Assignment 8 – Movie Listings

Goals

- Create a UI that dynamically displays movie showings in a list
- Use MVC techniques to separate layers of program functionality
- Use an explicit intent to launch a new activity that displays movie details
- Allow the user to add new movies and add comments to existing ones

Required naming convention (replace # with the current assignment number)

- **Application Name**
    - A#
- **Company Domain**
    - firstname.lastname.itp341
- **Package Name** (should be automatically generated)
    - itp341.lastname.firstname.a#.app
- **Zip File** (include entire project folder)
    - A#.Lastname.FirstName.zip

Hint: Passing objects in intents

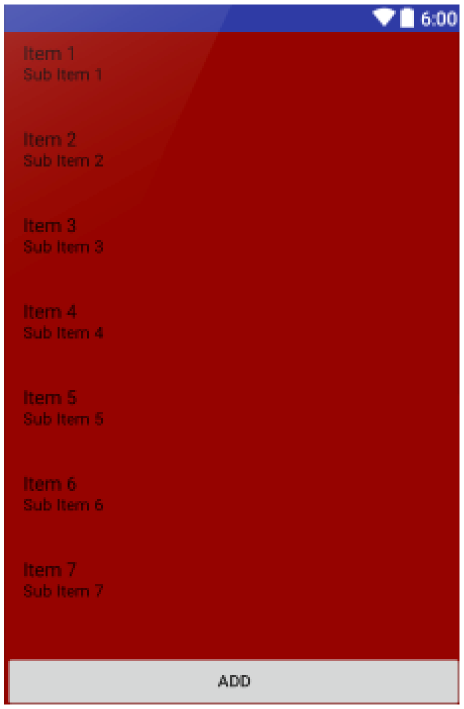
- Any custom object can be put into an **extra** and passed via an **intent**
- All that is required is that the class be **serializable**, which is a way of representing an object as a series of bytes, which can then be saved
- To make a class **serializable**, the class must implement the serializable interface
  `public class MyAwesome class implements serializable { … }`
- To put an object into the intent, use
  `putExtra(…)`
- To retrieve an object from an intent
  `getSerializableExtra(…)`

Requirements

- Create new Android Application Project
- Follow default prompt, but make sure to choose **Empty Activity**
- Model (these are **not** expected to persist across sessions)
    - Create a model class for a Movie (**Movie**) with the following:
        - Methods
            - Constructor
            - toString
            - getters and setters

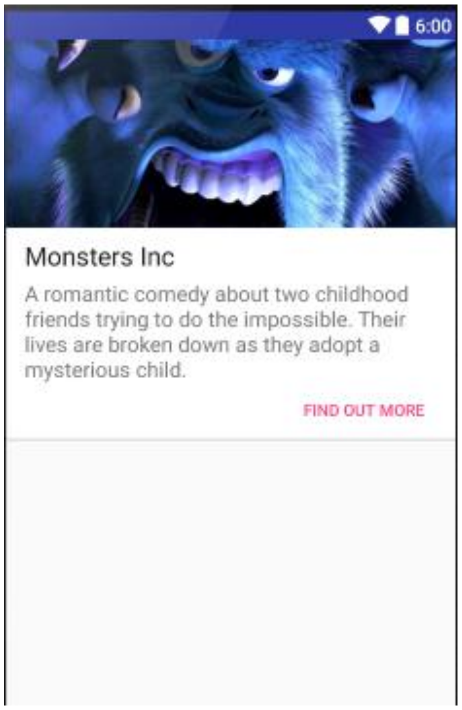
- Instance variables for all the following (naming up to you)
  - title
  - description
  - genre (this should be an int)
  - List of comments (must be resizable, just Strings)
  - URL (optional)
- Create a singleton which manages all **Movie** objects that can:
  - Get an instance of the singleton
  - Get the number of **Movie** objects
  - Get a **Movie** object at a specific index
  - Add a **Movie** object
  - Add a comment to a specific **Movie** object

- For all views:
  - Follow the rules in the Android Design Guide. This includes height of widgets, spacing, padding, etc. http://developer.android.com/design/style/metrics-grids.html
- For all Activities:
  - Be sure to create class members that will refer to the widget you want to access later
  - This means you should only call **findViewById** in the **onCreate** method
- **MainActivity** – view



