Mandatory assignment 01

Dennis Thinh Tan Nguyen

February 28, 2016

Contents

1 Important Backend design choices

During the design and development of Tingle V.4, many design choices have been made to improve the backend of application compared to Tingle V.3. Some of the most important choices are described in the following subsections

1.1 MVC architecture

Tingle is utilising the MVC architectural design pattern to separate the system into three interconnected parts. This was previously introduced in Tingle V2 with a simple implementation of the MVC design pattern (see figure 1 for class diagram).

As for Tingle V4 with the introduction of fragments and other implementation choices such as dependency injection, the MVC pattern enabled one to extend the model, controller or view in a clear way and keep complexity within each of their respected set of responsibility (see figure 2 for class diagram).

1.2 Repository pattern with dependency injection

To enable future extensions of the repository, a generic repository pattern has been adopted. The previous hardcoded thingsDB (TingleV2-3) in TingleActivity made it impracticable to extend or even replace a given storage solution. Consequently, in Tingle V4, the following components have been introduced:

- ThingsDatabase
- IRepository
- InMemoryRepository
- Entity

ThingsDatabase is a class that act as a repository in which activities can perform CRUD operations on. It is provided as a singleton to TingleActivity. Other fragments can then request a reference to the instance.
ThingsDatabase can be injected with any repository that implements the IRepository interface. The repository dependency is being handled by Dagger2 Dependency Framework[1], which is a fully static, compile-time third-party dependency injection framework.

Currently, TingleApp V4 does not have a persistent way of storing database entities, but is, however, using the InMemoryRepository that stores items in the memory. This class is then injected to ThingsDatabase.

[1]Dagger2 website: http://google.github.io/dagger/

1.2.1 IDs for Things/entities

Another abstract class has also been introduced called Entity which serves as a superclass for the Thing Class. The Entity class contains an ID which is used to uniquely identify a Thing, reduce future code duplication but also separate application logic from storage. A repository can take classes that are derived from Entity and store them.

1.3 Fragment callbacks and eventhandlers

Another important design choice is to use event handlers whenever a fragment is required to call back to the host activity. By way of example, when a user presses the add button, one may also want to update the view of a list containing items. One solution is to retrieve the host activity within a fragment and then call the appropriate method.

This is not an effective solution as this will increase coupling but also expose unnecessary implementations. So, we use event handlers that a fragment can call. A host activity may then implement the event handlers and specify what will happen whenever an event is executed. In Tingle V4, this is used to change fragments within a shared container when in portrait mode and also update the things list when an item is added or deleted in landscape mode.

2 User interface

The current portrait UI of the main page can be seen on figure 4 and the list page in portrait can be seen on figure 5. The landscape UI can be seen on figure 6.A complete manual is added to the appendix section 6.1

Main page
The main page allows a user to add, look up or get an overview of most recent added item.

List Page
The list page gives the user a full overview of all of the added items as well as the option of deleting an item.

2.1 Usability design choices

A couple of design choices has been made to improve the usability of Tingle. Every button, text boxes and interactive widgets have been, at best, placed at the bottom of the screen so the user can comfortably reach them.

3 Functionality extensions to Tingle V3

The following functionality has been added

- Landscape split view
- Delete function

The landscape split view allows a user to see the list page as well as the main page on the same screen whenever the device is set to landscape mode. The delete function allows a user to delete a given stored item

4 Test

4.1 Unit testing of model

Automated unit tests have been performed on the model of Tingle. That is, testing the repository and database functionalities. Mockit framework has been used to create mocks when testing ThingsDatabse (See figure 3 in appendix for test results of Tingle V4).

4.2 Manual testing of View and application

Taken the current state and complexity of Tingle V4, system tests and integration tests have been performed manually. However, one may want to use Android instrumentation tests to perform automated UI tests when the complexity of Tingle increases.

5 Bugs

So far most bugs have been fixed in Tingle V4.

6 Appendix

6.1 Manual of Tingle

Adding items
To add an item one can enter the name and location of the item in the two edit boxes and press the "Add new thing" button. The recently added header will then be updated with the newly added item.

