

**Android Application to test hearing impairments in adults**

by

**Junaid Malik**

This report is submitted in partial fulfilment of the requirements of the B.Sc. Honours Degree in Networking Applications & Services, of the Dublin Institute of Technology

June 2nd, 2015

Supervisor: **Dr Ray Lynch**

School of Electrical and Electronic Engineering

Year Coordinator:

Joseph Kellegher DT080B/4 Report

**Declaration**

I, the undersigned, declare that this report is entirely my own written work, except where otherwise accredited, and that it has not been submitted for a degree or award to any other university of institution.

**Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Contents

[Acknowledgments 6](#_Toc420933946)

[Abstract 7](#_Toc420933947)

[Chapter 1 Introduction 8](#_Toc420933948)

[1.1 How we hear sound? 10](#_Toc420933949)

[1.2 The Audiogram 11](#_Toc420933950)

[1.3 Pure Tone Audiometry 11](#_Toc420933951)

[1.4 Best Procedure for Pure tone audiometry 12](#_Toc420933952)

[1.5 Advantage and Disadvantage of manual self-testing 12](#_Toc420933953)

[1.6 Audiogram Interpretation 14](#_Toc420933954)

[Chapter 2 Purpose of the Project 17](#_Toc420933955)

[2.2 Goal of the Project 17](#_Toc420933956)

[2.3 Motivation 17](#_Toc420933957)

[2.4 Considerations 17](#_Toc420933958)

[2.5 Measurement 17](#_Toc420933959)

[Chapter 3 Setting up the development environment 18](#_Toc420933960)

[3.1 Development tools 18](#_Toc420933961)

[3.2 Implementing Source Control with Git 18](#_Toc420933962)

[3.3 Using Source Tree Version Control System and adding git-hub 21](#_Toc420933963)

[3.4 Android Activities 25](#_Toc420933964)

[4.5 Android life-cycle 25](#_Toc420933965)

[Chapter 4 Project Management 28](#_Toc420933966)

[4.1 User Story Mapping 29](#_Toc420933967)

[4.2 Pivotal Tracker to track the project backlog 29](#_Toc420933968)

[4.4 Android Best Practises 31](#_Toc420933969)

[Chapter 5 Requirements Analysis 32](#_Toc420933970)

[5.1 Use Case Diagram 32](#_Toc420933971)

[5.2 Class Diagrams 33](#_Toc420933972)

[Chapter 6 System Design 35](#_Toc420933973)

[6.1 UX/UI Design 36](#_Toc420933974)

[Chapter 7 System Implementation 39](#_Toc420933975)

[8.1 Implementing the Activities 39](#_Toc420933976)

[7.2 Implementing Frequency Generator 40](#_Toc420933977)

[7.3 Implementing Audiogram with Androidplot library 42](#_Toc420933978)

[7.4 Implementing Fragments 43](#_Toc420933979)

[7.5 Implementing Rate this app plug in 44](#_Toc420933980)

[Chapter 8 Testing 45](#_Toc420933981)

[8. 1 Test Structure 47](#_Toc420933982)

[8.2 Testing preconditions 48](#_Toc420933983)

[Chapter 9 Conclusion 50](#_Toc420933984)

[Appendices 51](#_Toc420933985)

[Appendix A: Code for FrequencyGen.java 51](#_Toc420933986)

[Appendix B: User Story Mapping 53](#_Toc420933987)

[Appendix C: XML code for main\_menu.xml 54](#_Toc420933988)

[Appendix D: Fragments in hearingtest\_view.xml 55](#_Toc420933989)

[1. Frequency\_fragment.xml 56](#_Toc420933990)

[2. Audiogram\_fragment.xml 57](#_Toc420933991)

[3. Java code for FragmentFrequency.java 57](#_Toc420933992)

[4. Java code for FragmentAudiogram.java 58](#_Toc420933993)

[Appendix E: Java code 59](#_Toc420933994)

[1. MainMenu.java 59](#_Toc420933995)

[2. HearingTestActivity.java 60](#_Toc420933996)

[3. MoreInfo.java 66](#_Toc420933997)

[4. MoreInfo2.java 67](#_Toc420933998)

[5. StartTest.java 67](#_Toc420933999)

[6. InfoView1.java 68](#_Toc420934000)

[7. InfoView2.java 68](#_Toc420934001)

[8. InfoView3.java 69](#_Toc420934002)

[9. InfoView4.java 69](#_Toc420934003)

[Appendix F: Log file of changes 70](#_Toc420934004)

[Appendix G: The AndroidManifest.xml 72](#_Toc420934005)

[Appendix H: Strings Resource file 74](#_Toc420934006)

[Appendix I: Layout Files 76](#_Toc420934007)

[1. calibration\_info.xml 76](#_Toc420934008)

[2. infoview1.xml 77](#_Toc420934009)

[3. infoview2.xml 77](#_Toc420934010)

[4. infoview3.xml 78](#_Toc420934011)

[5. infoview4.xml 78](#_Toc420934012)

[6. loadtest.xml 79](#_Toc420934013)

[Appendix J: Open Source Libraries 79](#_Toc420934014)

[1. SimpleXYSeries.java by AndroidPlot developers 79](#_Toc420934015)

[2. RateThisApp.java by Keisuke Kobayashi 84](#_Toc420934016)

[Bibliography 87](#_Toc420934017)

**Tables**

**Table 1‑1 Grade of impairment** 8

**Table 2‑1 Noise levels during audiometric testing: Accordance with ANSI S3.1-1991.** 12

**Figures**

[Figure 1 Diagram of ear. 9](#_Toc420933915)

[Figure 2 Typical Audiogram 10](#_Toc420933916)

[Figure 3 Typical activity flow of hearing test 12](#_Toc420933917)

[Figure 4 The audiogram shows loudness and examples of their environments 13](#_Toc420933918)

[Figure 5 Presbyacusis audiogram 14](#_Toc420933919)

[Figure 6 Noise induced audiogram 14](#_Toc420933920)

[Figure 7 Symmetrical hearing loss 15](#_Toc420933921)

[Figure 8 Git- Hub Client Application 18](#_Toc420933922)

[Figure 9 Git-Hub Bash module 18](#_Toc420933923)

[Figure 10 Checkout style using Unix style 19](#_Toc420933924)

[Figure 11 Initial commit of project to git. 19](#_Toc420933925)

[Figure 12 Git config commands 22](#_Toc420933926)

[Figure 13 Pushing updates to repository 23](#_Toc420933927)

[Figure 14 Sprint cycle 27](#_Toc420933928)

[Figure 15 Progress report of project 28](#_Toc420933929)

[Figure 16 Backlog stories and backlog items 29](#_Toc420933930)

[Figure 17 Gantt chart of project backlog items 30](#_Toc420933931)

[Figure 18 Use-Case Diagram of main menu 31](#_Toc420933932)

[Figure 19 Simple class diagram. 32](#_Toc420933933)

[Figure 20 Class diagram obtained from analysis 33](#_Toc420933934)

[Figure 21 Detailed class diagram of system 33](#_Toc420933935)

[Figure 22 A professional audiometer used in clinics 34](#_Toc420933936)

[Figure 23 the architecture of the apk. 35](#_Toc420933937)

[Figure 24 Quick layout design created of UI 35](#_Toc420933938)

[Figure 25 Initial mock design of the main menu 36](#_Toc420933939)

[Figure 26 Final design of the main menu 37](#_Toc420933940)

[Figure 27 Diagram of how the methods are called for each button 38](#_Toc420933941)

[Figure 28 Hierarchy of XYPlot 41](#_Toc420933942)

[Figure 29 XYPlot template created from XML 41](#_Toc420933943)

[Figure 30 Architecture of Testing in Android 46](#_Toc420933944)

[Figure 31 Creating the JUnit test case 48](#_Toc420933945)

# Acknowledgments

I would like to thank Dr Ray Lynch, my project supervisor who thought me Mobile Development providing a very important understanding of application design and architecture and also helped me and gave me guidance during implementation of the project.

Secondly I would like to thank Paula Kelly for introducing me to Java, providing me with excellent knowledge that has been crucial not only for this project but has helped me have a greater overall understanding of Software Design.

Lastly I would like to thank the School of Electronics and Communications Engineering who gave me a chance to enter this course and to receive study from great professionals.

Also would like to thank my family and friends who encouraged me and supported in my effort to develop this project.

# Abstract

Hearing is an important ability to human beings in which we perceive sounds. Sound waves occur over a wide spectrum of frequencies. The human ear is sensitive to a frequency band within that spectrum that is expressed in decibels (dB). Frequencies capable of being heard by humans are called audio or sonic. The range is typically considered to be between 20 Hz and 20,000 Hz (Hertz). Frequencies higher than audio are referred to as ultrasonic, while frequencies below audio are referred to as infrasonic. A loss of the ability to hear sound frequencies in the normal range that is specified is called hearing impairment.

# Chapter 1 Introduction

A hearing test provides a means to diagnose an individual’s hearing impairment or deficiencies. ISO 8253-1:2010 is an international standard for PTA (Pure Tone Audiometry). This standard specifies procedures and requirements for PTA air conduction and bone conduction. Pure tone audiometry is the measurement of an individual’s hearing across a range of test frequencies using a standardized test method that specifies the procedure for determining the threshold, the range of test frequencies and presentation levels, and the way thresholds are presented in a graphical manner and including the symbols user to depict objects. We use PTA to evaluate possible hearing losses and to determine the type of hearing loss that an individual may have.

**Table 1‑1 Grade of impairment**

|  |  |  |
| --- | --- | --- |
| **Grade of impairment** | **Audiometric ISO value (500, 1000, 2000, 4000)** | **Impairment Description** |
| 0 (No impairment) | 25 dBHL | No or very slight hearing problems |
| 1 (Slight impairment) | 26-40 dBHL | Able to hear and repeat spoken words in normal voice |
| 2 (Moderate impairment | 41-60 dBHL | Able to hear but raised voice |
| 3 (Sever impairment) | 61-80 dBHL | Able to hear some words shouted |
| 4 (Profound impairment) | 81 dBHL | Unable to hear and understand a shouted voice |

The above table shows grade of impairment, level of decibels and the impairment description corresponding to each grade. [7]

Hearing loss can be defined as the amount a person’s hearing level changes as a result of some adverse influence. This means that some structure or function of the ear that is crucial to hearing has been damaged.

There are three main forms of hearing loss; Sensor neural affecting the cochlea, Conductive hearing loss affecting the ear canal and mixed hearing loss which is a combination of both sensor neural and conductive hearing loss.

The World Health Organization (WHO) defines “disabling hearing impairment in adults as a permanent unaided hearing threshold level (average for frequencies 0.5, 1, 2, 4 kHz) for the better ear of 41 dB or greater (WHO, 2001).2 In children under 15 years of age, disabling hearing impairment is defined as permanent unaided hearing threshold level (average for frequencies 0.5, 1, 2, 4 kHz) for the better ear of 31 dB or greater.” [7]

The auditory pathway includes the external ear, the middle ear and the inner ear, followed by the auditory nerve ending up in the auditory centres in the auditory cortex.

* The external ear consists of the pinna, ear canal and eardrum. Sound travels down the ear canal, through the eardrum and causing it to move or vibrate.
* The middle ear is a space behind the eardrum that contains three small bones called ossicles*.* This string of tiny bones is connected to the eardrum at one end and to the oval window at the other end which connects to the inner ear. Vibrations from the eardrum cause the ossicles to vibrate which, in turn, creates movement of the fluid in the inner ear.
* Movement of the fluid in the inner ear, or cochlea, causes changes in hair cells*.* This movement of the hair cells sends electric signals from the inner ear up the auditory nerve to the brain. [7]

The brain then interprets these electrical signals as sound. Figure 1 - shows the different areas of the ear described below. [11]

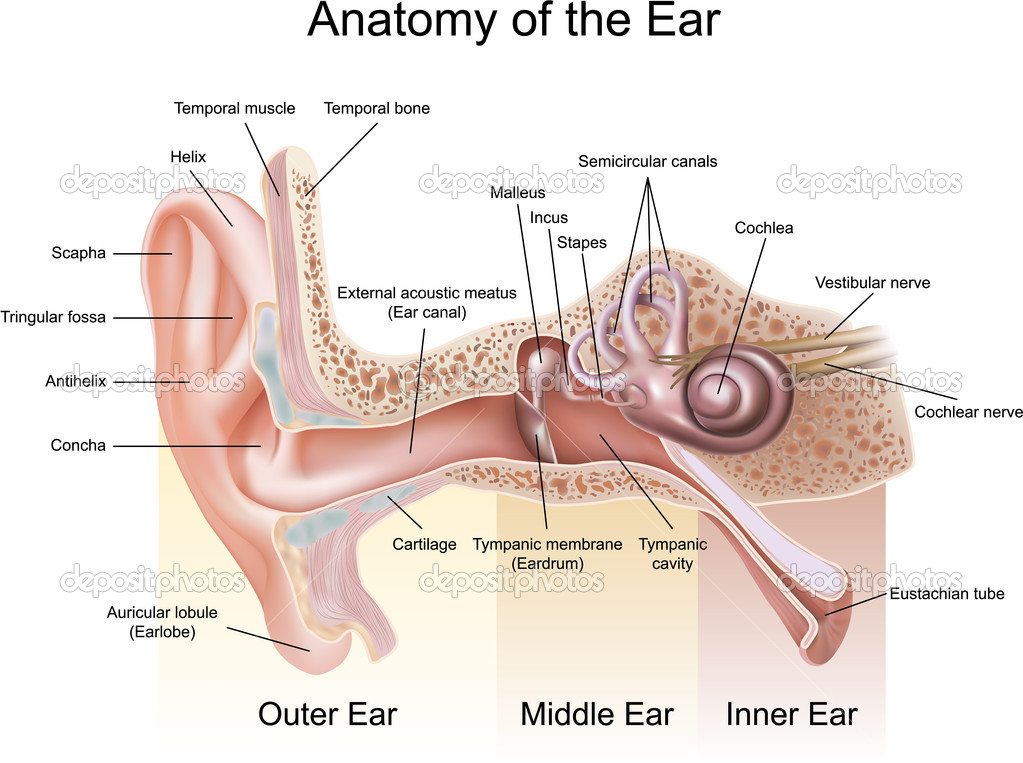


Figure 1 Diagram of ear.

From the above diagram we can see the middle ear this is where most conductive hearing loss occurs this is treated by medical treatment or through surgery. Sensor neural hearing loss occurs in the inner ear as the nerves become damaged and do not properly transmit their signals to the brain. This happens become of aging of the inner ear cells. Sensor neural hearing loss is the most common type hearing loss among adults. It is not often medically or surgically treatable. This type of hearing loss is can be treated with the help of hearing aids. When hearing loss is conductive and Sensor neural this is called mixed hearing loss.

## 1.1 How we hear sound?

Sound waves enter the outer ear and through the ear canal. The eardrum vibrates from incoming sound waves and sends those vibrations to bones in middle malleus, incus and stapes. The sound vibrations go from the air to fluid vibrations in cochlea of the inner ear. Once fluid inside cochlea vibrates waves form along the basilar membrane. When hair cells move, the bending of the hair cells causes chemicals to create an electrical signal. The auditory nerve than carries this signal to the brain which then is translated into a sound that we hear and understand. [15]

## 1.2 The Audiogram

The audiogram plots decibel (dB) values on the Y-Axis on an audiogram. If a loss of hearing is present then the graph will have higher points with increased dB signal where the person is having difficulty hearing the tone. The more severe the hearing loss the more the dB values on the graph will going downwards. On the X-Axis we have the plotting of the frequency pure tones in hertz (Hz). An Audiogram usually plots sound on the left lowest 125Hz and increasing to 8000 KHz.