  - Create a similar layout to the one above (color not necessary)

- Additionally, create a new layout (we suggest you name it **layout_list_movie**.xml) similar to the one below:



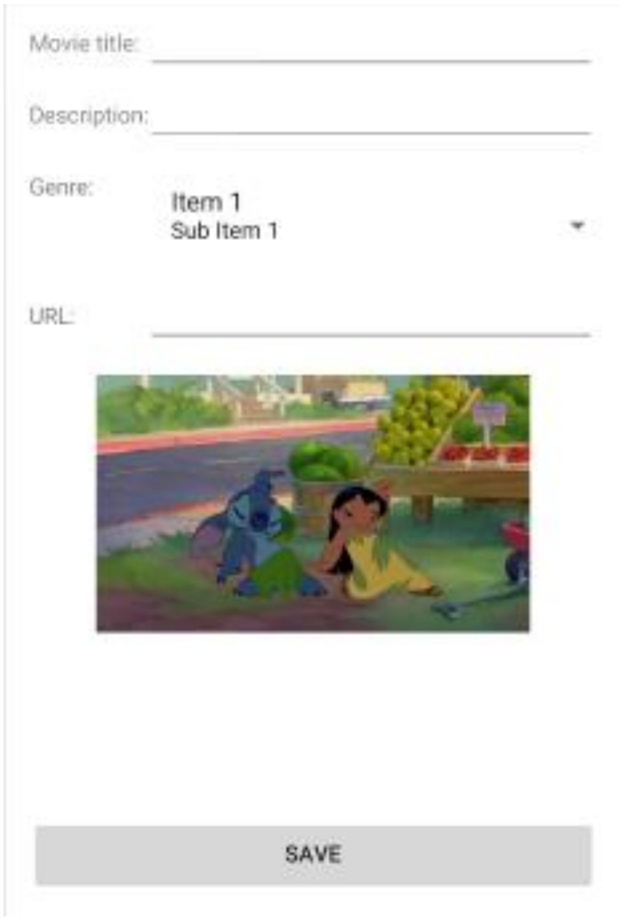
A romantic comedy about two childhood friends trying to do the impossible. Their lives are broken down as they adopt a mysterious child.

- Note that the bottom right corner of the layout holds a Button

- **MainActivity** – code
    - Create and store an instance of your **MovieListAdapter**
    - Assign your list to use your adapter.
    - Create a reference to your **Movie** adding button
    - When the **Movie** adding button is pressed, explicitly launch an intent to the **CreateActivity** and expect a result
    - Create a public **refresh** method which calls your adapter's **notifyDataSetChanged** method
    - When you come back from a child Activity, call your **refresh** method to update the list of **Movies**
    - Adapter code
        - Create an adapter that subclasses **ArrayAdapter** (we suggest you name it **MovieListAdapter**)
        - Take in and store **Content** as a member variable through the constructor
        - In the **getView** method, do the following:
            - Inflate a new View using the extra layout (**layout_list_movie**)
            - Create references to any necessary widgets

