PianoPlay

Analysis and Design Document

Student: Alexandra Georgeana Czako

**Group: 30238**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

This project is an Android application. This application is used to learn how to play the piano. The main activity will be a piano. Everytime the user touches the screen on a specific piano key, he/she would pe able to hear a sound.

# Elaboration – Iteration 1.1

# Domain Model

*[Define the domain model and create the conceptual class diagrams]*

# Architectural Design

## Conceptual Architecture

**Model–view–presenter** (**MVP**) is a derivation of the model–view–controller (MVC) architectural pattern which mostly used for building user interfaces. In MVP, the presenter assumes the functionality of the “middle-man”. In MVP, all presentation logic is pushed to the presenter. MVP advocates separating business and persistence logic out of the Activity and Fragment.

[Model View Presenter](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter)

* View more separated from Model. The Presenter is the mediator between Model and View.
* Easier to create unit tests
* Generally there is a one to one mapping between View and Presenter, with the possibility to use multiple Presenters for complex Views
* Listen to user action and model updates
* Updates model and view as well
* **Model**

In an application with a good layered architecture, this model would only be the gateway to the domain layer or business logic. See it as the provider of the data we want to display in the view. Model’s responsibilities include using APIs, caching data, managing databases and so on.

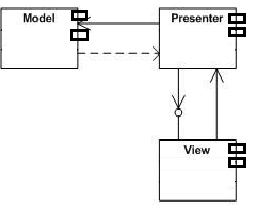
* **View**

The View, usually implemented by an Activity, will contain a reference to the presenter. The only thing that the view will do is to call a method from the Presenter every time there is an interface action.

* **Presenter**

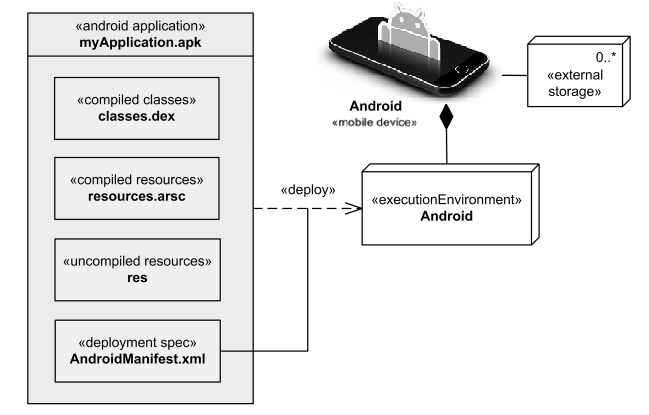
The Presenter is responsible to act as the middle man between View and Model. It retrieves data from the Model and returns it formatted to the View. But unlike the typical MVC, it also decides what happens when you interact with the View.

## Package Design



## Component and Deployment Diagrams

Deployment diagram



# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

# Data Model

*[Create the data model for the system.]*

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

One test scenario could be a simple tap on the screen. If the user hears a sound, then this functionality of the application works well.

# Future improvements

*[Present future improvements for the system]*

# Bibliography

* <https://medium.com/cr8resume/make-you-hand-dirty-with-mvp-model-view-presenter-eab5b5c16e42>