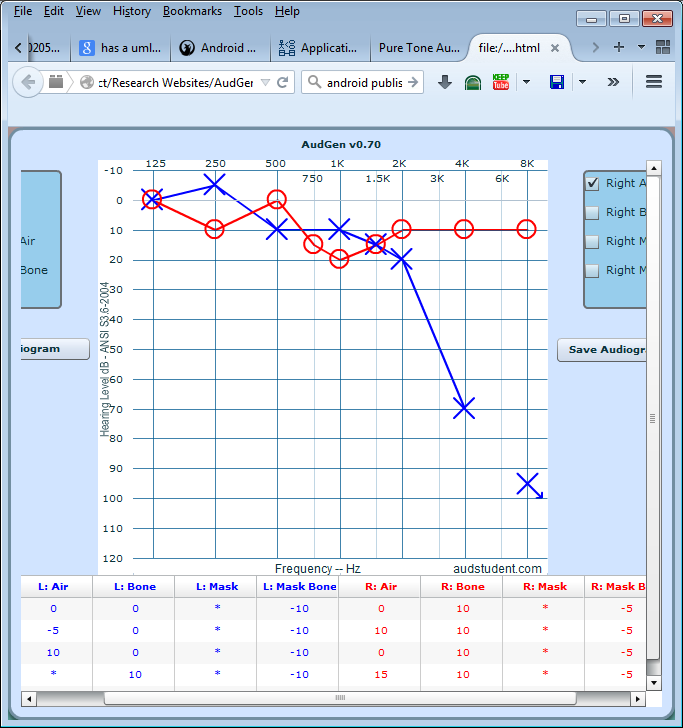


Figure 2 Typical Audiogram

## 1.3 Pure Tone Audiometry

Hearing thresholds of humans can be found by using behavioural hearing tests or physiological tests. An audiogram can be obtained using a behavioural hearing test called Audiometry. For humans the test involves different tones being played at a specific frequencies (pitch) and intensity (loudness). When this test is done with a doctor, the person hears the sound they raise their hand or press a button so that the tester knows that they have heard the sound. The lowest sound the person hears is recorded. The entire auditory system from ear canal to auditory cortex is tested. When a test is decided a method of selecting the correct method has to be chosen. There is three main methods manual, self-recording and microprocessor. I decided to choose manual audiometry method for the design of the application as it is most practical as the user decides the test environment. However this does not make the application design the most scientifically accurate. In the manual audiometry the frequency and intensity are controlled by the user. When hearing is tested the frequencies of the range of 125Hz to 8000 Hz are usually tested. One of the most important things in audiometry is the environment in which the subject is undertaking the test. There are different standards depending on the region which require different calibration.

**Table 2‑1 Noise levels during audiometric testing: Accordance with ANSI S3.1-1991.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Octave-Band Centre Frequency | 500 Hz | 1000 Hz | 2000 Hz | 4000 Hz | 8000 Hz |
| ANSI S3.1-1991 (Rounded to the nearest whole decibel) | 22 | 30 | 34 | 42 | 45 |
| OSHA Table D-2 | 27 | 30 | 32 | 42 | 45 |
| OSHA Table D-1 | 49 | 40 | 47 | 57 | 62 |

## 1.4 Best Procedure for Pure tone audiometry

When beginning a test it is best to start with the right ear but the user decides what ear he wants checked. A pulsing tone that goes on and off is usually recommended; in the design of the application the user presses a button to hear the tone. The optimal test environment is quiet and free from anything that could distract user. Sometimes it is useful to wear noise-reducing earphones for best performance but regular earphones can also be used.

## 1.5 Advantage and Disadvantage of manual self-testing

The advantage of self-testing is that it can be done regularly and quickly as no specialised equipment is required. On the other hand the disadvantage is that with manual testing a user can make mistakes, press wrong button or test same ear twice. This can introduce errors into the audiogram which can make it useless from a scientific view. There are many things that can go wrong with accurate results example; too much background noise, earphones not placed properly and tester bias when a person cannot hear.



Figure 3 Typical activity flow of hearing test

From the above a typical hearing test consists of the following choices. A user selects if we can hear tone or cannot as the option is selected the result is recorded on a graph, in this case an Audiogram. Audiograms help us to explain audiometric readings.

## 1.6 Audiogram Interpretation

In order to interpret audiogram there are usually certain characteristics that explain the results of the audiogram. These characteristics can determine the type of hearing loss.

Below are some examples of audiograms and sample interpretation.

* Monaural loss: Loss is only in one ear.
* Binaural loss: Loss is in both ears.
* Symmetrical: Hearing is relatively uniform in both ears.
* Asymmetrical: Hearing loss in one ear is significantly worse than the other ear.

The shape of the audiogram is also an important factor to determining the type of loss. There is flat loss, sloping loss and other shapes show different aging and noise damage on the audiogram.

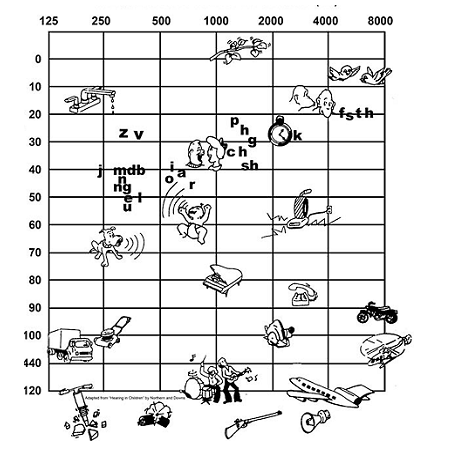


Figure 4 The audiogram shows loudness and examples of their environments

The above is a good example of the different scenarios and their associated decibel levels that a person can imagine. With a low sound level at 10 dBs the chirping of birds and at 120 dBs a pneumatic drill and a jet aircraft. [20]

Presbyacusis is an age related hearing loss. Affects high frequencies much more than lower. From the above we can see that sounds have to be made louder before the person can hear them.

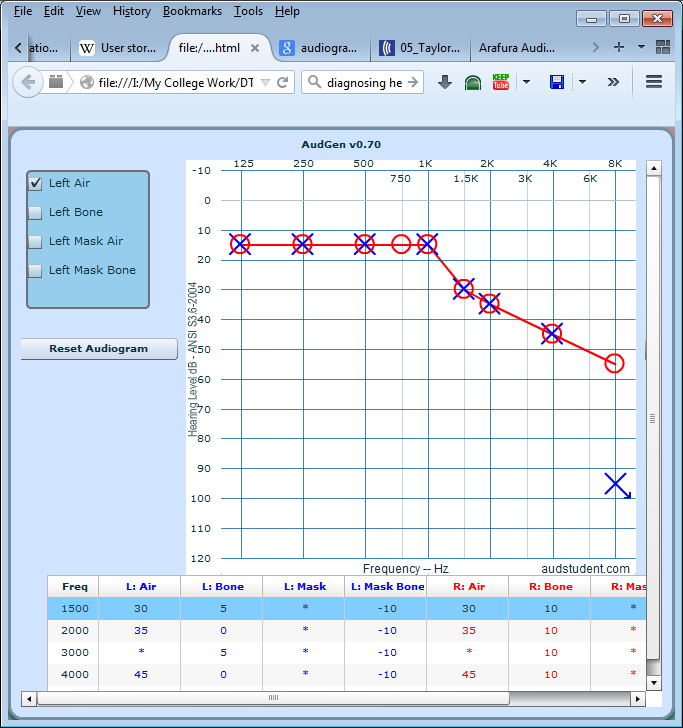


Figure 5 Presbyacusis audiogram

Noise induced hearing loss is where a hearing organ has been damaged for example the cochlea.

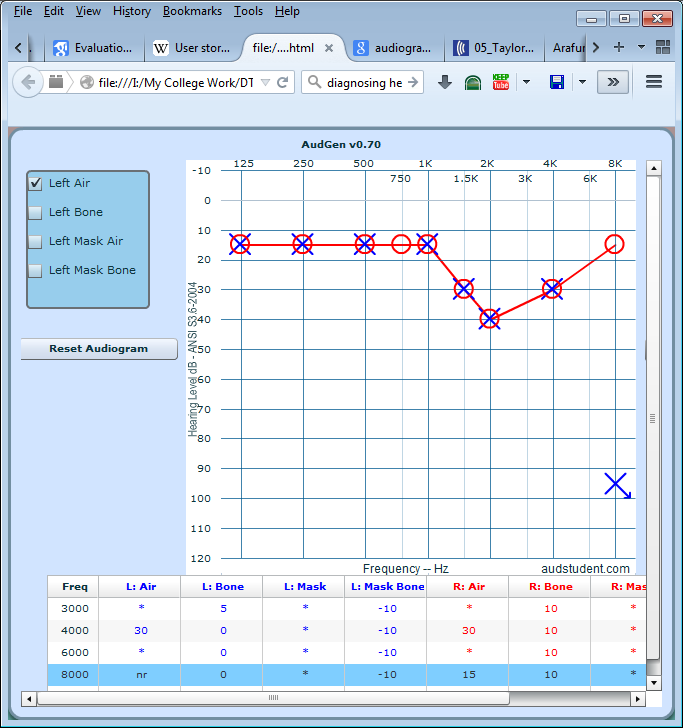


Figure 6 Noise induced audiogram

Symmetrical hearing loss is when both ears have same hearing loss.

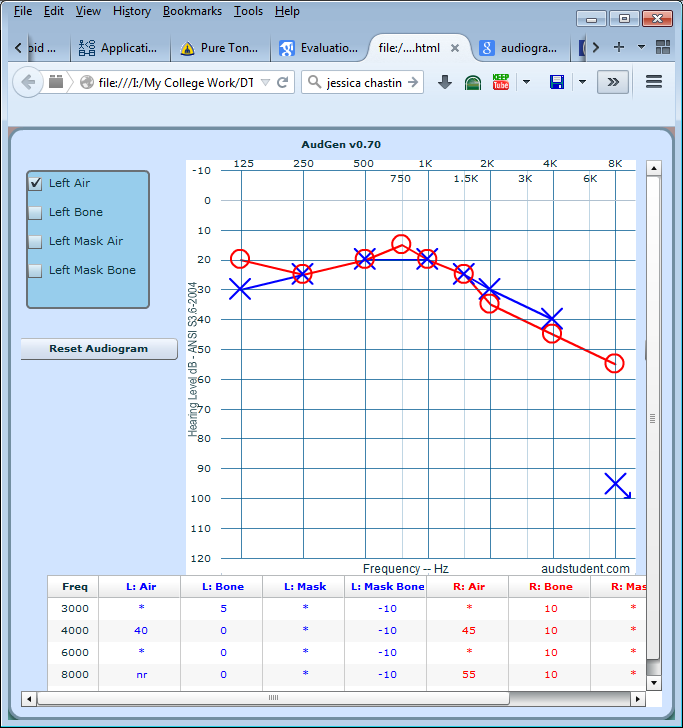


Figure 7 Symmetrical hearing loss

# Chapter 2 Purpose of the Project

The purpose of this project was to develop an application that would help identify common hearing impairment by using an audiogram which would picture how a person hears. The audiogram is used to describe the hearing of a person across different frequencies. It can be used as a tool to determine amount of damage done or determine the cause.

## 2.2 Goal of the Project

The goal was to develop and application to determine hearing impairments in adults.

## 2.3 Motivation

An easy and simple application that will allow users to quickly do hearing tests for an assessment, where they get results by reading the audiogram.

## 2.4 Considerations

The problem of hearing impairment is an increasing problem across most populations of the world and is something that people should be aware of and by taking hearing tests they can be made more aware of their own level of hearing and how to best keep it healthy. In the EU it is estimated that more than 55 million people are having hearing impairments and the costs in the EU is estimated to 160 billion Euros per year [10]. According to a study, a mild hearing loss costs 2000 euros per individual each year, a moderate costs 6,600 and severe 11,000 euros.[9]

From the above studies we can see that it is a problem that needs to be carefully analysed and more research needs to be done.

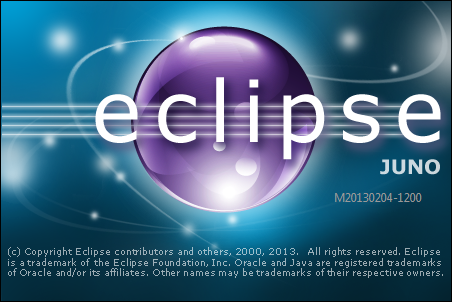
## 2.5 Measurement

The advantage brought to users through this application is that it provides free and quick self-test ability. Where-as many Android applications or web applications require connection to the internet using databases, this current project does all processing locally on the users device.

# Chapter 3 Setting up the development environment

## 3.1 Development tools

The integrated development environment (IDE) I will be using is Eclipse Juno, which is used to develop Android Applications using Java to develop the application. For the project management aspect of the project I will be using MS Visio for UML diagrams and UI Wire-framing Tool Pencil for drawing design and test user interfaces. The test tones can be generated using the android Tone Generator class. To comply with ISO 8253 -1:2010 I made be making use of an open source graph plotting library called AndroidPlot to generate the audiogram [5].



There are many different versions of Eclipse and a newer Android Studio was released, however I continued using Eclipse Juno as it is the same as Android Studio which mainly has a graphical and performance upgrade for newer PC’s. However a switch to Android Studio is simple as all that is required is importing the project into Android Studio.

## 3.2 Implementing Source Control with Git

Version control is one of the most important aspects of a software project. It’s an essential part of development where we can process the management of several files and different revisions of those files. The version control system provides us with a central hub where the code is located, it maintains a history of the work on the project, central place for backup and making separate copies of the project is not required and it also allows reversibility, meaning we can reverse changes [12]. For this project I learned to use Git-Hub where I created my own repository and licensed my project under the Apache license. In order to implement source control into the project a number of steps had to be taken. First the installation of the git-hub client for windows was required.



Figure 8 Git- Hub Client Application

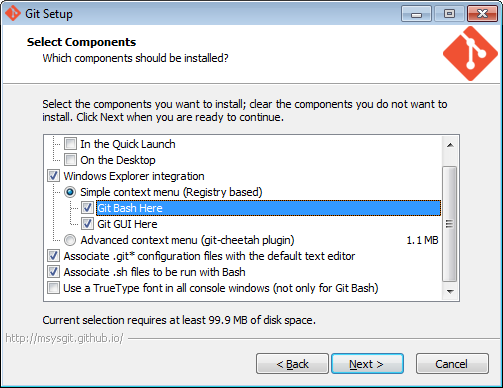


Figure 9 Git-Hub Bash module

I chose the Bash module as I am familiar with Bash scripting and chose the associated extensions for git files.

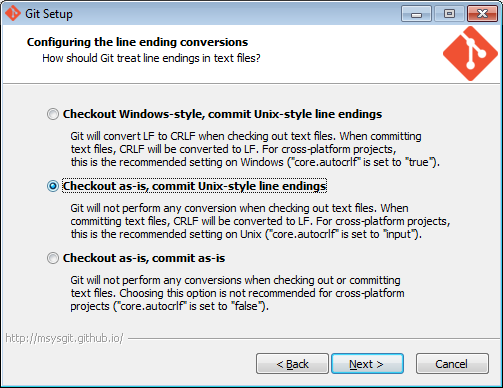


Figure 10 Checkout style using Unix style

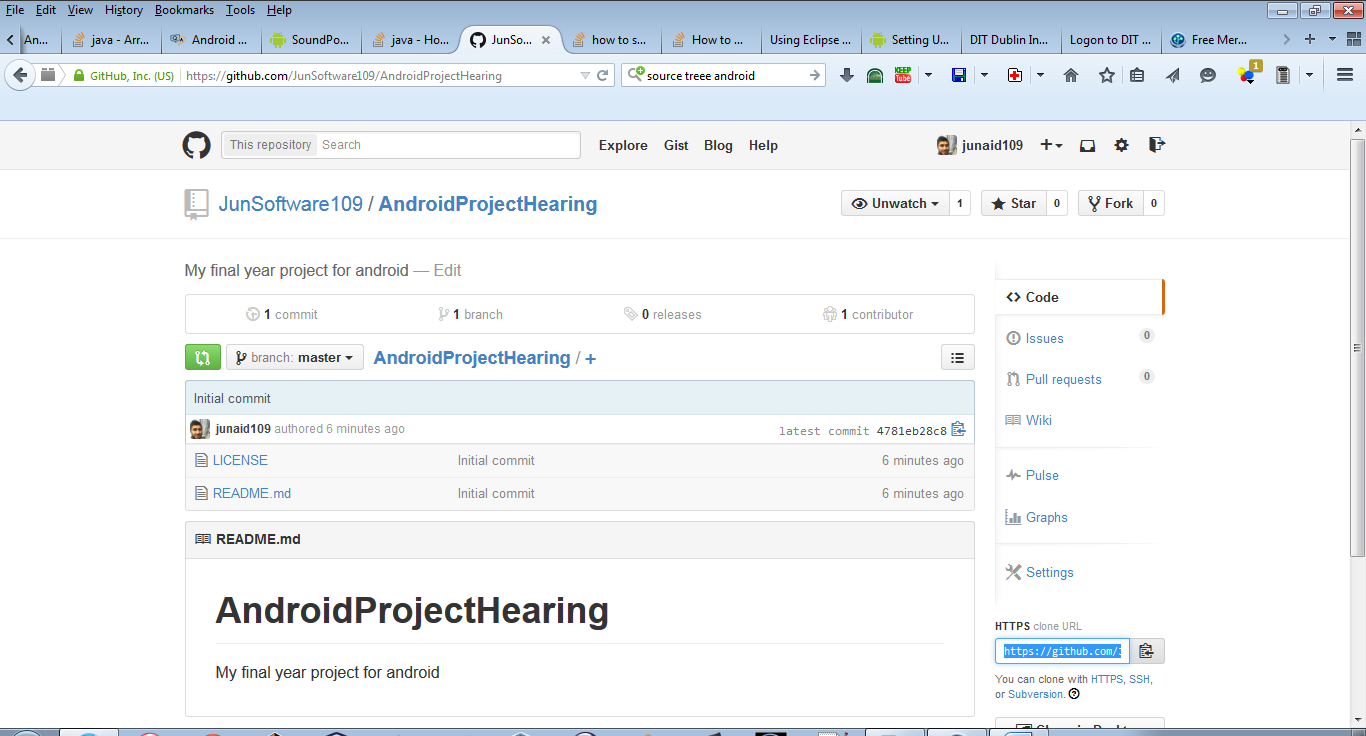


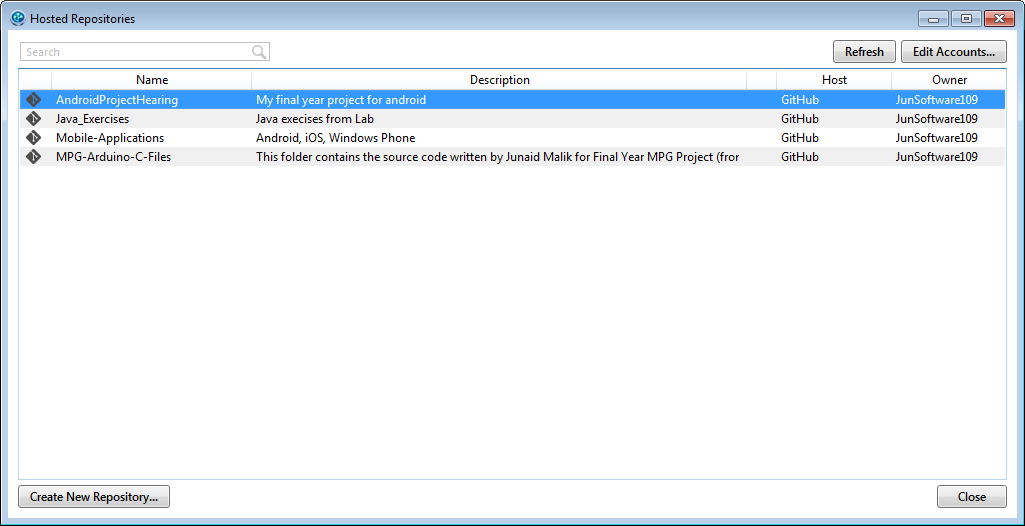
Figure 11 Initial commit of project to git.