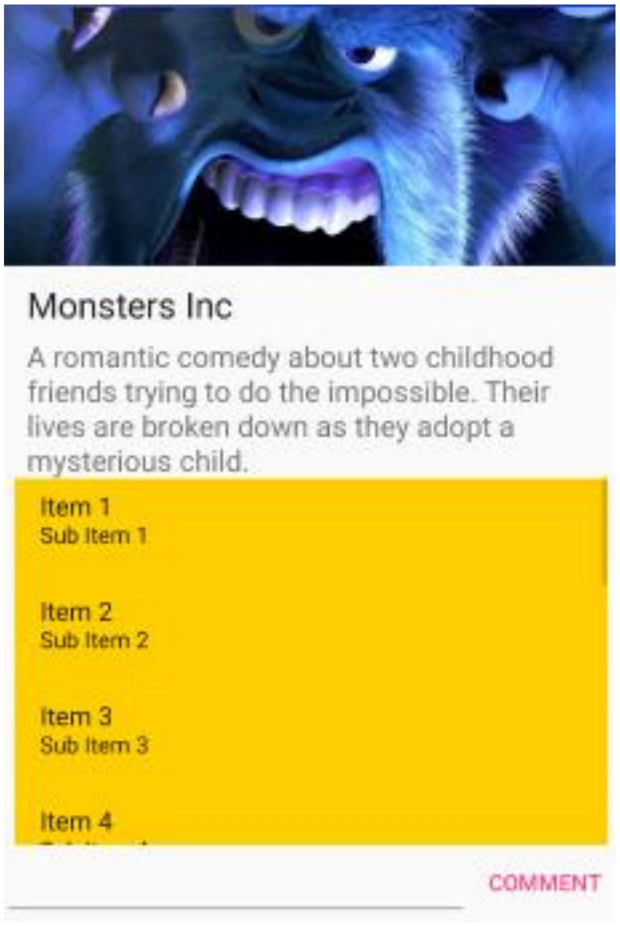
- Pull the appropriate **Movie** from the **Movie** Singleton
- Fill in the inflated view's information using the **Movie** (Hint: this is easier after you complete CreateActivity)
- Set the button's tag to the current position (Hint: use the **setTag** method)
- When the button is pressed, explicitly launch an intent to the **DetailActivity** while passing in the tag of the view given to you in the **onClick** method

- **CreateActivity** – view
  - Create a similar layout to the one below:



  - There should be at least 5 movie genres, we suggest:
    - Horror
    - Action
    - Drama
    - Comedy
    - Sci-Fi

  - Each movie genre should have an image added to the app, we have included our suggestions (hint: to keep things small, only add them for high pixel density)
- **CreateActivity** – code
  - Create references to each widget needed to create a **Movie** item
  - When selecting a genre, you should display that genre's image as a preview to the user
  - The URL field is optional (see extra credit)
  - When the save button is pressed, store all the information the user entered into a **Movie** object and then add it to the **Movie** singleton. Then return to the parent activity
- **DetailActivity** – view
  - Create a similar layout to the one below:



Monsters Inc

A romantic comedy about two childhood friends trying to do the impossible. Their lives are broken down as they adopt a mysterious child.

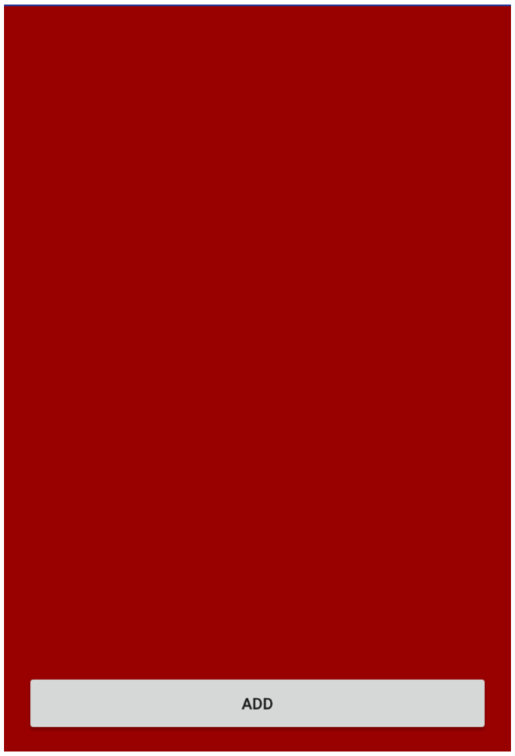
- **DetailActivity** – code
  - Fill in your widgets with **Movie** data as necessary.
  - Remember to add a list with a simple adapter for the list of comments for the **Movie**

