Lecture
Model View Controller s/w architecture pattern

- DT228/3

- Dr. Susan McKeever

Structuring your code and classes..

When you create your various classes in an application, you want to:

- Promote reusability (e.g. panels for OK/Cancel buttons might be used in several screens)

- Minimise code maintenance when something changes (e.g. dB structure, new GUI device)

- Maintain as much flexibility as possible

Model View controller

- MVC is a common s/w architecture pattern used for interactive applications (i.e. applications that have a Graphical User i/f)

- Model = classes that deal with storing and using data
- View = classes or code that deal with the GUI part
- Controller = (non graphical) classes that bridge the View and the Model

MVC

- Philosophy

- Keep the parts that may be re-used elsewhere or changed SEPARATE

Model

- Model = classes that deal with storing and using **data**
  - If data structure(s) or rules change.. Just change these classes
  - Can have multiple different GUIs using the same model
    - E.g. calculator functions add/substract etc
    - Different GUIs (standard web, accessible web etc)

View

- View = classes that deal with the GUI part
  - If the GUI layout changes, don't (necessarily) need to change the model or control classes
    - E.g. change borders of buttons

Controller

- Controller = (non graphical) classes that bridge the View and the Model
  - Kept separate
    - Listener classes that trigger behaviour
    - Generally.. Consists of everything that is not the layout or not directly storing/using data

MVC – Philosophy

- Separation of components for:

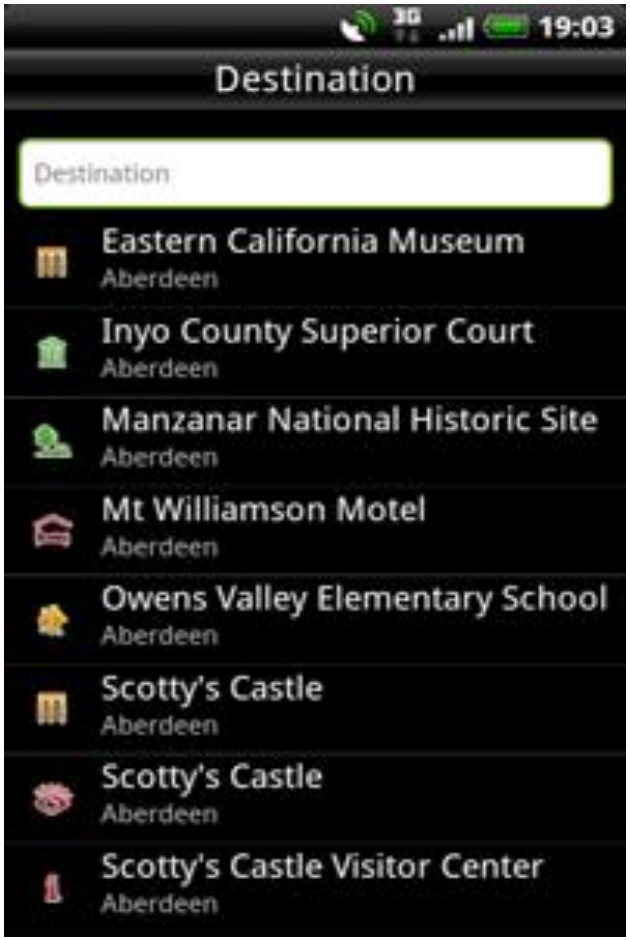
- **Reuse** – e.g. supposing two different screens in Android have same appearance?
- Maintainability e.g. supposing I want to change colours of all my screen. Does the java code change?

Does Android support MVC s/w architecture pattern?

- Somewhat...
- Abstracted screen layouts..View
  - Easy to maintain if changed without necessarily changing M/C parts
- Static resources are kept separate from dynamic code.. .. bit of model
- Dynamic model code.. developer determines
  E.g. put database code into separate classes from the activity
- Controller code?

Example

- App – this screen displays a list of tourist destinations



MVC – List example

Classes /layouts needed?

For View bit..

- Listview Layout to display list etc
- Row layout

For Model bit..

- What "data" is involved? What can be done to it? Is there a dB involved?

- Put dB connection/operatios into separate class(es) to the activity – then other activities can use this class for dB connection and operations and code is centralised

For Controller bit..

What's needed apart from GUI and doing the model part? Activity..

Other things..

- Giant all purpose class for everything – wrong!
  - Hard to maintain
  - Nothing reusable
- Decide which "parts" might be reusable
  - E.g. database code
  - Parts of screen (OK/cancel panels) – Android supplies `<Fragment>`
- Keep MVC as a guiding principle but…

Finally

- As with any design
  - There is no single "right" answer to how you break your app into classes
  - But design in advance, rather than just letting your code wander along in a single class or two…
- MVC is one s/w architecture
  - Some techniques we've seen contradict it
    - E.g. implementing listeners directly in the activity code
    - Anonymous listeners – embedded against the GUI component – has its own advantages..
    - These approaches have their own merits..
  - Be aware of the trade offs.