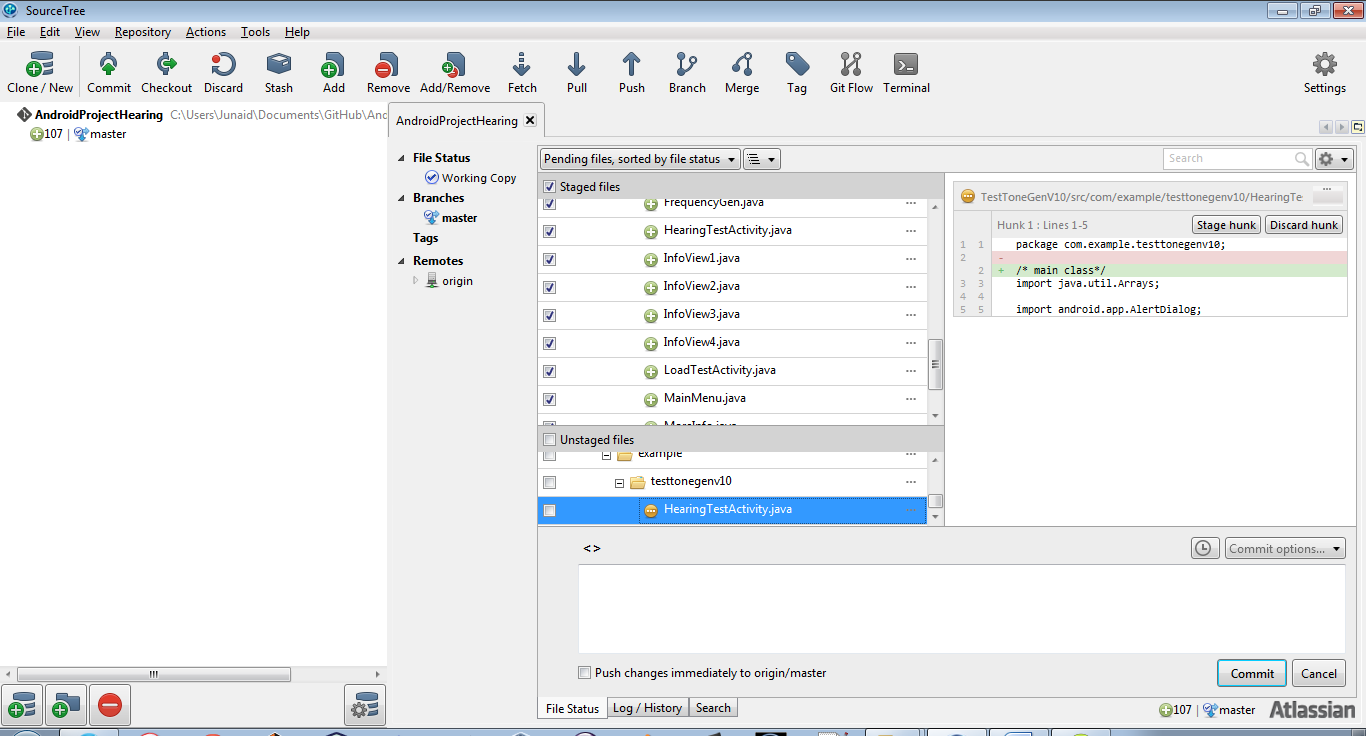
The repository is created along with license.

## 3.3 Using Source Tree Version Control System and adding git-hub

I used Source Tree by Atlassian for version control which is a very powerful VCS. These stores all the project files onto Git including source code files, documentation, build files configuration files, assets and libraries. Once I

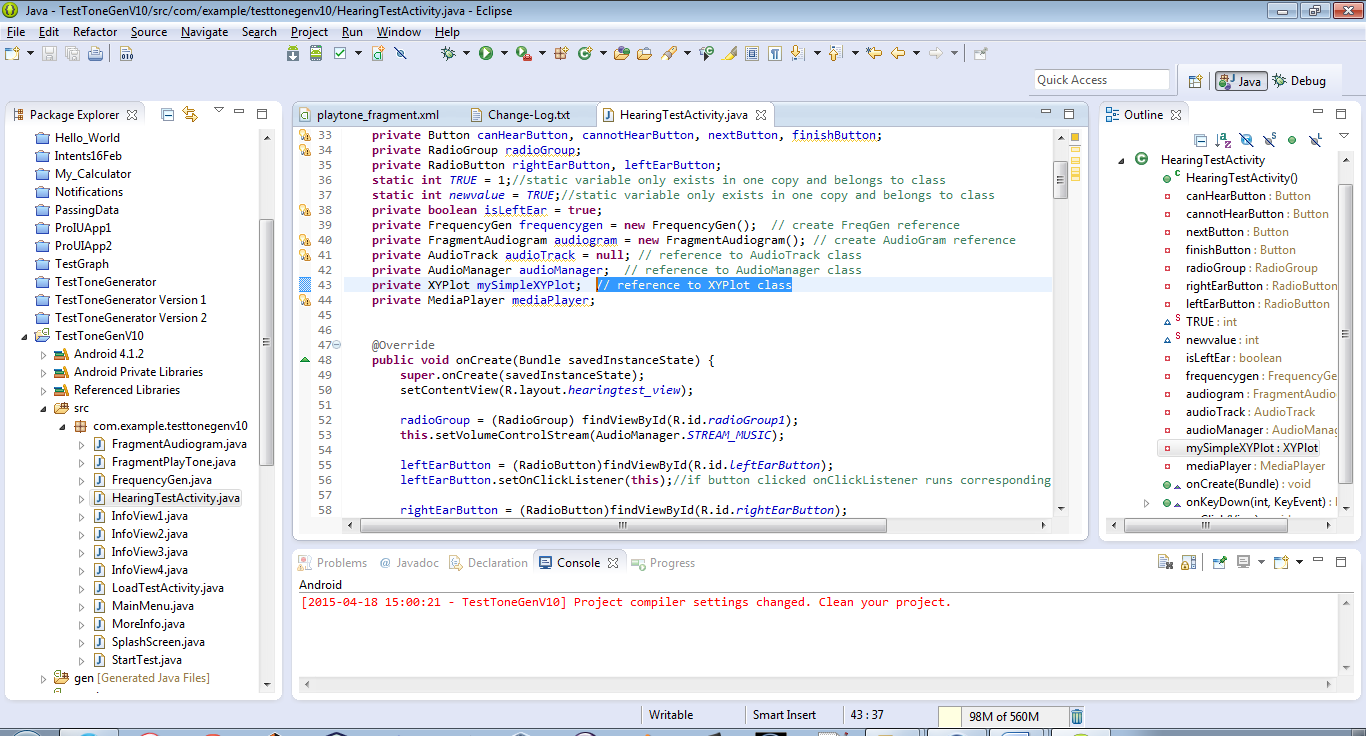
SourceTree Logo[16]



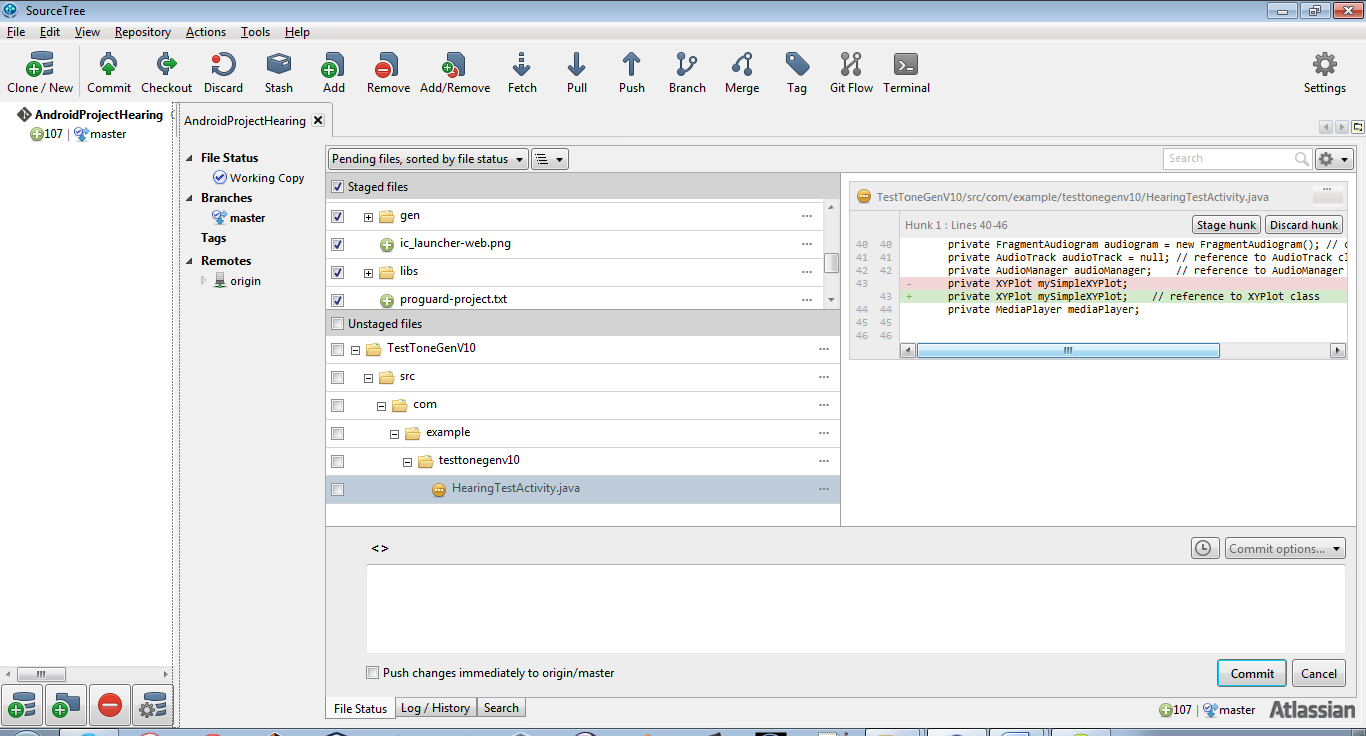
Once I have installed source tree, I linked the git hub account

count and repository,

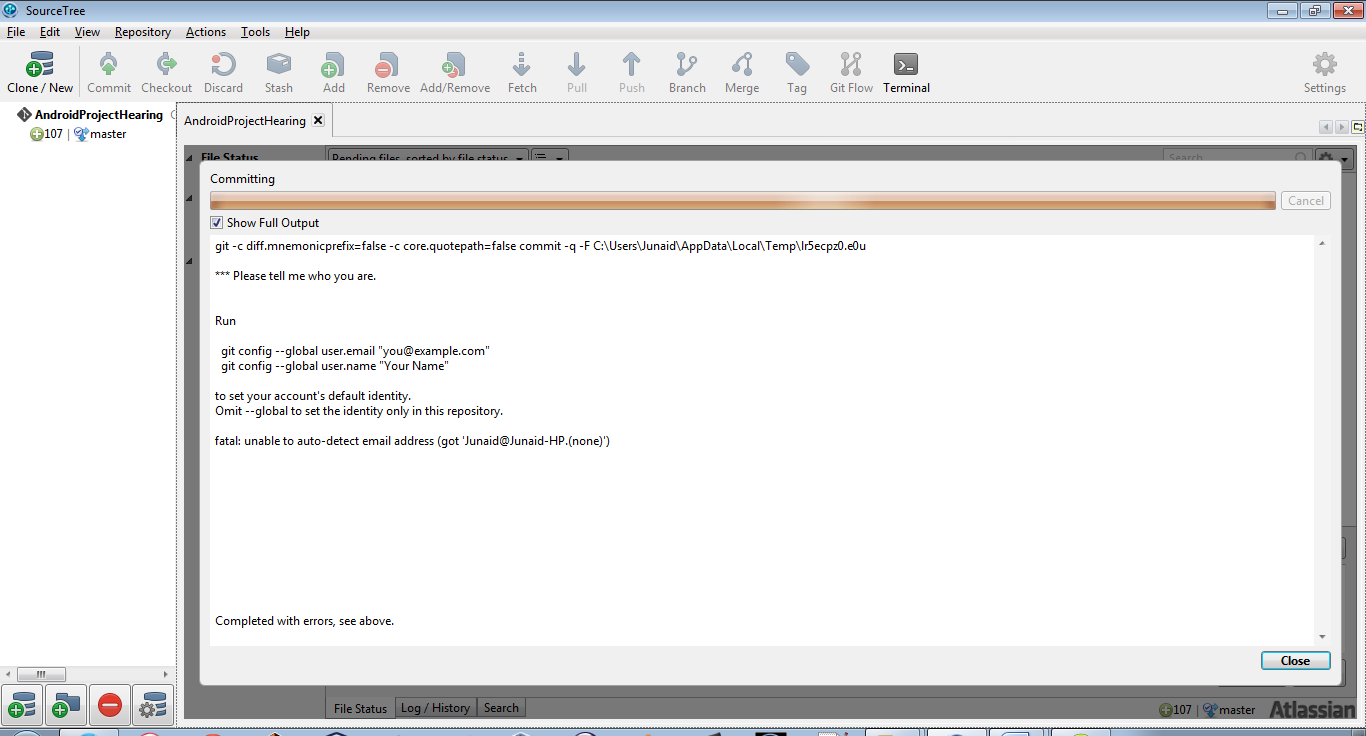
The project is added once we create a copy of it in folder where the version control



We add some comments to HearingTestActivity class.



Any changes made to code are saved in source tree and we can see from the above that highlighted code in green is what we added,



Once happy with changes made we can commit changes made.

To commit changes we have to install Git for windows this adds the command line tools to run git.