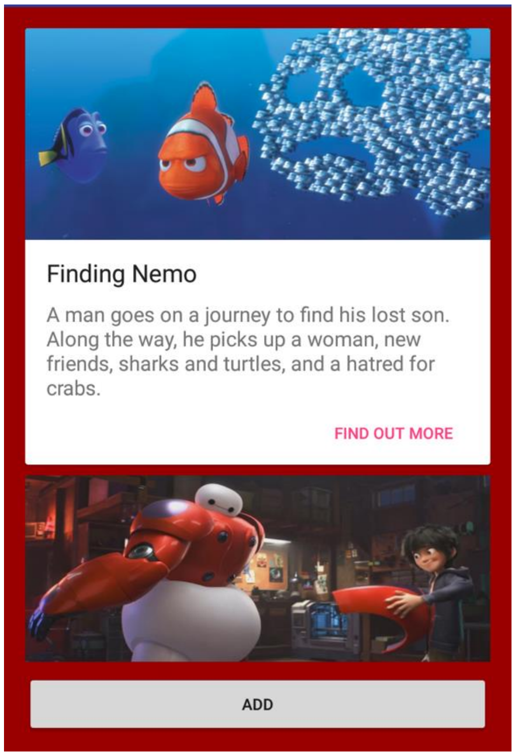
  - Allow users to write a new comment for the **Movie** using the **EditText** at the bottom of the view. You should set a listener for the **Button** to save this comment to the **Movie** object's list of comments using the **Movie** singleton
  - You probably want to clear out the **EditText** whenever a comment is saved
  - Be sure to call the adapter's **notifyDataSetChanged** method when a new comment is added to the **Movie**
  - Any comments added to **Movies** should be visible when you leave the **DetailActivity** and then open it again for the same **Movie**

Extra Credit

- Use Picasso and the optional URL field to display customized images for **Movies** in your list and in **DetailActivity**
- Use **CardView** as the layout for your list items and follow Material Design standards
- Change user comments to include time stamps and names. At this point, is there a better way to store comments than a list for each **Movie** object?

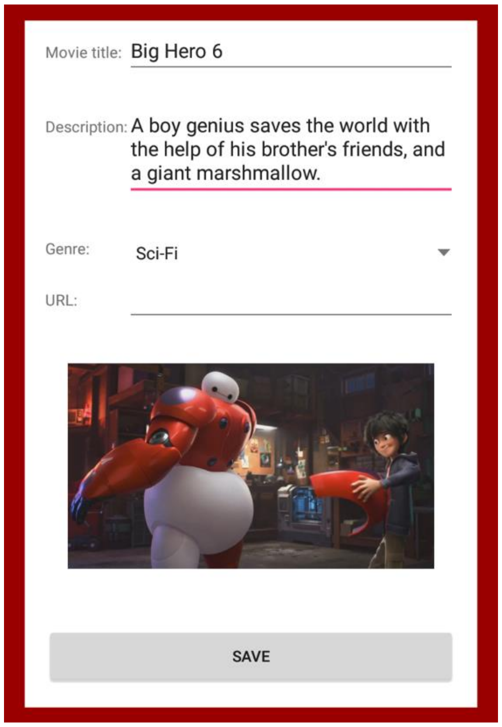
Sample Output





Finding Nemo

A man goes on a journey to find his lost son. Along the way, he picks up a woman, new friends, sharks and turtles, and a hatred for crabs.

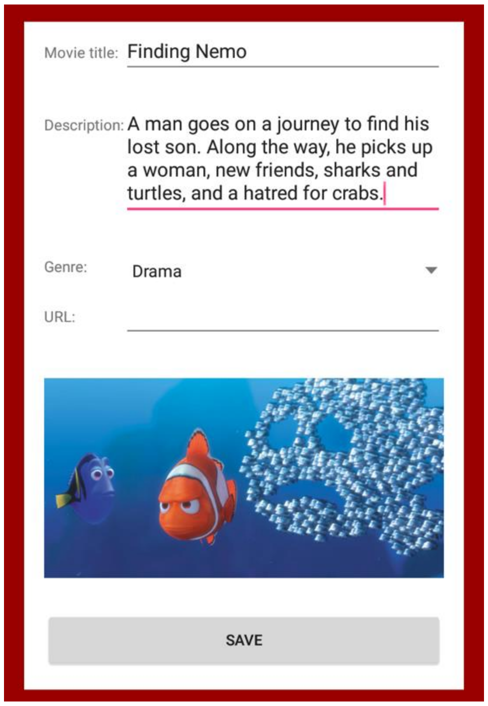


Movie title: Big Hero 6

Description: A boy genius saves the world with the help of his brother's friends, and a giant marshmallow.

