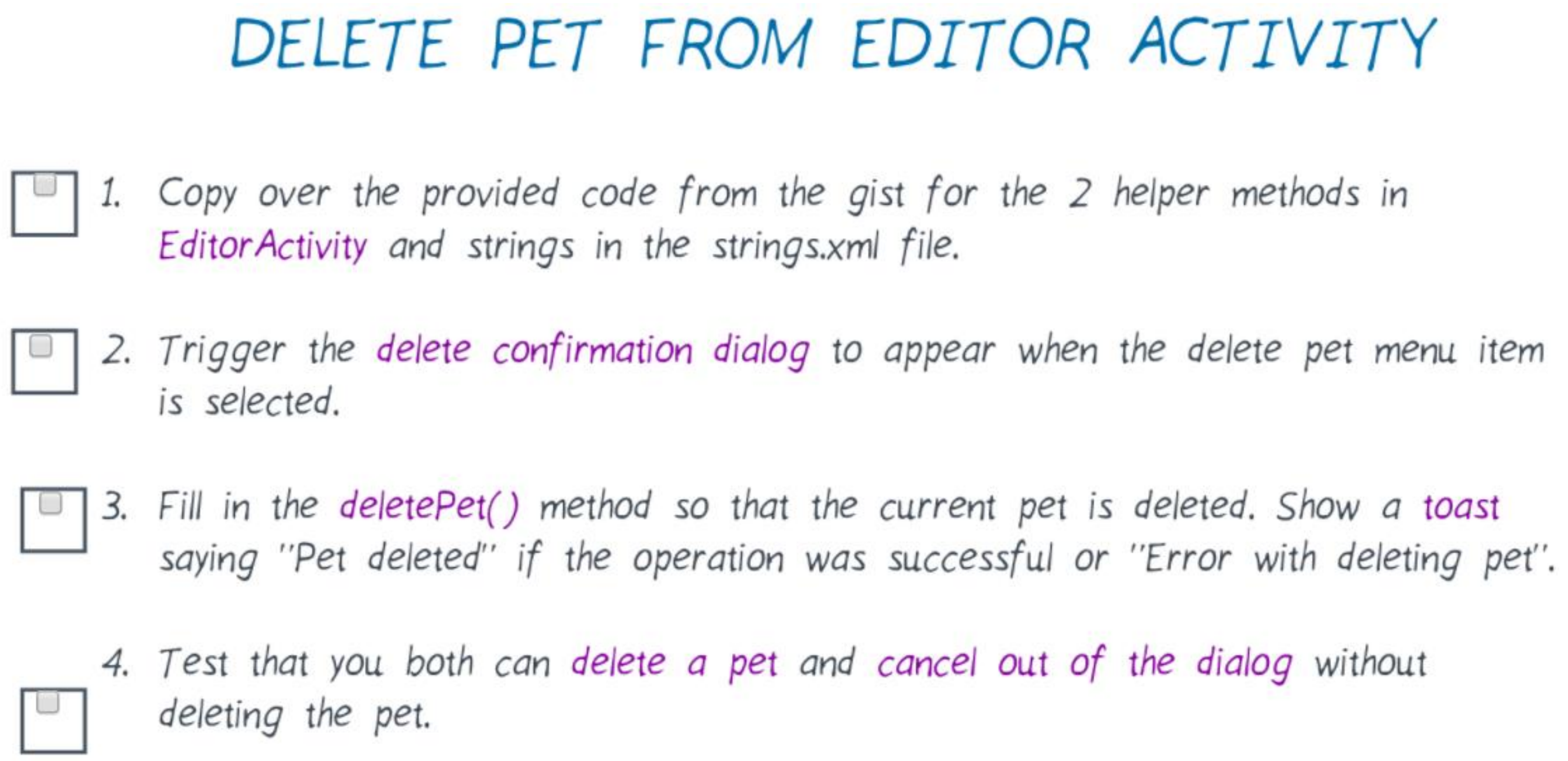
26 - Hide delete menu option for new pets (14)

EditorActivity-- onPrepareOptionMenu() put after onCreateOptionMenu()



```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    super.onPrepareOptionsMenu(menu);
    // If this is a new pet, hide the "Delete" menu item.
    if (mCurrentPetUri == null) {
        MenuItem menuItem = menu.findItem(R.id.action_delete);
        menuItem.setVisible(false);
    }
    return true;
}
```

27 - Delete pet from editor menu item



DELETE PET FROM EDITOR ACTIVITY

☐ 1. Copy over the provided code from the gist for the 2 helper methods in EditorActivity and strings in the strings.xml file.

☐ 2. Trigger the delete confirmation dialog to appear when the delete pet menu item is selected.

☐ 3. Fill in the deletePet() method so that the current pet is deleted. Show a toast saying "Pet deleted" if the operation was successful or "Error with deleting pet".

☐ 4. Test that you both can delete a pet and cancel out of the dialog without deleting the pet.