Search for item
To search for a stored item, one can write the name of that given item in the "What" edit box (not case-sensitive) and press the "look up" button. A toast will then appear denoting its location.

Accessing list page
To get to the list page in portrait mode, one can press the "show all" button or flip the screen to go to landscape mode.

Delete item
To delete an item, one can press on an entry on the list to select the desired item. A toast will appear denoting what item is currently selected. By pressing the "Delete" button, the selected item will then be removed from storage.

6.2 Class Diagrams

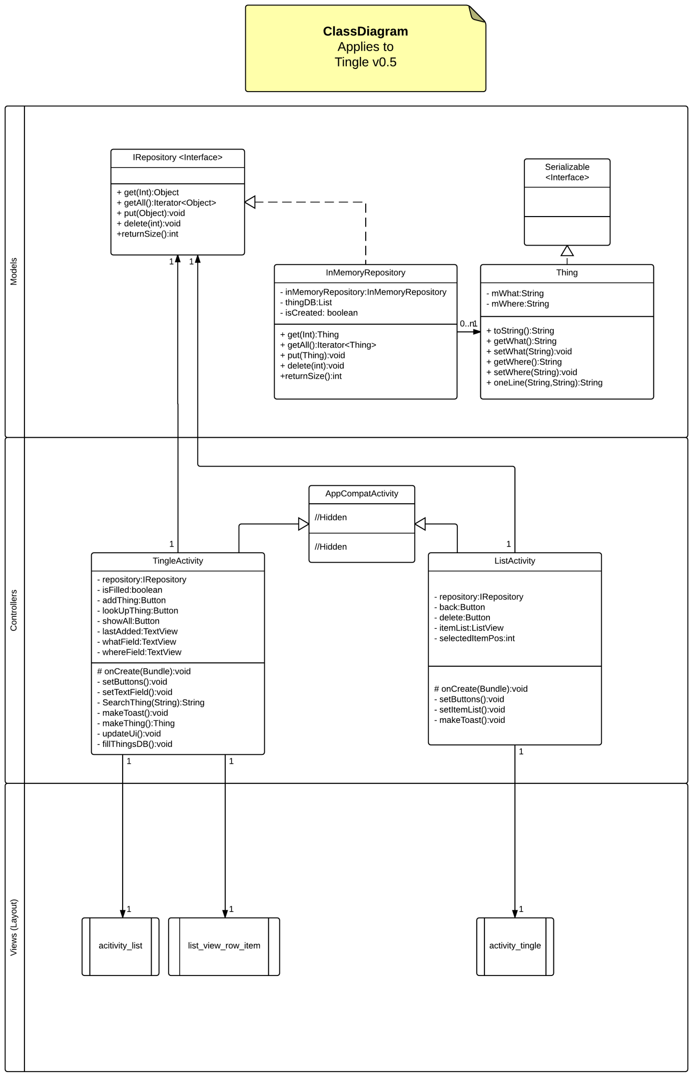


Figure 1: Class Diagram of Tingle V2

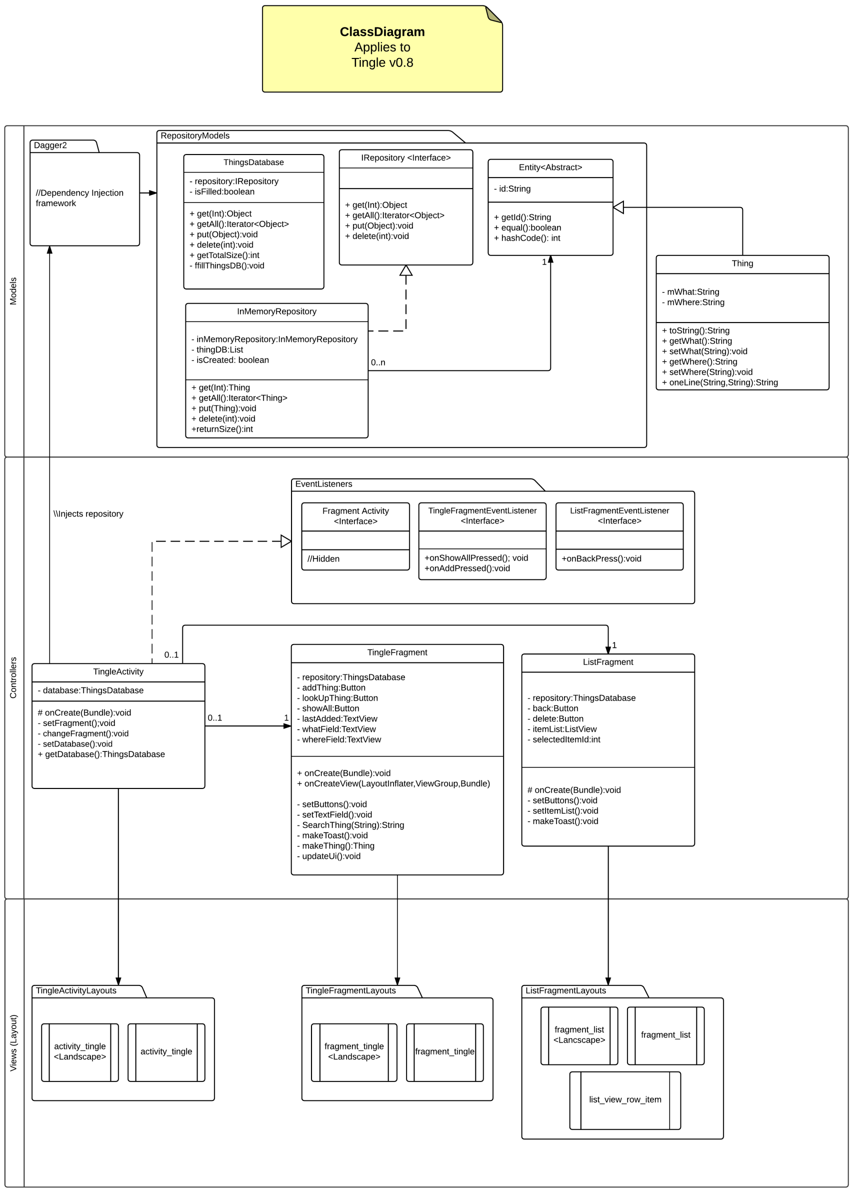
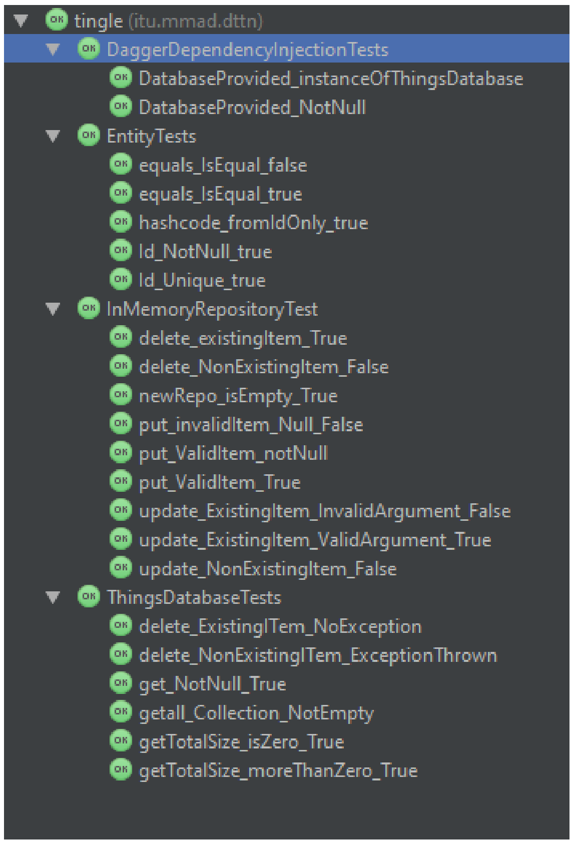