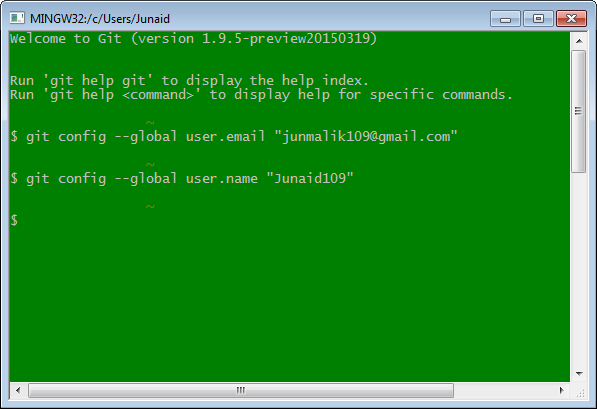


Figure 12 Git config commands

Once I set the git config commands in git bash shell we can commit any changes made in source tree

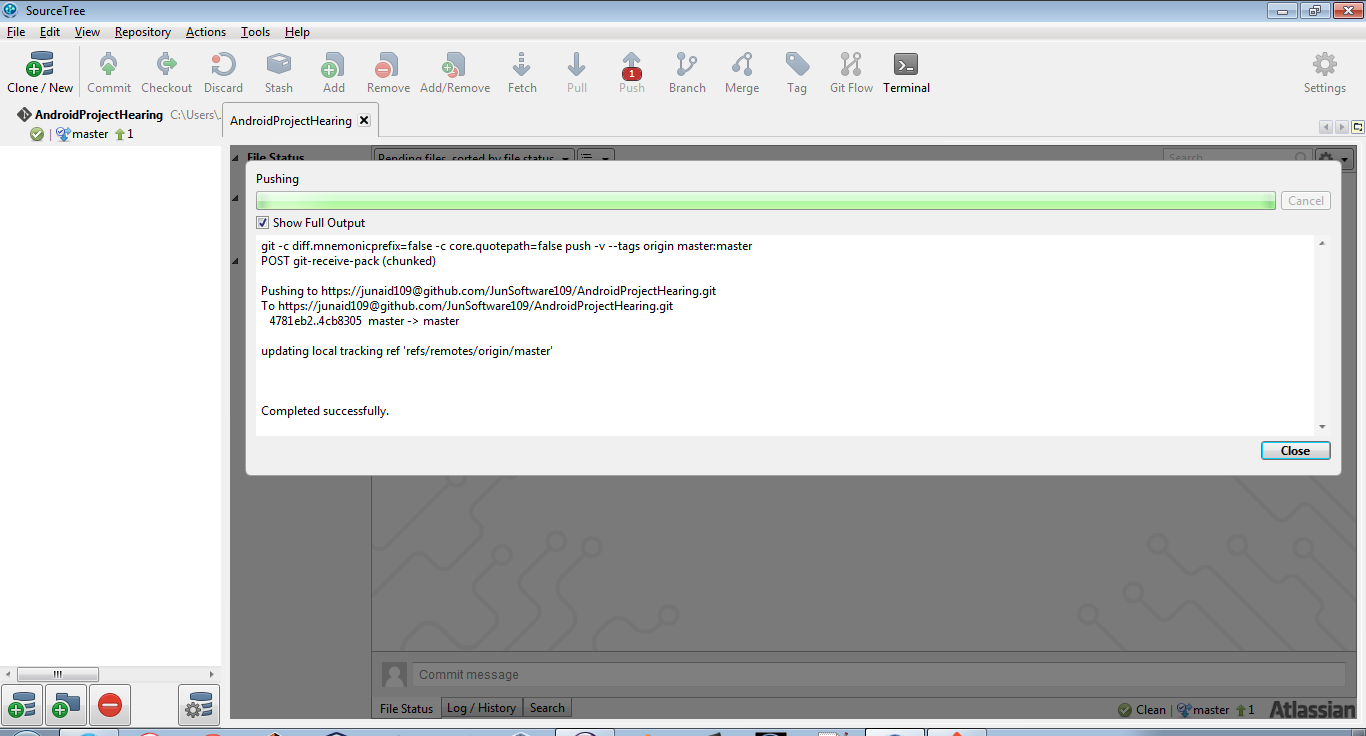


Figure 13 Pushing updates to repository

The files are then added to the main git repository.

## 3.4 Android Activities

An activity represents a single screen that the user sees on the device. An application usually consists of multiple activities. Activities are the most observable part of the application. In Android, you can be looking at an activity of one application, but shortly after you could start another activity in a completely separate application. For example, if you are in the Calendar application and you decide to call a friend, you would be launching the activity to bring up the phone application in the Calendar application.

## 4.5 Android life-cycle

Activity can be expensive on the device CPU and RAM. It can sometime involve creating a new Linux process, allocating memory for all the new objects, inflating the objects from XML layouts, and setting up the screen. In Android the activity life cycle is managed by the Activity Manager.

Activity Manager is responsible for creating, destroying, and managing activities. For example, when the user starts an application for the first time, the Activity Manager will create its activity and put it onto the screen. Later, when the user switches screens, the Activity Manager will move that previous activity to a holding place. This way, if the user wants to go back to an older activity, it can be started more quickly. Older activities that the user hasn’t used in a while will be destroyed in order to free more space for the currently active one. This mechanism is designed to help improve the speed of the user interface and thus improve the overall user experience.

Programming for Android is conceptually different than programming for some other environments. In Android, you find yourself responding more to certain changes in the state of your application rather than driving that change yourself. It is a managed, container-based environment similar to programming for Java applets or servlets. So, when it comes to an activity life cycle, you don’t get to say what state the activity is in, but you have plenty of opportunity to say what happens during the transitions from state to state. The figure below shows the states that an activity can go through. [13]



Managing the lifecycle of activities by implementing call-back methods is crucial to developing a strong and flexible application. The lifecycle of an activity is directly affected by its association with other activities,.

Call-back methods are essentially in the three Activity states:

Resumed

The activity is in the foreground of the screen and has user focus. (This state is also sometimes referred to as "running".)

Paused

Another activity is in the foreground and has focus, but this one is still visible. That is, another activity is visible on top of this one and that activity is partially transparent or doesn't cover the entire screen. A paused activity is completely alive (the Activity object is retained in memory, it maintains all state and member information, and remains attached to the window manager), but can be killed by the system in extremely low memory situations.

Stopped

The activity is completely obscured by another activity (the activity is now in the "background"). A stopped activity is also still alive (the Activity object is retained in memory, it maintains all state and member information, but is not attached to the window manager). However, it is no longer visible to the user and it can be killed by the system when memory is needed elsewhere.

# Chapter 4 Project Management

In this project I chose an iterative approach towards development. With the UX design, UI design and software development stages all iterative pieces of each sprint cycle. Bit by bit each feature was added. With an iterative approach I was able to focus on each small piece and create a working bit. My aim was to have a working piece of software by every week; this further enabled clarity and gave me quick feedback by demonstrating different version to peers or friends.



Figure 14 Sprint cycle

Each piece of the development cycle was broken down into 1 to 2 weekly sprints which consisted of adding functionality to each part of the project. I used user story mapping which helped draw out the project, this acted as a visual guide to the project, which solidified my understanding of the requirements and if any changes t occurred were easy foreseeable and changeable.

## 4.1 User Story Mapping

User story mapping it a practise in that’s helps us create a visual structure of release planning for each item in the backlog. The activities in the map are arranged in order of priority. The story map creates version of the software that we can visualise and see what parts will affect other parts. This approach helps with incremental delivery and creates deadlines. Refer to image of Appendix B.

User story is description in one or two sentences as to what the end user of the system wants; it’s an effort to capture requirements. In practise it is done on simple sticky notes.

## 4.2 Pivotal Tracker to track the project backlog

The main project management software I used for this project was pivotal tracker, this is a very powerful tool that helps with task management, velocity tracking, iteration planning and also produces progress charts. [18]

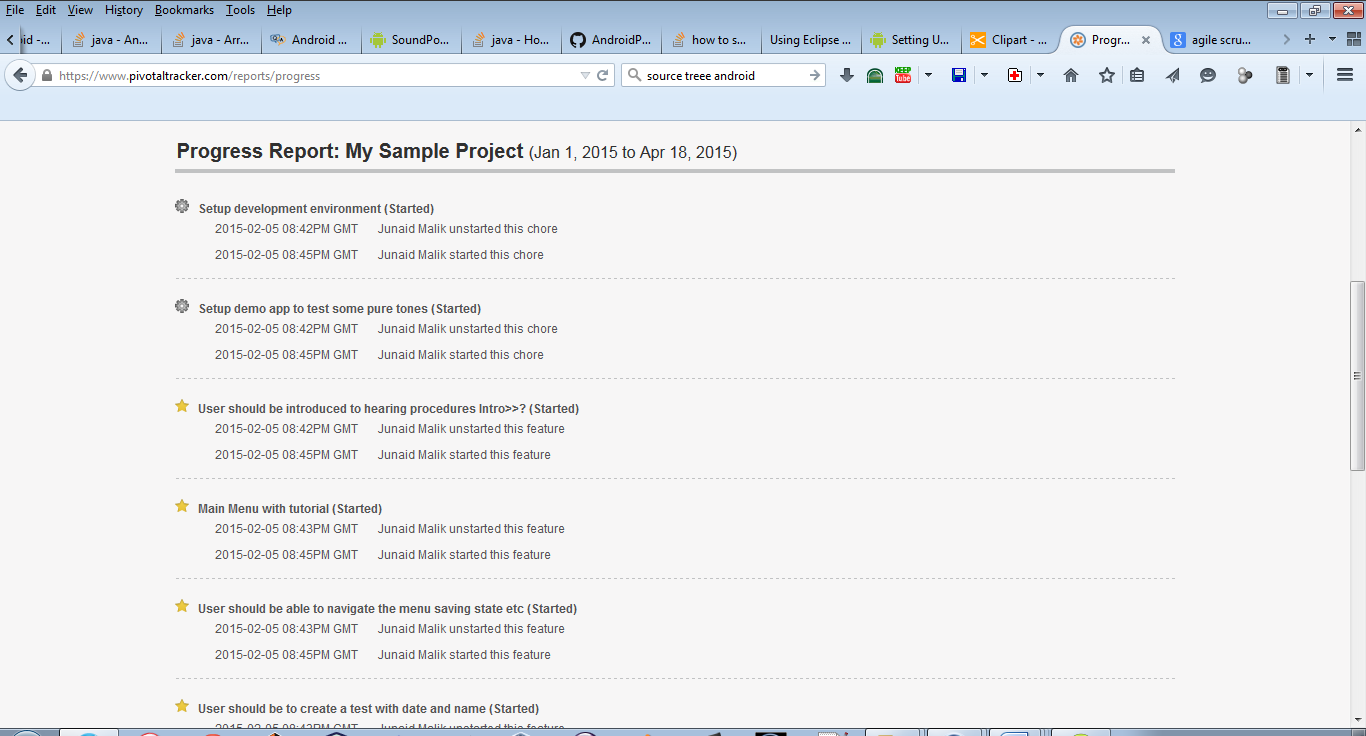


Figure 15 Progress report of project

The above shows some of the main features I added for each sprint. Continually recording results and keeping track of deliverables required dedication and patience towards project management. This further helped me establish a clear and focused view of the requirements.

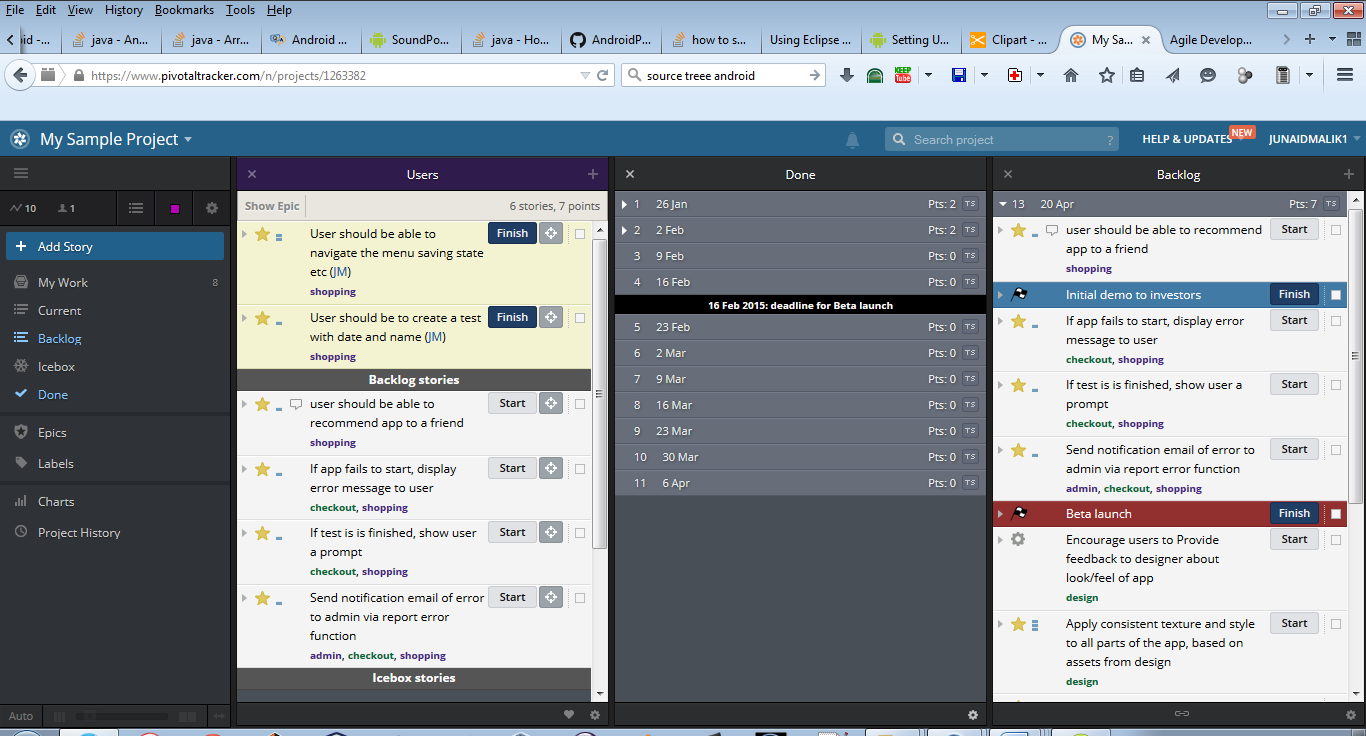


Figure 16 Backlog stories and backlog items

The backlog stories are converted into working software after 1-2 weeks of development they are then added as features at each stage of the product. Iterations enable us to adjust to changes when they are necessary. As the change of requirements in software is inevitable these stories are not concrete as they change depending upon what is working and what does not.

The benefits of this agile iteration approach are powerful because as time is the most valuable asset, for this project I had to carefully manage my time by keeping track of each task.



Figure 17 Gantt chart of project backlog items

## 4.4 Android Best Practises

Some of the best practises that I followed in developing for Android were as follows:

* App installs on the users external disk. Refer to Appendix E.
* Features are easily accessible for different type of users from young to old.
* Layout of information is kept in uniform order and with consistent look and feel to the overall application design.
* Responsive and stable programming to avoid crashes by always catching exceptions.
* A change log of each major and minor change was maintained for each cycle. Refer to Appendix F.
* Maintained the strings resources file, which contains text strings so that we do not have to add the strings into the main body of the code, but instead we reference each string that we want to use from the strings.xml file. This is crucial practise that enables clean code and also provides us with the ability to localize our app into different languages with a much easier technique. Refer to Appendix H.
* High res and low res icons for icons and images.

# Chapter 5 Requirements Analysis

A requirements document specifies the purpose of the system. With the Hearing Test Application the users must be able to test their left and right ear, view results from these tests, save and load tests and view general information about hearing.

## 5.1 Use Case Diagram

The user interface has a screen that displays menu to the user and a simple interface so user can navigate.



Figure 18 Use-Case Diagram of main menu

The use-case diagram helps model the users interactions between the application and the user.

The software encapsulates the functionality and tries to mimic how a hardware device would be. In the design of this application I used a class for frequency generation and a XY plotter to act as the audiogram.

## 5.2 Class Diagrams



Figure 19 Simple class diagram.

The above class demonstrates the three main classes that will be used. Each represents an association with a “has-a” relationship which demonstrates composition. Composition allows us to place sub-objects inside another class. Composition is a form of code reuse so it is useful to include this in the design. This means Hearing Test class has a frequency generator and “has-a” audiogram. This means that the hearing test class contains objects of type frequency generator and audiogram fragment.

Classes used in the system:

* Hearing Test Activity.java
* Frequency generator.java
* Frequency Fragment.java
* AudiogramFragment.java
* MainMenu.java
* MoreInfo.java



Figure 20 Class diagram obtained from analysis



Figure 21 Detailed class diagram of system

Detailed class diagram represents most of major classes that user interacts with.

# Chapter 6 System Design



Figure 22 A professional audiometer used in clinics

The above is a professional audiometer used in clinics it is priced at around €1650. [14]

The classes can be modelled by observing the mechanism of the audiometer. It contains a DSP unit which produces the pure tone so we create a frequency generator class, the audiometer may produce an audiogram with separate software on a pc but we can create a class that will either hold the information for audiogram or render it onto the users screen. The main activity is the Hearing Test Activity, this class contains main behaviour of the overall system, and this is where user selects which ear to test and which frequency to test at.

The diagram below demonstrates the architecture of the android package file which consists of the classes that are in the Dalvik Executable format (DEX). Dalvik is virtual machine in Android operating system. Programs are written in Java and compiled into byte code for JVM which is then translated into Dalvik byte code and stored in .dex or .odex (Optimized Dalvik).



Figure 23 the architecture of the apk.