Copy the showDeleteConfirmationDialog() and string.xml

- https://gist.github.com/udacityandroid/ca98067c255d6ef324eb4e430645e428
- Put showDeleteConfirmationDialog() after showUnsavedChangeDialog()
- Paste the code for string.xml

EditorActivity – onOptionItemSelected()

- Call showDeleteConfirmationDialog

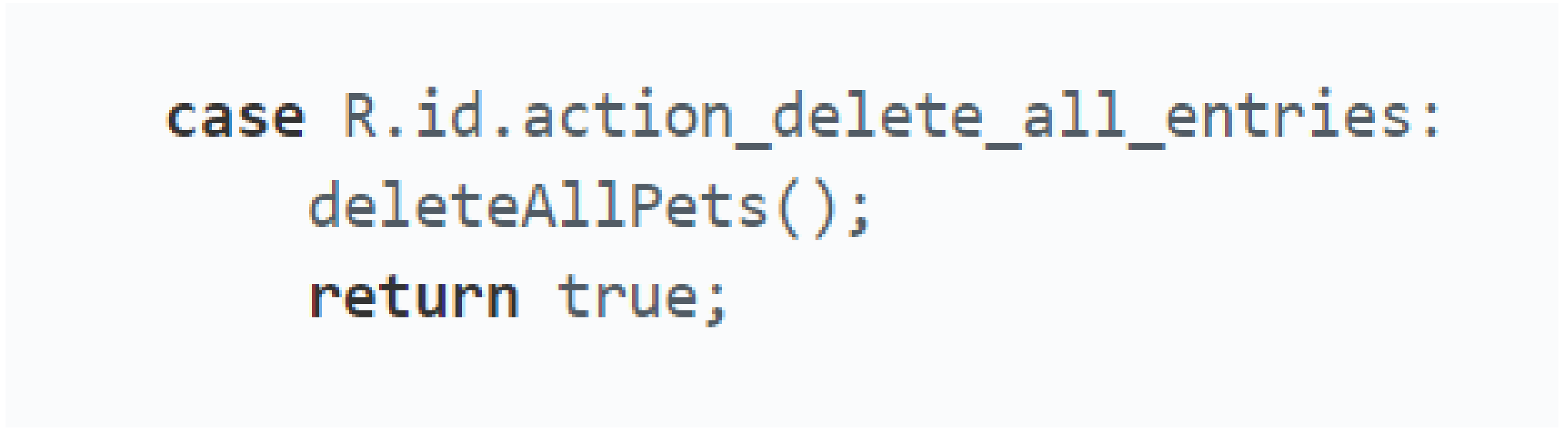
```
// Respond to a click on the "Delete" menu option
case R.id.action_delete:
    // Pop up confirmation dialog for deletion
    showDeleteConfirmationDialog();
    return true;
```

Fill in deletePet()

```java
private void deletePet() {
    // Only perform the delete if this is an existing pet.
    if (mCurrentPetUri != null) {
        // Call the ContentResolver to delete the pet at the given content URI.
        // Pass in null for the selection and selection args because the mCurrentPetUri
        // content URI already identifies the pet that we want.
        int rowsDeleted = getContentResolver().delete(mCurrentPetUri, null, null);

        // Show a toast message depending on whether or not the delete was successful.
        if (rowsDeleted == 0) {
            // If no rows were deleted, then there was an error with the delete.
            Toast.makeText(this, getString(R.string.editor_delete_pet_failed),
                    Toast.LENGTH_SHORT).show();
            } else {
                // Otherwise, the delete was successful and we can display a toast.
                Toast.makeText(this, getString(R.string.editor_delete_pet_successful),
                        Toast.LENGTH_SHORT).show();
            }
        }

        // Close the activity
        finish();
    }
}
```

28 - Delete all pets from catalog menu item CatalogActivity – onOptionsItemSelected()

Call deleteAllPets() @ onOptionItemSelected()



```
case R.id.action_delete_all_entries:
    deleteAllPets();
    return true;
```

In CatalogActivity - Create deleteAllPets()

```
private void deleteAllPets() {
    int rowsDeleted = getContentResolver().delete(PetEntry.CONTENT_URI, null, null);
    Log.v("CatalogActivity", rowsDeleted + " rows deleted from pet database");
}
```

29 - Display “Unknown breed” for pets without breed

- Add in string.xml

```
<!-- Label for the pet's breed if the breed is unknown [CHAR LIMIT=20] -->
<string name="unknown_breed">Unknown breed</string>
```

PetCursorAdapter.java – bindView()

```java
        // Read the pet attributes from the Cursor for the current pet
        String petName = cursor.getString(nameColumnIndex);
        String petBreed = cursor.getString(breedColumnIndex);

        // If the pet breed is empty string or null, then use some default text
        // that says "Unknown breed", so the TextView isn't blank.
        if (TextUtils.isEmpty(petBreed)) {
            petBreed = context.getString(R.string.unknown_breed);
        }

        // Update the TextViews with the attributes for the current pet
        nameTextView.setText(petName);
        summaryTextView.setText(petBreed);
    }
}
```