Figure 2: Class Diagram of Tingle V4

6.3 Test Results



```
▼ tingle (itu.mmad.dttn)
  ▼ DaggerDependencyInjectionTests
      DatabaseProvided_instanceOfThingsDatabase
      DatabaseProvided_NotNull
  ▼ EntityTests
      equals_IsEqual_false
      equals_IsEqual_true
      hashcode_fromIdOnly_true
      Id_NotNull_true
      Id_Unique_true
  ▼ InMemoryRepositoryTest
      delete_existingItem_True
      delete_NonExistingItem_False
      newRepo_isEmpty_True
      put_invalidItem_Null_False
      put_ValidItem_notNull
      put_ValidItem_True
      update_ExistingItem_InvalidArgument_False
      update_ExistingItem_ValidArgument_True
      update_NonExistingItem_False
  ▼ ThingsDatabaseTests
      delete_ExistingITem_NoException
      delete_NonExistingITem_ExceptionThrown
      get_NotNull_True
      getall_Collection_NotEmpty
      getTotalSize_isZero_True
      getTotalSize_moreThanZero_True
```

Figure 3: Unit test results of Tingle V4

6.4 UI pictures

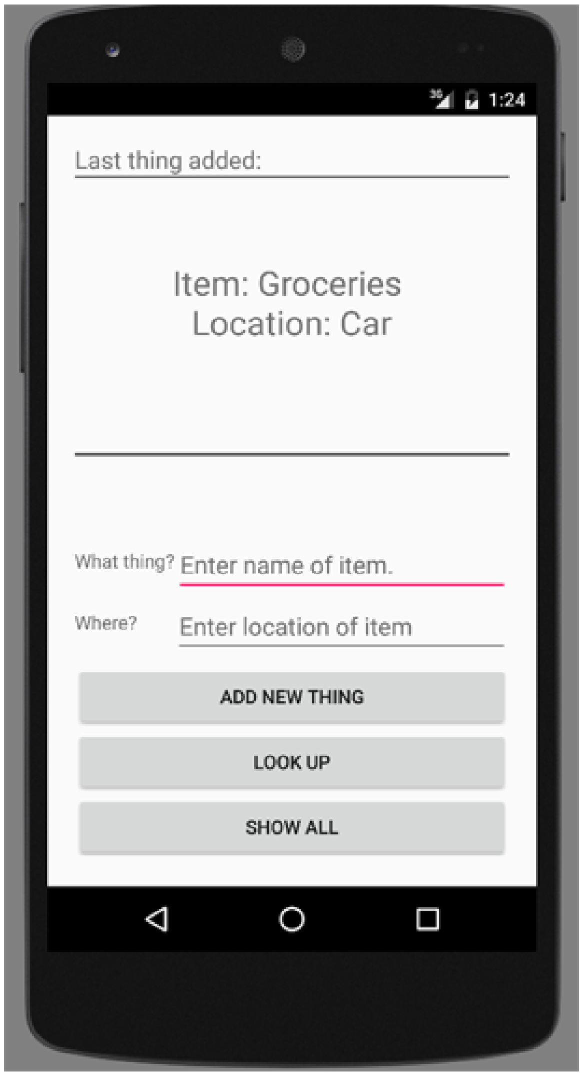
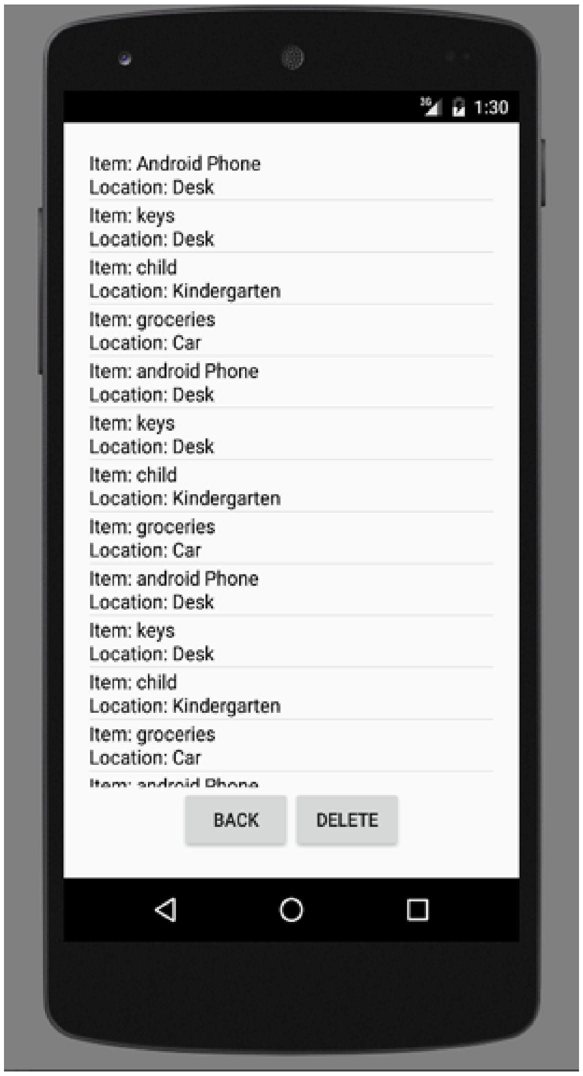