## 6.1 UX/UI Design

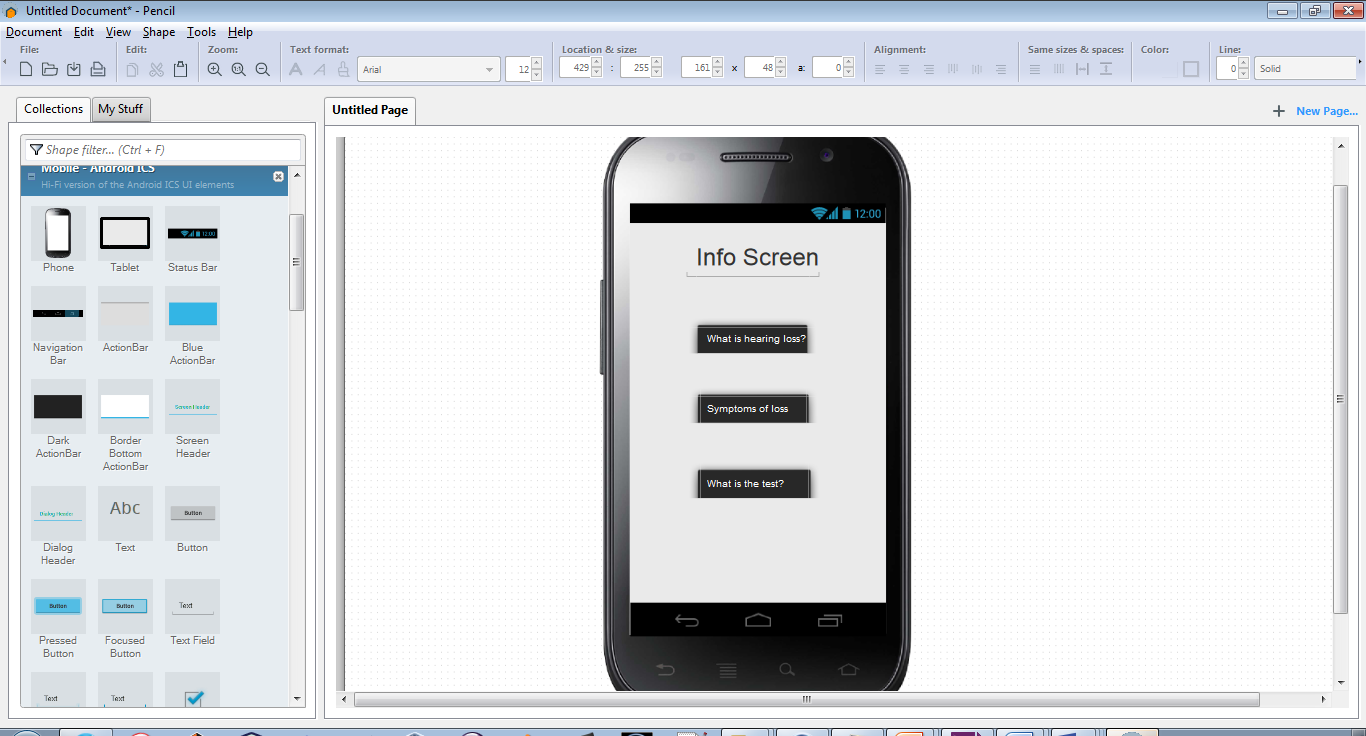


Figure 24 Quick layout design created of UI

Using pencil UI designer we make quick designs, to visualise the look and feel of the application, which later can be adapted into real design in the layout of the XML file. [21]

Quick UI wireframe models can sometimes be quickly created by hand but sometimes it is more useful to create diagrams with tools such as pencil.

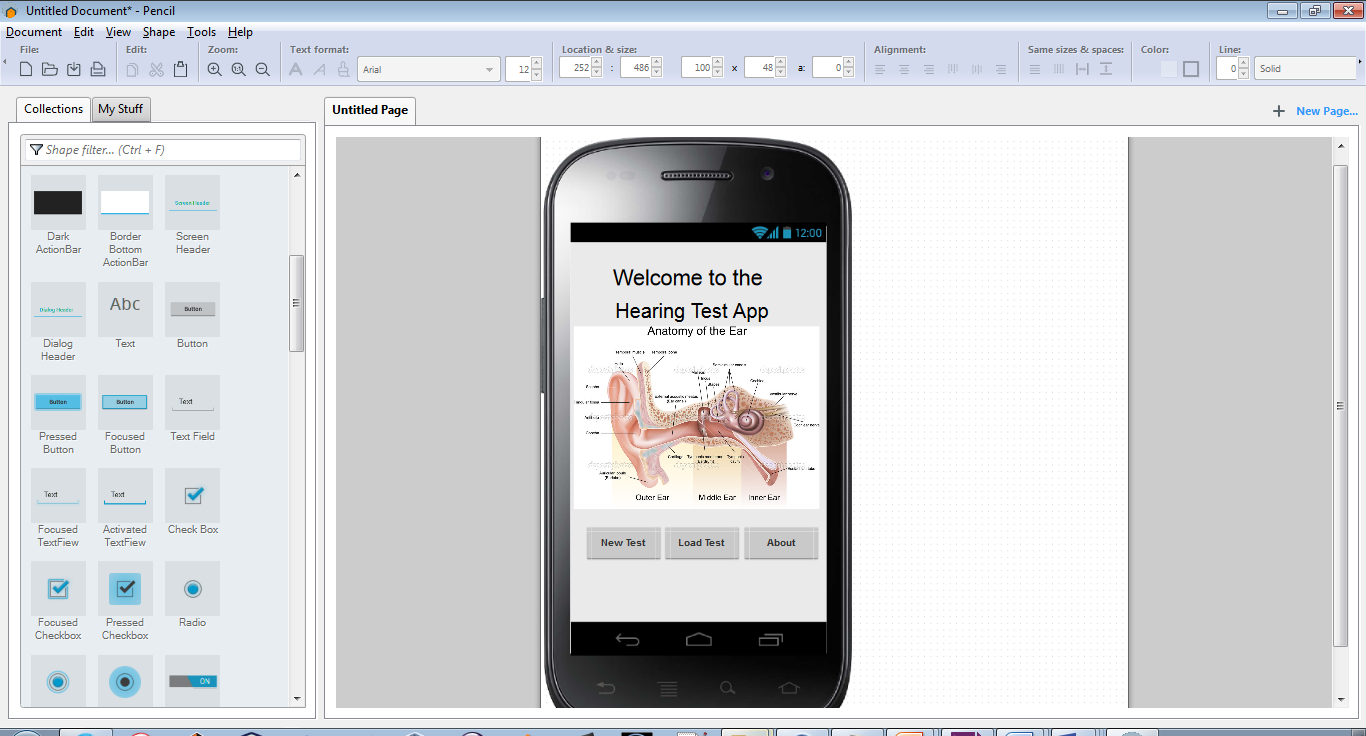


Figure 25 Initial mock design of the main menu

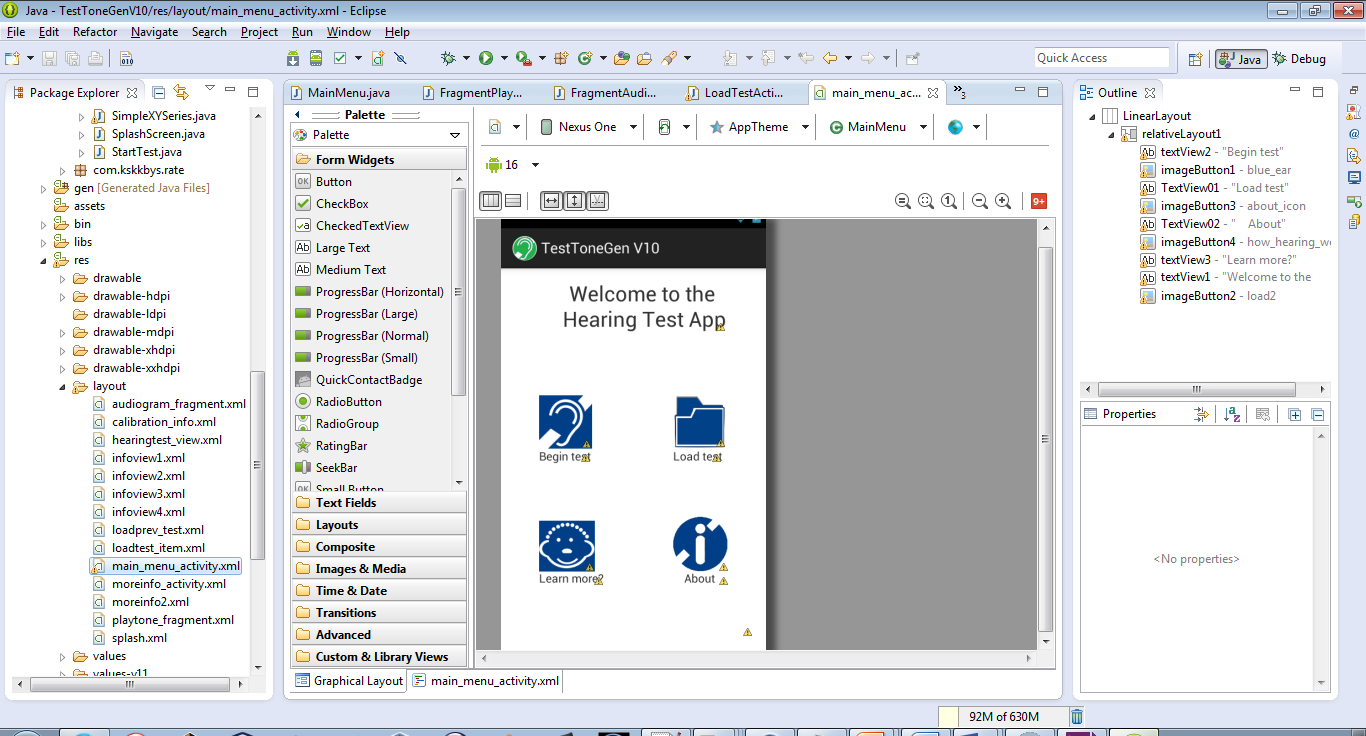


Figure 26 Final design of the main menu

I revised the design into a more elegant and simple view with clickable image buttons. Refer to XML layout in Appendix D

# Chapter 7 System Implementation

## 8.1 Implementing the Activities



Figure 27 Diagram of how the methods are called for each button

Each button has as onClick() method which the is specified in the XML file. Once button is clicked, an intent object is created which takes instance of current class and class to be launched. Intents provide the ability to bind classes together which helps us to launch them along. Refer to Appendix E.

Another important aspect to activities is declaring them in the Android Manifest because if not declared the application will crash with an exception error. In the Manifest we use an activity tag and pass in location of class in the project package. Refer to Appendix G.

## 7.2 Implementing Frequency Generator

Generating the pure tone requires the use of basic sine wave. In the Frequency Generator class we create an array of generated sound samples and pass it into the AudioTrack.write function. This function takes data to the audio sink for playback in streaming mode. The byte class I sued as it wraps primitive value byte in an object. The encoding scheme is PCM 16 bits per sample. Pulse code modulation encodes an audio waveform in the time domain. The frequency and sample rate measures how many samples are plated each second. We use getChannelConfiguration which return the configured channel configuration. [23]

The encoding plays an important role as the byte array is in played in form of audio. The audio track write method takes in three parameters one is the byte array this is the audio data, second is the offset bit and last is the size of the bytes.

We create an object of type short which is a 16-bit signed two’s complement integer. The short class wraps a value of primitive type in an object. Refer to Appendix A.

Configuring the sample rate 44,100 gives the highest quality sound. We test each pure tone and compare if it sounds is as good [24].



## 7.3 Implementing Audiogram with Androidplot library

In order to render the audiogram we require a comprehensive library to create our audiogram. There are two most popular open source libraries available, both were tried and tested, these included AndroidPlot and GraphView. [24]

After testing both GraphView and AndroidPlot, the latter was more useful as it contained helpful examples and its source code was more understandable. To implement AndroidPlot within the project first we have to include it within the project build path. Refer to Appendix D: Audiogram\_fragment.xml

The main audiogram is implemented into the HearingTestActivity class. This class contains a method to render XYPlot which we associate with out XYPlot which we declared in our audiogram XML file. We create a reference object of type XYPlot this contains information for the plot XY axis and colour of background. This XYPlot class further inherits from XYSeries, XYSeriesFormatter and XYSeriesRenderer. These are all interfaces or abstract classes which provide impartial or no implementation.

Figure 28 Hierarchy of XYPlot

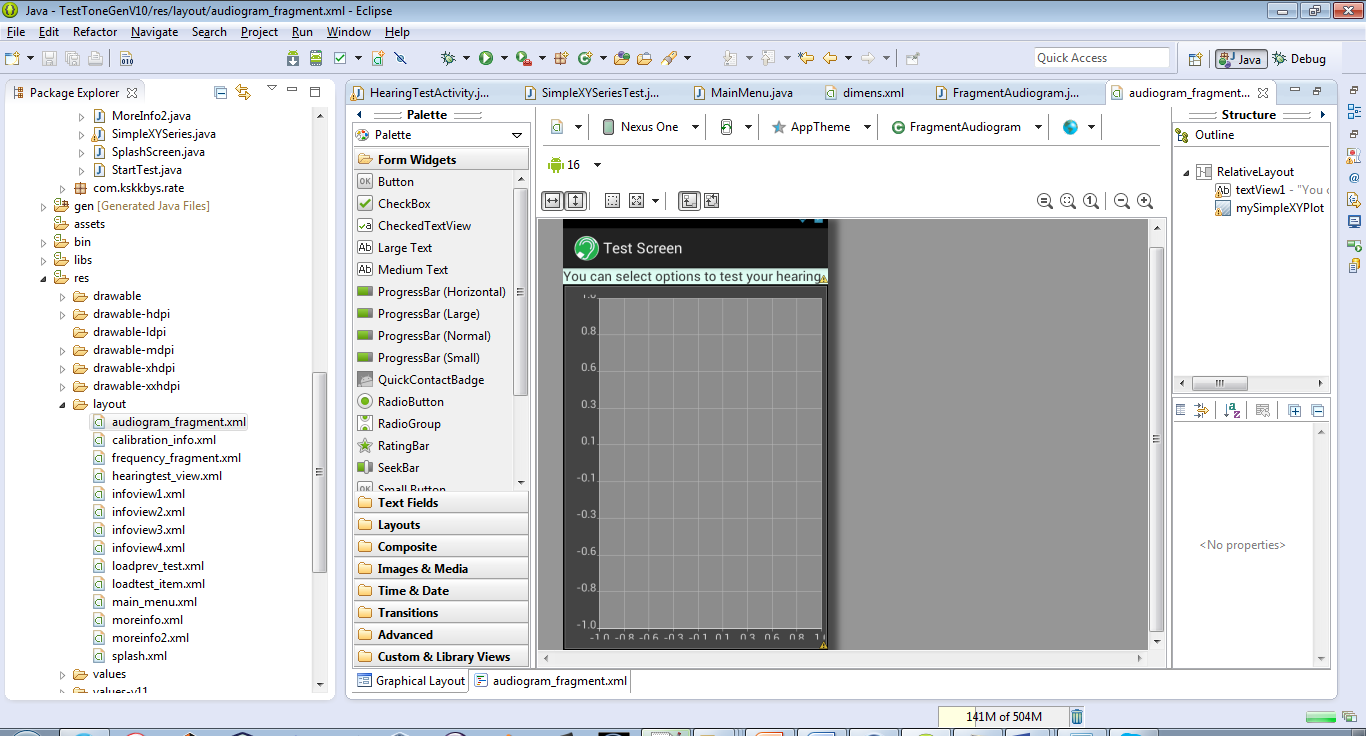




Figure 29 XYPlot template created from XML

## 7.4 Implementing Fragments

Fragments in android can be described as an activity within an activity. I chose to implement fragments as I needed a way to combine multiple activities that required different classes acting on screen view. In my design two fragments were added to one layout view that is the HearingTestActivity. In order to implement this I first created a layout view for the playtone\_fragment this is where the user controls the testing options. The other fragment is then of the audiogram which only draws a XYPlot using the Android plot library. The two fragments are then combined into one layout, the reason for doing this is that we can separate the logic of both classes to their own respected layouts and that once we embed them as fragments, the code is less coupled and not totally dependent on other methods .

The implementation of the fragments was one of the trickiest aspects of the project as it took me a long time to implement. There many of ways to implement fragments a there is no best solution so it was trial and error to discover which method worked best. The method that worked for me was where I had fragments tags declared in the hearing test layout and have a class tag defined and also a tools tag for each fragment. The tools tag defines which layout is rendered and is very important. Refer to code in Appendix D.



Figure 30 Diagram of fragment associations.

To complete the implementation of fragments we create two classes for each xml file. These two classes contain a onCreateView method which inflates the layout into the hearingtest\_view.xml activity. The inflater instantiates a layout XML file into the View object. The inflation uses XML file at build time. Last thing to make sure is to include the android-support-v4.jar file in the project build path. Refer to Appendix D.