Genre: Sci-Fi

URL:

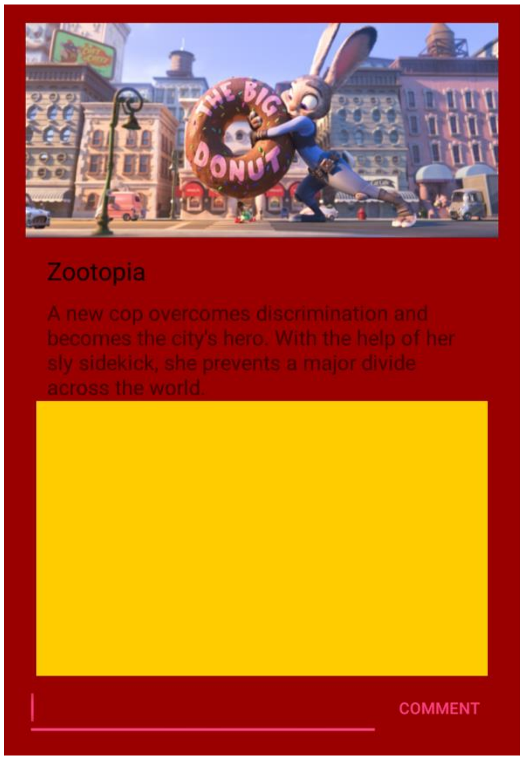


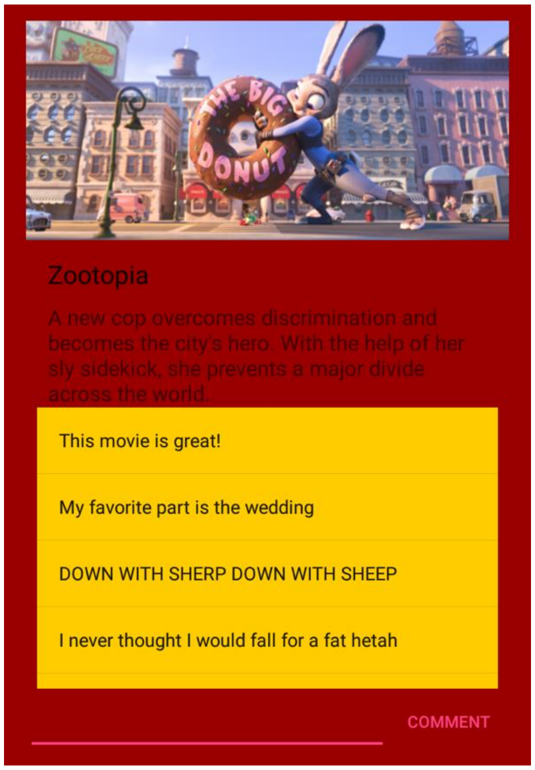
Movie title: Finding Nemo

Description: A man goes on a journey to find his lost son. Along the way, he picks up a woman, new friends, sharks and turtles, and a hatred for crabs.

Genre: Drama

URL:


Zootopia
A new cop overcomes discrimination and becomes the city's hero. With the help of her sly sidekick, she prevents a major divide across the world.
COMMENT


Zootopia
A new cop overcomes discrimination and becomes the city's hero. With the help of her sly sidekick, she prevents a major divide across the world.
This movie is great!
My favorite part is the wedding
DOWN WITH SHERP DOWN WITH SHEEP
I never thought I would fall for a fat hetah
COMMENT

Deliverables

1. A compressed file containing your app. Follow the guidelines for full credit.
   Here are the instructions for submission
   a) Navigate to your project folder.
   b) Include the entire folder in a zip file
   c) Upload zip file to Blackboard site for our course

Grading

Item	Points
Movie model and Movie singleton are implemented and used	10
MainActivity can go to CreateActivity and updates on return	5
MainActivity displays a list using custom list view/adapter	15
MainActivity can go to DetailActivity and the selected index	5
CreateActivity creates a Movie object and updates the singleton	5
DetailActivity displays Movie object's data using singleton	5
DetailActivity displays Movie object's comments in a list	5
Total	**50**