Figure 4: Main page in portrait mode



1:30

Item: Android Phone
Location: Desk

Item: keys
Location: Desk

Item: child
Location: Kindergarten

Item: groceries
Location: Car

Item: android Phone
Location: Desk

Item: keys
Location: Desk

Item: child
Location: Kindergarten

Item: groceries
Location: Car

Item: android Phone
Location: Desk

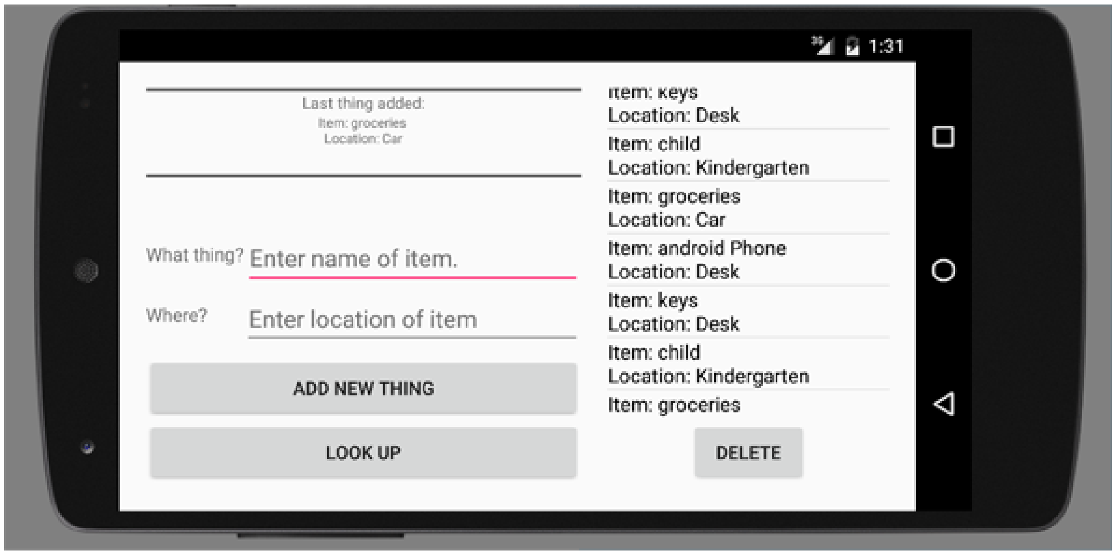
Item: keys
Location: Desk

Item: child
Location: Kindergarten

Item: groceries
Location: Car

Item: android Phone

Figure 5: List page in portrait mode



Item: keys
Location: Desk
Item: child
Location: Kindergarten
Item: groceries
Location: Car
Item: android Phone
Location: Desk
Item: keys
Location: Desk
Item: child
Location: Kindergarten
Item: groceries

Figure 6: Landscape mode