## 7.5 Implementing Rate this app plug in

I implemented a simple rate this app plugin written by Keisuke Kobayashi. It is an open source library and was available through git-hub. I decided to implement this feature to add to the applications user experience. As this plugin provides the user with a simple prompt alert box which provides a layer of interactivity to rate the application on the Google Play Store.

Implementing the library is very simple, RTA plug in is very useful in that provides an intuitive and simple way to receive user feedback. It encourages user interactivity by prompting user to give feedback and rate the app depending on their experience.

# Chapter 8 Testing

To create a test project in Eclipse with ADT:

1. In Eclipse, select File > New > Other. This opens the Select a Wizard dialog.

2. In the dialog, in the Wizards drop-down list, find the entry for Android, then click the toggle to the left. Select Android Test Project, then at the bottom of the dialog click Next. The New Android Test Project wizard appears.

3. Next to Test Project Name, enter a name for the project. You may use any name, but you may want to associate the name with the project name for the application under test. One way to do this is to take the application's project name, append the string "Test" to it, and then use this as the test package project name.

The name becomes part of the suggested project path, but you can change this in the next step.

4. In the Content panel, examine the suggested path to the project. If Use default location is set, then the wizard will suggest a path that is a concatenation of the workspace path and the project name you entered. For example, if your workspace path is /usr/local/workspace and your project name is MyTestApp, then the wizard will suggest /usr/local/workspace/MyTestApp. To enter your own choice for a path, unselect Use default location, then enter or browse to the path where you want your project.

To learn more about choosing the location of test projects, please read Testing Fundamentals.

5. In the Test Target panel, set An Existing Android Project, click Browse, then select your Android application from the list. You now see that the wizard has completed the Test Target Package, Application Name, and Package Name fields for you (the latter two are in the Properties panel).

6. In the Build Target panel, select the Android SDK platform that the application under test uses.

7. Click Finish to complete the wizard. If Finish is disabled, look for error messages at the top of the wizard dialog, and then fix any problems.

The testing framework has these key features:

• Android test suites are based on JUnit. You can use plain JUnit to test a class that doesn't call the Android API, or Android's JUnit extensions to test Android components. If you're new to Android testing, you can start with general-purpose test case classes such as AndroidTestCase and then go on to use more sophisticated classes.

• The Android JUnit extensions provide component-specific test case classes. These classes provide helper methods for creating mock objects and methods that help you control the lifecycle of a component.

• Test suites are contained in test packages that are similar to main application packages, so you don't need to learn a new set of tools or techniques for designing and building tests.

• The SDK tools for building and tests are available in Eclipse with ADT, and also in command-line form for use with other IDEs. These tools get information from the project of the application under test and use this information to automatically create the build files, manifest file, and directory structure for the test package.

• The SDK also provides monkeyrunner, an API for testing devices with Python programs, and UI/Application Exerciser Monkey, a command-line tool for stress-testing UIs by sending pseudo-random events to a device.

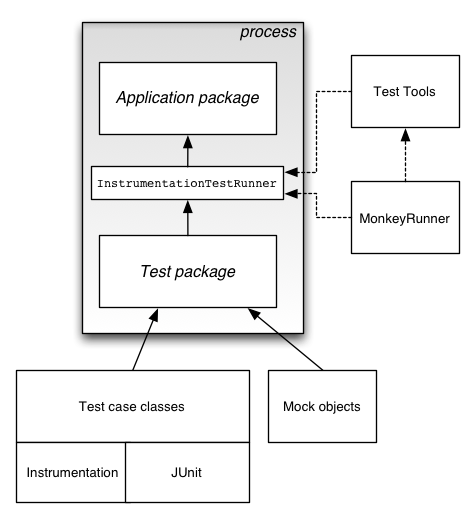


Figure 30 Architecture of Testing in Android

## 8. 1 Test Structure

Android's build and test tools assume that test projects are organized into a standard structure of tests, test case classes, test packages, and test projects.

Android testing is based on JUnit. In general, a JUnit test is a method whose statements test a part of the application under test. You organize test methods into classes called test cases (or test suites). Each test is an isolated test of an individual module in the application under test. Each class is a container for related test methods, although it often provides helper methods as well.

In JUnit, you build one or more test source files into a class file. Similarly, in Android you use the SDK's build tools to build one or more test source files into class files in an Android test package. In JUnit, you use a test runner to execute test classes. In Android, you use test tools to load the test package and the application under test, and the tools then execute an Android-specific test runner.

| **Test area** | **Description** |
| --- | --- |
| Activity life cycle events | You should test if you activity handles the Android life cycle events correctly. You should also test if the configuration change events are handled well and if instance state of your user interface components is restored. |
| File system and database operations | Write and read access from and to the file system should be tested including the handling of databases. |
| Different device configurations | You should also test if your application behaves well on different device configurations. |

## 8.2 Testing preconditions

It is good practice in Android testing to have one method called testPreconditions() which tests the pre-conditions for all other tests. If this method fails, you know immediately that the assumptions for the other tests have been violated.

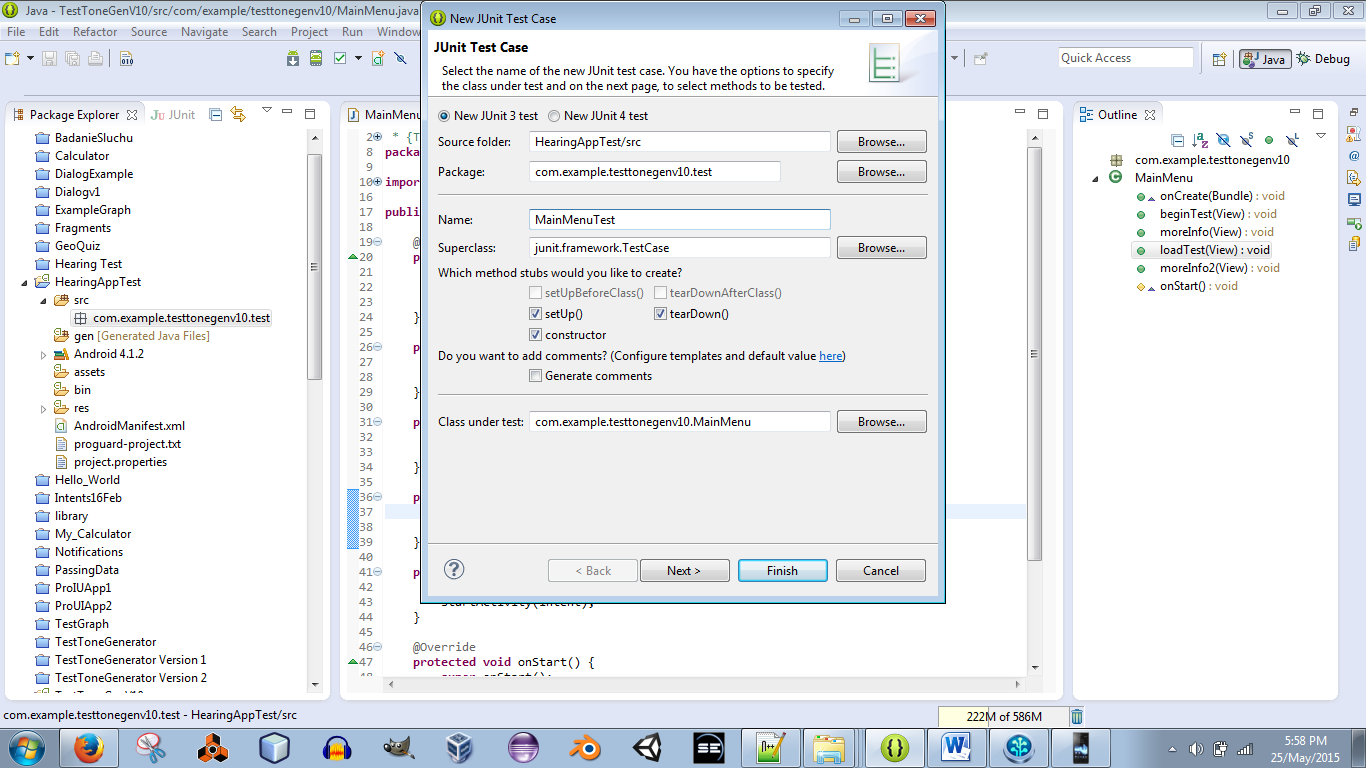


Figure 31 Creating the JUnit test case

# Chapter 9 Conclusion

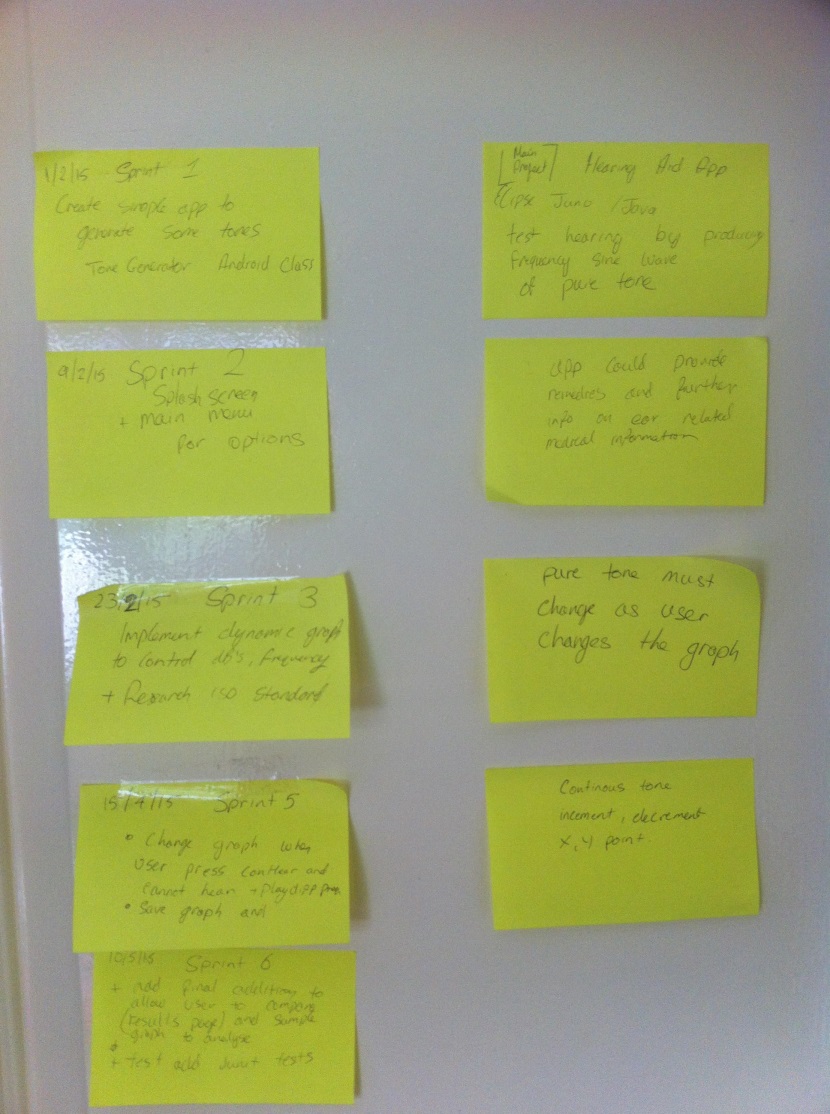
In conclusion I learned how to create an Android application with good design. I learned how to use version control system and effectively use it within a project and with Git-Hub.

# Appendices

## Appendix A: Code for FrequencyGen.java

|  |
| --- |
| /\*  \* {This class is used to run methods based on what user chooses  \* for the Frequency Generator }  \*  \* @version Build {1.0} (14 May 2015)  \* @author Junaid Malik  \*/  package com**.**example**.**testtonegenv10**;**  **import** android**.**app**.**Activity**;**  **import** android**.**media**.**AudioFormat**;**  **import** android**.**media**.**AudioManager**;**  **import** android**.**media**.**AudioTrack**;**  **import** android**.**media**.**MediaPlayer**;**  **import** android**.**widget**.**Toast**;**  // this class is based on Marble Mice: Generate And Play A Tone In Android  // www.marblemice.blogspot.ie/2010/04/generate-and-play-tone-in-android.html  public class FrequencyGenerator **extends** Activity **{**  public final int duration **=** 5**;** // length of tone in seconds  public int sampleRate **=** 44100**;** // sample rate at 44,100 times per second  public final int numSamples **=** duration **\*** sampleRate**;**// in order to avoid  // distortion  // number of samples =  // duration multiplied  // by sample rate  public final double sample**[]** **=** **new** double**[**numSamples**];**  public double freqOfTone**;** // frequency in Hz  public final byte generatedSound**[]** **=** **new** byte**[**2 **\*** numSamples**];** // array of  // genSoun =  // 2 times  // the  // number of  // samples  public static final int LEFT\_EAR **=** 1**;** // immutable int for left ear  public static final int RIGHT\_EAR **=** 2**;** // immutable int for light ear  public AudioTrack audioTrack **=** **null;** // null object of type audioTrack  public void genTone**()** **{**  // fill out the array with samples  **for** **(**int counter **=** 0**;** counter **<** numSamples**;** **++**counter**)** **{**  sample**[**counter**]** **=** Math**.**sin**(**2 **\*** Math**.**PI **\*** counter // x(t)=A Sin 2pi n  // fa/fs  **/** **(**sampleRate **/** freqOfTone**));**  **}**  // convert to 16 bit pcm sound array  // assumes the sample buffer is normalised.  int index **=** 0**;**  // enhanced for loop going through sampa array  **for** **(**final double doubleValue **:** sample**)** **{**  // scale to maximum amplitude of 32,767  final short value **=** **(**short**)** **((**doubleValue **\*** 32767**));**  // in 16 bit wav PCM, first byte is the low order byte  // 0xff00 is hex literal for the value 65,280 bitwise  // it resets the rightmost 8 bits  // operation, right shift to 8bits  // 0x00ff only the rightmost 8 bits are kept  // value is anded with 0x00FF  // Anding with 8 zeros we are masking off 8 right MSB  generatedSound**[**index**++]** **=** **(**byte**)** **(**value **&** 0x00ff**);**  generatedSound**[**index**++]** **=** **(**byte**)** **((**value **&** 0xff00**)** **>>>** 8**);**  **}**  **}**  /\*  \* audiotrack plays pcm audio buffers data is pushed to audiotrack object in  \* streaming mode the system writes continous data using write() method this  \* method takes three params audioData, offfsetinBYtes and sizeinBytes  \*/  public void playSound**()** **{**  audioTrack **=** **new** AudioTrack**(**AudioManager**.**STREAM\_MUSIC**,** sampleRate**,**  AudioFormat**.**CHANNEL\_OUT\_STEREO**,** AudioFormat**.**ENCODING\_PCM\_16BIT**,**  numSamples**,** AudioTrack**.**MODE\_STATIC**);**  audioTrack**.**write**(**generatedSound**,** 0**,** generatedSound**.**length**);**  audioTrack**.**getChannelConfiguration**();**  **try** **{**  audioTrack**.**play**();**  **}** **catch** **(**Exception e**)** **{** // error message if not playable  Toast**.**makeText**(**getApplicationContext**(),** "Error playing audio"**,**  Toast**.**LENGTH\_SHORT**).**show**();**  **}**  **}**  public void playTone**()** **{**  playTone**(**1.0f**,** 1.0f**);** // this method plays tone with full  // volume in both ear  **}**  public void playTone**(**float left**,** float right**)** **{** // overloaded playTone  // method with volume  // determined by user  Float leftVolume **=** left**;**  Float rightVolume **=** right**;**  **try** **{**  audioTrack**.**write**(**generatedSound**,** 0**,** generatedSound**.**length**);**// pass  // genSound  // into  // write  // method  audioTrack**.**setStereoVolume**(**leftVolume**,** rightVolume**);** // left and  // right  // are set  // by  // user  // choice  audioTrack**.**play**();**  **}** **catch** **(**Exception e**)** **{** // error message if not playable  Toast**.**makeText**(**getApplicationContext**(),** "Error playing audio"**,**  Toast**.**LENGTH\_SHORT**).**show**();**  **}**  **}**  public void playTone**(**int earSelect**)** **{**  **switch** **(**earSelect**)** **{**  **case** LEFT\_EAR**:**  playTone**(**1.0f**,** 0.0f**);**  **break;**  **case** RIGHT\_EAR**:**  playTone**(**0.0f**,** 0.05f**);**  **break;**  **}**  **}**  public void start**()** **{**  audioTrack**.**play**();**  **}**  public void stop**()** **{**  audioTrack**.**stop**();**  **}**  **}** |

## Appendix B: User Story Mapping



## Appendix C: XML code for main\_menu.xml

<LinearLayout 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"**

android:background=**"#FFFFFF"**

android:paddingBottom=**"@dimen/activity\_vertical\_margin"**

android:paddingLeft=**"@dimen/activity\_horizontal\_margin"**

android:paddingRight=**"@dimen/activity\_horizontal\_margin"**

android:paddingTop=**"@dimen/activity\_vertical\_margin"**

tools:context=**"com.example.testtonegenv10.MainMenu"** >

<RelativeLayout

android:id=**"@+id/relativeLayout1"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"** >

<TextView

android:id=**"@+id/textView2"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentLeft=**"true"**

android:layout\_below=**"@+id/imageButton1"**

android:layout\_marginLeft=**"30dp"**

android:text=**"Begin test"** />

<ImageButton

android:id=**"@+id/imageButton1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignLeft=**"@+id/textView2"**

android:layout\_alignTop=**"@+id/imageButton2"**

android:background=**"#FFFFFF"**

android:onClick=**"beginTest"**

android:src=**"@drawable/blue\_ear"** />

<TextView

android:id=**"@+id/TextView01"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignBaseline=**"@+id/textView2"**

android:layout\_alignBottom=**"@+id/textView2"**

android:layout\_alignLeft=**"@+id/imageButton2"**

android:text=**"Load test"** />

<ImageButton

android:id=**"@+id/imageButton3"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_above=**"@+id/TextView02"**

android:layout\_alignLeft=**"@+id/TextView01"**

android:background=**"#FFFFFF"**

android:onClick=**"moreInfo"**

android:src=**"@drawable/about\_icon"** />

<TextView

android:id=**"@+id/TextView02"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignLeft=**"@+id/imageButton3"**

android:layout\_alignParentBottom=**"true"**

android:layout\_alignRight=**"@+id/imageButton3"**

android:layout\_marginBottom=**"61dp"**

android:text=**" About"** />

<ImageButton

android:id=**"@+id/imageButton4"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_above=**"@+id/TextView02"**

android:layout\_alignLeft=**"@+id/textView2"**

android:background=**"#FFFFFF"**

android:onClick=**"moreInfo2"**

android:src=**"@drawable/how\_hearing\_works"** />

<TextView

android:id=**"@+id/textView3"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignBaseline=**"@+id/TextView02"**

android:layout\_alignBottom=**"@+id/TextView02"**

android:layout\_alignLeft=**"@+id/imageButton4"**

android:text=**"Learn more?"** />

<TextView

android:id=**"@+id/textView1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentTop=**"true"**

android:layout\_centerHorizontal=**"true"**

android:gravity=**"top|center"**

android:text=**"@string/hello\_world"**

android:textSize=**"25dp"** />

<ImageButton

android:id=**"@+id/imageButton2"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignRight=**"@+id/textView1"**

android:layout\_below=**"@+id/textView1"**

android:layout\_marginTop=**"76dp"**

android:background=**"#FFFFFF"**

android:onClick=**"loadTest"**

android:src=**"@drawable/load2"** />

</RelativeLayout>

</LinearLayout>

## Appendix D: Fragments in hearingtest\_view.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<RelativeLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:layout\_width=**"fill\_parent"**

android:layout\_height=**"fill\_parent"**

android:orientation=**"horizontal"** >

<fragment

android:id=**"@+id/audiogram\_fragment"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

android:layout\_alignParentBottom=**"true"**

android:layout\_marginBottom=**"140dp"**

class=**"com.example.testtonegenv10.FragmentAudiogram"**

tools:layout=**"@layout/audiogram\_fragment"** />

<fragment

android:id=**"@+id/playtone\_fragment"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"149dp"**

android:layout\_alignParentBottom=**"true"**

android:layout\_alignParentLeft=**"true"**

class=**"com.example.testtonegenv10.FragmentPlayTone"**

tools:layout=**"@layout/playtone\_fragment"** />

</RelativeLayout>

### Frequency\_fragment.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<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"**

android:orientation=**"vertical"**

tools:context=**"com.example.testtonegenv10.FragmentPlayTone"** >

<Button

android:id=**"@+id/cannotHearButton"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentBottom=**"true"**

android:layout\_alignParentRight=**"true"**

android:text=**"Cant hear"** />

<Button

android:id=**"@+id/canHearButton"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentBottom=**"true"**

android:text=**"Can hear"** />

<Button

android:id=**"@+id/finishButton"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentBottom=**"true"**

android:layout\_toLeftOf=**"@+id/cannotHearButton"**

android:text=**"Barely hear"** />

<RadioGroup

android:id=**"@+id/radioGroup1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_above=**"@+id/canHearButton"**

android:layout\_alignParentLeft=**"true"**

android:orientation=**"horizontal"** >

<RadioButton

android:id=**"@+id/leftEarButton"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:checked=**"true"**

android:text=**"Left"** />

<RadioButton

android:id=**"@+id/rightEarButton"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:text=**"Right"** />

</RadioGroup>

<Button

android:id=**"@+id/playFrequencyButton"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignBottom=**"@+id/radioGroup1"**

android:layout\_alignParentRight=**"true"**

android:text=**"Play Tone"** />

<Button

android:id=**"@+id/nextFrequencyButton"**

style=**"?android:attr/buttonStyleSmall"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_above=**"@+id/playFrequencyButton"**

android:layout\_alignParentRight=**"true"**

android:text=**"Next Frequency"** />

</RelativeLayout>

### Audiogram\_fragment.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<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"**

android:orientation=**"vertical"**

tools:context=**"com.example.testtonegenv10.FragmentAudiogram"** >

<TextView

android:id=**"@+id/textView1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentLeft=**"true"**

android:layout\_alignParentRight=**"true"**

android:layout\_alignParentTop=**"true"**

android:background=**"#E0FFF5"**

android:text=**"You can select options to test your hearing"**

android:textSize=**"16sp"** />

<com.androidplot.xy.XYPlot

android:id=**"@+id/mySimpleXYPlot"**

android:layout\_width=**"313dp"**

android:layout\_height=**"210dp"**

android:layout\_alignParentBottom=**"true"**

android:layout\_alignParentLeft=**"true"**

android:layout\_alignParentRight=**"true"**

android:layout\_below=**"@+id/textView1"**

android:layout\_weight=**"3.49"** />

</RelativeLayout>

### Java code for FragmentFrequency.java

package com.example.testtonegenv10;

import android.os.Bundle;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

public class FragmentFrequency extends Fragment {

public FragmentFrequency() {

} // empty constructor

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// ---Inflate the layout for this fragment---

return inflater.inflate(R.layout.frequency\_fragment, container, false);

}

}

### Java code for FragmentAudiogram.java

package com.example.testtonegenv10;

import android.os.Bundle;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

public class FragmentAudiogram extends Fragment {

public FragmentAudiogram() {

}

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

return inflater.inflate(R.layout.audiogram\_fragment, container, false);

}

}

## Appendix E: Java code

### MainMenu.java

/\*

\* {This class is used to run methods based on what user chooses

\* for the main menu }

\*

\* @version Build {0.6} (26 March 2015)

\* @author Junaid Malik

\*/

package com**.**example**.**testtonegenv10**;**

**import** com**.**kskkbys**.**rate**.**RateThisApp**;**

**import** android**.**app**.**Activity**;**

**import** android**.**content**.**Intent**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**view**.**View**;**

public class MainMenu **extends** Activity **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{** // activity is strarted

// here

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**main\_menu**);** // main menu view is loaded first

**}**

public void beginTest**(**View v**)** **{**

Intent intent **=** **new** Intent**(this,** StartTest**.**class**);** // intent used to launch activity

startActivity**(**intent**);**

**}**

public void moreInfo**(**View v**)** **{**

Intent intent **=** **new** Intent**(**MainMenu**.this,** MoreInfo**.**class**);**

startActivity**(**intent**);**

**}**

public void loadTest**(**View v**)** **{**

Intent intent **=** **new** Intent**(**MainMenu**.this,** LoadTestActivity**.**class**);**

startActivity**(**intent**);**

**}**

public void moreInfo2**(**View v**)** **{**

Intent intent **=** **new** Intent**(**MainMenu**.this,** MoreInfo2**.**class**);**

startActivity**(**intent**);**

**}**

// code from rate this app by Keisuke Kobayashi

@Override

protected void onStart**()** **{**

**super.**onStart**();**

// Monitor launch times and interval from installation

RateThisApp**.**onStart**(this);**

// If the criteria is satisfied, "Rate this app" dialog will be shown

RateThisApp**.**showRateDialogIfNeeded**(this);**

**}**

**}**

### HearingTestActivity.java

package com**.**example**.**testtonegenv10**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Iterator**;**

**import** android**.**app**.**AlertDialog**;**

**import** android**.**content**.**Context**;**

**import** android**.**content**.**DialogInterface**;**

**import** android**.**graphics**.**Bitmap**;**

**import** android**.**graphics**.**BitmapFactory**;**

**import** android**.**graphics**.**BitmapShader**;**

**import** android**.**graphics**.**Color**;**

**import** android**.**graphics**.**Matrix**;**

**import** android**.**graphics**.**RectF**;**

**import** android**.**graphics**.**Shader**;**

**import** android**.**media**.**AudioManager**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**util**.**Log**;**

**import** android**.**view**.**KeyEvent**;**

**import** android**.**view**.**View**;**

**import** android**.**view**.**View**.**OnClickListener**;**

**import** android**.**widget**.**Button**;**

**import** android**.**widget**.**RadioButton**;**

**import** android**.**widget**.**Toast**;**

**import** com**.**androidplot**.**xy**.**BoundaryMode**;**

**import** com**.**androidplot**.**xy**.**LineAndPointFormatter**;**

**import** com**.**androidplot**.**xy**.**PointLabelFormatter**;**

**import** com**.**androidplot**.**xy**.**XYPlot**;**

**import** com**.**androidplot**.**xy**.**XYSeries**;**

**import** com**.**androidplot**.**xy**.**XYStepMode**;**

public class HearingTestActivity **extends**

android**.**support**.**v4**.**app**.**FragmentActivity **implements** OnClickListener **{**

public HearingTestActivity**()** **{** // empty default constructor

**}**

private Button canHearButton**,** cannotHearButton**,** nextFrequencyButton**,**

finishButton**,** playFrequencyButton**;**

private RadioButton rightEarButton**,** leftEarButton**;**

static int defaultdB **=** 40**;** // default dB level (yVal point)

private FrequencyGenerator frequencygen **=** **new** FrequencyGenerator**();** // create FreqGen

// reference

private FragmentAudiogram audiogram **=** **new** FragmentAudiogram**();** // create

// AudioGram

// reference

// private AudioManager audioManager; // reference to AudioManager class

private XYPlot mySimpleXYPlot**;** // reference to XYPlot class

private SimpleXYSeries series1**,** series2**;**

//private int[] frequencies;

private int currentFreq **=** 0**;**

private int indexYval **=** 0**;** // initial index position for yVals

private int indexXvals **=** 0**;**

private int FREQ\_LEN **=** 120**;** // length of array for x vals

private int DB\_LEN **=** 120**;** // length of array for y vals

// private int[] frequencies = new frequencies;

Integer**[]** frequencies **=** **{**250**,** 500**,** 1000**,** 2000**,** 4000**,** 6000**,** 8000**};**

Integer**[]** xVals **=** **new** Integer**[**FREQ\_LEN**];** // array of type Integer with

// length defined

Integer**[]** yVals **=** **new** Integer**[**DB\_LEN**];**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**hearingtest\_view**);** // setting the view with

// defined XML file

startUp**();** // opens up info box

**this.**setVolumeControlStream**(**AudioManager**.**STREAM\_MUSIC**);** // reference to

// current

// volume stream

//frequencies = getResources().getIntArray(R.array.frequencies);

// frequencies = getResources().getIntArray(R.array.frequencies);

leftEarButton **=** **(**RadioButton**)** findViewById**(**R**.**id**.**leftEarButton**);**

leftEarButton**.**setOnClickListener**(this);**// if button clicked

// onClickListener runs

// corresponding onClick()

// method

rightEarButton **=** **(**RadioButton**)** findViewById**(**R**.**id**.**rightEarButton**);**

rightEarButton**.**setOnClickListener**(this);**// if button clicked

// onClickListener runs

// corresponding onClick()

// method

playFrequencyButton **=** **(**Button**)** findViewById**(**R**.**id**.**playFrequencyButton**);**

playFrequencyButton**.**setOnClickListener**(this);**// if button clicked

// onClickListener

// runs corresponding onClick()

// method

nextFrequencyButton **=** **(**Button**)** findViewById**(**R**.**id**.**nextFrequencyButton**);**

nextFrequencyButton**.**setOnClickListener**(this);**// if button clicked

// onClickListener

// runs corresponding onClick()

// method

canHearButton **=** **(**Button**)** findViewById**(**R**.**id**.**canHearButton**);**

canHearButton**.**setOnClickListener**(this);**// if button clicked

// onClickListener runs

// corresponding onClick()

// method

cannotHearButton **=** **(**Button**)** findViewById**(**R**.**id**.**cannotHearButton**);**

cannotHearButton**.**setOnClickListener**(this);**// if button clicked

// onClickListener runs

// corresponding onClick()

// method

// initialize our XYPlot reference:

mySimpleXYPlot **=** **(**XYPlot**)** findViewById**(**R**.**id**.**mySimpleXYPlot**);**

graphSettings**();** // retrieve layout for audiogram

**}**

// onClick method gets called each time a button is pressed

@Override

public void onClick**(**View v**)** **{**

// audio manager provides access to volume control

// Context is an interface to global info about app environment

// context implementation provided by OS.

AudioManager audioManager **=** **(**AudioManager**)** getSystemService**(**Context**.**AUDIO\_SERVICE**);**

// Run the function findViewById and pass it R.id.button1 as parameters

// Find out which button was pushed based on its ID

// Switch statement checks for which button was checked and changes that

**switch** **(**v**.**getId**())** **{**

**case** R**.**id**.**nextFrequencyButton**:** **{**

setFreqIndex**();**

//setFreqValue();

indexYval **+=** 1**;**

currentFreq **+=** 1**;**

yVals**[** indexYval **]** **=** 40**;** // set next index to default value 40

updatePlot**();**

**}**

**case** R**.**id**.**playFrequencyButton**:** **{**

frequencygen**.**freqOfTone **=** frequencies**[**currentFreq**];**

frequencygen**.**genTone**();**

frequencygen**.**playSound**();**

**}**

**case** R**.**id**.**leftEarButton**:** **{**

**try** **{**

**if** **(**leftEarButton**.**isChecked**())**

frequencygen**.**playTone**(**FrequencyGenerator**.**LEFT\_EAR**);**

**else**

frequencygen**.**playTone**(**0.0f**,** 1.0f**);**

**}** **catch** **(**Exception e**)** **{**

Toast**.**makeText**(**getApplicationContext**(),** "Error"**,**

Toast**.**LENGTH\_SHORT**).**show**();**

**}**

**break;**

**}**

**case** R**.**id**.**rightEarButton**:** **{**

**try** **{**

**if** **(**rightEarButton**.**isChecked**())**

frequencygen**.**playTone**(**FrequencyGenerator**.**RIGHT\_EAR**);**

**}** **catch** **(**Exception e**)** **{**

Toast**.**makeText**(**getApplicationContext**(),** "Error"**,**

Toast**.**LENGTH\_SHORT**).**show**();**

**}**

**break;**

**}**

**case** R**.**id**.**canHearButton**:** **{**

**try** **{**

yVals**[**indexYval**]** **-=** 1**;**

updatePlot**();**

audioManager**.**setStreamVolume**(**AudioManager**.**STREAM\_MUSIC**,**

AudioManager**.**ADJUST\_LOWER**,** AudioManager**.**FLAG\_VIBRATE**);**

**}** **catch** **(**Exception e**)** **{**

Toast**.**makeText**(**getApplicationContext**(),** "Error"**,**

Toast**.**LENGTH\_SHORT**).**show**();**

**}**

**break;**

**}**

**case** R**.**id**.**cannotHearButton**:** **{**

**try** **{**

yVals**[**indexYval**]** **+=** 1**;**

updatePlot**();**

audioManager**.**adjustStreamVolume**(**AudioManager**.**STREAM\_MUSIC**,**

AudioManager**.**ADJUST\_RAISE**,** AudioManager**.**FLAG\_VIBRATE**);**

**}** **catch** **(**Exception e**)** **{**

Toast**.**makeText**(**getApplicationContext**(),** "Error"**,**

Toast**.**LENGTH\_SHORT**).**show**();**

**}**

**break;**

**}**

**}** // end of switch case

**}** // end of onClick methodrightEarButton

private void setFreqIndex**()** **{**

xVals**[**0**]** **=** 250**;**

xVals**[**1**]** **=** 500**;**

xVals**[**2**]** **=** 1000**;**

xVals**[**3**]** **=** 2000**;**

xVals**[**4**]** **=** 3000**;**

xVals**[**5**]** **=** 4000**;**

xVals**[**6**]** **=** 6000**;**

xVals**[**7**]** **=** 8000**;**

// if(xVals[indexXvals] == xVals[7] ) {

// xVals[0] = 40;

// }

**}**

private void setDecibels**()** **{**

//yVals[ indexYval + 1 ] = 40;

// for (int i = 1; i < DB\_LEN; i += 1) {

//yVals[i] = ((int) + yVals[i]);

// }

// int i = 0;

// yVals[i++] = 40;

updatePlot**();**

**}**

private void setFreqValue**()** **{**

frequencygen**.**freqOfTone **=** frequencies**[**currentFreq**];**

**if** **(**currentFreq **>** 6**)** **{**

frequencies**[**0**]** **=** currentFreq**;**

**}**

currentFreq **+=** 1**;**

**}**

// nested class of SimpleXYSeries

public SimpleXYSeries getSeries**()** **{**

**if** **(**leftEarButton**.**isChecked**())** **{**

**return** **new** SimpleXYSeries**(**Arrays**.**asList**(**xVals**),**

Arrays**.**asList**(**yVals**),** "Left Ear"**);**

**}** **else** **{**

**return** **new** SimpleXYSeries**(**Arrays**.**asList**(**xVals**),**

Arrays**.**asList**(**yVals**),** "Right Ear"**);**

// same as above:

**}**

**}**

/\*\*

\*

\*/

private void updatePlot**()** **{**

// Remove all current series from each plot

Iterator**<**XYSeries**>** iterator1 **=** mySimpleXYPlot**.**getSeriesSet**().**iterator**();**

**while** **(**iterator1**.**hasNext**())** **{**

XYSeries setElement **=** iterator1**.**next**();**

mySimpleXYPlot**.**removeSeries**(**setElement**);**

**}**

drawBackground**();**

**if** **(**leftEarButton**.**isChecked**())** **{**

// Create a formatter to use for drawing a series using

// LineAndPointRenderer

// and configure it from xml:

LineAndPointFormatter series1Format **=** **new** LineAndPointFormatter**();**

series1Format**.**setPointLabelFormatter**(new** PointLabelFormatter**());**

series1Format**.**configure**(**getApplicationContext**(),**

R**.**xml**.**line\_point\_formatter\_with\_plf1**);**

mySimpleXYPlot**.**clear**();**

series1 **=** **(**SimpleXYSeries**)** getSeries**();** // call getSeries function

mySimpleXYPlot**.**addSeries**(**series1**,** series1Format**);**

mySimpleXYPlot**.**redraw**();** // redraw series

**}**

**else** **{**

LineAndPointFormatter series2Format **=** **new** LineAndPointFormatter**();**

series2Format**.**setPointLabelFormatter**(new** PointLabelFormatter**());**

series2Format**.**configure**(**getApplicationContext**(),**

R**.**xml**.**line\_point\_formatter\_with\_plf2**);**

mySimpleXYPlot**.**clear**();**

series2 **=** **(**SimpleXYSeries**)** getSeries**();** // call getSeries function

// add a new series' to the xyplot:

mySimpleXYPlot**.**addSeries**(**series2**,** series2Format**);**

mySimpleXYPlot**.**redraw**();**

**}**

// Setup our Series with the selected number of elements

// series1 = new SimpleXYSeries(Arrays.asList(series1Numbers),

// SimpleXYSeries.ArrayFormat.Y\_VALS\_ONLY, "Us");

// series2 = new SimpleXYSeries(Arrays.asList(series2Numbers),

// SimpleXYSeries.ArrayFormat.Y\_VALS\_ONLY, "Them");

**}**

private void clearPlot**()** **{**

// Remove all current series from each plot

Iterator**<**XYSeries**>** iterator1 **=** mySimpleXYPlot**.**getSeriesSet**().**iterator**();**

**while** **(**iterator1**.**hasNext**())** **{**

XYSeries setElement **=** iterator1**.**next**();**

mySimpleXYPlot**.**removeSeries**(**setElement**);**

**}**

**}**

private void drawBackground**()** **{**

RectF rect **=** mySimpleXYPlot**.**getGraphWidget**().**getGridRect**();**

BitmapShader myShader **=** **new** BitmapShader**(**Bitmap**.**createScaledBitmap**(**

BitmapFactory**.**decodeResource**(**getResources**(),**

R**.**drawable**.**audiogram\_background**),** 1**,** **(**int**)** rect

**.**height**(),** **false),** Shader**.**TileMode**.**CLAMP**,**

Shader**.**TileMode**.**CLAMP**);**

Matrix m **=** **new** Matrix**();**

m**.**setTranslate**(**rect**.**left**,** rect**.**top**);**

myShader**.**setLocalMatrix**(**m**);**

mySimpleXYPlot**.**getGraphWidget**().**getGridBackgroundPaint**()**

**.**setShader**(**myShader**);**

**}**

public void graphSettings**()** **{**

mySimpleXYPlot**.**setTitle**(**"Audiogram"**);**

mySimpleXYPlot**.**setRangeLabel**(**"[dB] Hearing level"**);**

mySimpleXYPlot**.**setDomainLabel**(**"Frequency Hz"**);**

mySimpleXYPlot**.**setDomainBoundaries**(**0**,** 8000**,** BoundaryMode**.**FIXED**);**

mySimpleXYPlot**.**setRangeBoundaries**(**120**,** 0**,** BoundaryMode**.**FIXED**);**

mySimpleXYPlot**.**setDomainStep**(**XYStepMode**.**INCREMENT\_BY\_VAL**,** 1000**);**

// mySimpleXYPlot.setRangeStep(XYStepMode.INCREMENT\_BY\_VAL, 10);

mySimpleXYPlot**.**getBackgroundPaint**().**setColor**(**Color**.**WHITE**);**

mySimpleXYPlot**.**getGraphWidget**().**getBackgroundPaint**()**

**.**setColor**(**Color**.**WHITE**);**

mySimpleXYPlot**.**getGraphWidget**().**getGridBackgroundPaint**()**

**.**setColor**(**Color**.**WHITE**);**

mySimpleXYPlot**.**getGraphWidget**().**getDomainLabelPaint**()**

**.**setColor**(**Color**.**BLACK**);**

mySimpleXYPlot**.**getGraphWidget**().**getRangeLabelPaint**()**

**.**setColor**(**Color**.**BLACK**);**

mySimpleXYPlot**.**getGraphWidget**().**getDomainOriginLabelPaint**()**

**.**setColor**(**Color**.**BLACK**);**

mySimpleXYPlot**.**getGraphWidget**().**getDomainOriginLinePaint**()**

**.**setColor**(**Color**.**BLACK**);**

mySimpleXYPlot**.**getTitleWidget**().**getLabelPaint**().**setColor**(**Color**.**BLACK**);**

mySimpleXYPlot**.**getTitleWidget**().**getLabelPaint**().**setTextSize**(**15**);**

mySimpleXYPlot**.**getTitleWidget**().**setHeight**(**30**);**

mySimpleXYPlot**.**getTitleWidget**().**setWidth**(**400**);**

mySimpleXYPlot**.**getGraphWidget**().**setPaddingTop**(**8**);**

mySimpleXYPlot**.**getGraphWidget**().**setPaddingBottom**(**12**);**

mySimpleXYPlot**.**getGraphWidget**().**setPaddingLeft**(-**8**);**

mySimpleXYPlot**.**getGraphWidget**().**setPaddingRight**(**18**);**

mySimpleXYPlot**.**getDomainLabelWidget**().**getLabelPaint**().**setTextSize**(**15**);**

mySimpleXYPlot**.**getDomainLabelWidget**().**setWidth**(**100**);**

mySimpleXYPlot**.**getDomainLabelWidget**().**setHeight**(**18**);**

mySimpleXYPlot**.**getDomainLabelWidget**().**getLabelPaint**()**

**.**setColor**(**Color**.**BLACK**);**

mySimpleXYPlot**.**getRangeLabelWidget**().**getLabelPaint**().**setTextSize**(**15**);**

mySimpleXYPlot**.**getRangeLabelWidget**().**setWidth**(**50**);**

mySimpleXYPlot**.**getRangeLabelWidget**().**setHeight**(**500**);**

mySimpleXYPlot**.**getRangeLabelWidget**().**getLabelPaint**()**

**.**setColor**(**Color**.**BLACK**);**

**}**

public void startUp**()** **{**

AlertDialog**.**Builder alertbox **=** **new** AlertDialog**.**Builder**(**

HearingTestActivity**.this);**

alertbox**.**setIcon**(**R**.**drawable**.**ic\_launcher**);**

alertbox**.**setMessage**(**"You will hear a series of tones for 5 seconds each."

**+** "\n\nSelect whether you can hear or cannot hear"

**+** "\n\nAt clearest dB you can hear a tone then choose select"**);**

alertbox**.**setTitle**(**"Getting started"**);**

alertbox**.**setPositiveButton**(**"Ok"**,** **new** DialogInterface**.**OnClickListener**()** **{**

public void onClick**(**DialogInterface arg0**,** int arg1**)** **{**

// finish used for destroyed activity

**}**

**});**

alertbox**.**show**();**

**}**

// Method for alert dialog, when user exits

public boolean onKeyDown**(**int keyCode**,** KeyEvent event**)** **{**

**if** **((**keyCode **==** KeyEvent**.**KEYCODE\_BACK**))** **{**

AlertDialog**.**Builder alertbox **=** **new** AlertDialog**.**Builder**(**

HearingTestActivity**.this);**

alertbox**.**setIcon**(**R**.**drawable**.**ic\_launcher**);**

alertbox**.**setTitle**(**"Cancel the test?"**);**

alertbox**.**setPositiveButton**(**"Yes"**,**

**new** DialogInterface**.**OnClickListener**()** **{**

public void onClick**(**DialogInterface arg0**,** int arg1**)** **{**

// finish used for destroyed activity

finish**();**

**}**

**});**

alertbox**.**setNegativeButton**(**"No"**,**

**new** DialogInterface**.**OnClickListener**()** **{**

public void onClick**(**DialogInterface arg0**,** int arg1**)** **{**

Toast**.**makeText**(**getApplicationContext**(),**

"Test continue"**,** Toast**.**LENGTH\_SHORT**).**show**();**

**}**

**});**

alertbox**.**show**();**

**}**

**return** **super.**onKeyDown**(**keyCode**,** event**);**

**}**

// onDestroy called when app exited

// we stop app from playing tone in background

@Override

protected void onDestroy**()** **{**

**super.**onDestroy**();**

Log**.**d**(**"HearingTest"**,** "onDestroy called"**);**

**try** **{**

frequencygen**.**audioTrack**.**release**();**

frequencygen**.**audioTrack **=** **null;**

frequencygen **=** **null;**

**}** **catch** **(**Exception e**)** **{**

Log**.**e**(**"HearingTest"**,** "onPause error: " **+** e**.**toString**());**

**}**

**}**

// onPause is when user leaves the current activity

// so we stop frequencygen class from generating tone

@Override

protected void onPause**()** **{**

**super.**onPause**();**

Log**.**d**(**"HearingTest"**,** "onPause called"**);**

**try** **{**

frequencygen**.**audioTrack**.**release**();**

frequencygen**.**audioTrack **=** **null;**

frequencygen **=** **null;**

**}** **catch** **(**Exception e**)** **{**

Log**.**e**(**"HearingTest"**,** "onPuse error: " **+** e**.**toString**());**

**}**

**}**

**}** // end of class

### MoreInfo.java

/\*\*

\* {This class runs the info activity and launches new intent based on what user chooses}

\*

\* @version Build {0.6} (26 March 2015)

\* @author Junaid Malik

\*/

package com**.**example**.**testtonegenv10**;**

**import** android**.**app**.**Activity**;**

**import** android**.**content**.**Intent**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**view**.**View**;**

public class MoreInfo **extends** Activity **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**moreinfo**);**

**}**

public void loadView1**(**View v**)** **{**

Intent intent **=** **new** Intent**(**MoreInfo**.this,** InfoView1**.**class**);**

startActivity**(**intent**);**

**}**

public void loadView2**(**View v**)** **{**

Intent intent **=** **new** Intent**(**MoreInfo**.this,** InfoView2**.**class**);**

startActivity**(**intent**);**

**}**

public void loadView3**(**View v**)** **{**

Intent intent **=** **new** Intent**(**MoreInfo**.this,** InfoView3**.**class**);**

startActivity**(**intent**);**

**}**

public void loadView4**(**View v**)** **{**

Intent intent **=** **new** Intent**(**MoreInfo**.this,** InfoView4**.**class**);**

startActivity**(**intent**);**

**}**

**}**

### MoreInfo2.java

package com**.**example**.**testtonegenv10**;**

**import** android**.**app**.**Activity**;**

**import** android**.**os**.**Bundle**;**

public class MoreInfo2 **extends** Activity **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**moreinfo2**);**

**}**

**}**

### StartTest.java

package com**.**example**.**testtonegenv10**;**

**import** android**.**app**.**Activity**;**

**import** android**.**content**.**Intent**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**view**.**View**;**

public class StartTest **extends** Activity **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**calibration\_info**);**

**}**

public void startTest**(**View v**)**

**{**

Intent intent **=** **new** Intent**(this,** HearingTestActivity**.**class**);**

startActivity**(**intent**);**

**}**

**}**

### InfoView1.java

/\*

\* {Simple class loads infoView1.java}

\*

\* @version Build {0.6} (26 March 2015)

\* @author Junaid Malik

\*/

package com**.**example**.**testtonegenv10**;**

**import** android**.**app**.**Activity**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**view**.**View**;**

**import** android**.**view**.**View**.**OnClickListener**;**

public class InfoView1 **extends** Activity **implements** OnClickListener **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**infoview1**);**

**}**

@Override

public void onClick**(**View v**)** **{**

// TODO Auto-generated method stub

**}**

**}**

### InfoView2.java

/\*

\* {Simple class loads infoView2.java}

\*

\* @version Build {0.6} (26 March 2015)

\* @author Junaid Malik

\*/

package com**.**example**.**testtonegenv10**;**

**import** android**.**app**.**Activity**;**

**import** android**.**os**.**Bundle**;**

**import** android**.**view**.**View**;**

**import** android**.**view**.**View**.**OnClickListener**;**

public class InfoView2 **extends** Activity **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**infoview2**);**

**}**

**}**

### InfoView3.java

package com**.**example**.**testtonegenv10**;**

**import** android**.**app**.**Activity**;**

**import** android**.**os**.**Bundle**;**

public class InfoView3 **extends** Activity **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**infoview3**);**

**}**

**}**

### InfoView4.java

package com**.**example**.**testtonegenv10**;**

**import** com**.**example**.**testtonegenv10**.**R**;**

**import** android**.**app**.**Activity**;**

**import** android**.**os**.**Bundle**;**

public class InfoView4 **extends** Activity **{**

@Override

public void onCreate**(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**infoview4**);**

**}**

**}**

## Appendix F: Log file of changes

Junaid Malik - Final Year Project 2015 - Hearing Aid App

------------------- MAIN PROJECT IS SAVED IN GIT HUB, BACKUP IN WORKSPACE ---------------

Changelog Legend:

-----------------------------------------------------------------------------------------

[+] = Added

[\*] = Changed

[^] = Moved

[=] = No Changes

[x] = Deleted

[!] = Bugs

------------------------------------------------------------------------------------------

(04/05/15) - Improvements - Version 1.09

[+] - increment for frequency array values

[\*] - change next frequency button to have method setFreqValue

------------------------------------------------------------------------------------------

(29/04/15) - Improvements - Version 1.08

[+] - increments yVals with simple (+= 1) etc

[+] - xVals are fixed and do need to be changed

[+] - added "rate this app" plug in by Keisuke Kobayashi - added library and main code

[+] - added strings for "rate this app"

[+] - added indexYval to control index position

[+] - next frequency always set 40

[+] - added strings and xml layout for morinfoview 2, also added to main menu

[+] - added info and new icon to main menu

------------------------------------------------------------------------------------------

(21/04/15) - Improvements - Version 1.05

[+] - added x and y graph to be updated using SimpleXYSeries

[\*] - formatted java files and xml files with cleaner layout

[\*] - updated app icons and image buttons

[\*] - added onPause and onDestroy methods for main class to stop playing tone in background

[\*] - changed headphone\_info to calibration\_info

[+] - added a custom SimpleXYSeries, removed code that limits x,y size

[+] - added preset frequency values as they never change

[+] - added colors.xml(values folder), nice\_button.xml and a drawable folder

[+] - removed multiple launchers from android manifest

------------------------------------------------------------------------------------------

(17/04/15) - Improvements - Version 1.0

[+] - Added radio buttons controlling left and right ear.

[+] - Updated icons with much cleaner look using MS Paint.

[+] - Added project to git hub.

[+] - Added list view of audiograms to load button

user can view old tests he has done and view them in a list

[+] - Added infoview4.xml which contains contact information.

[+] - Added SimpleXYSeries getSeries() method to add series which can be changed by user.

------------------------------------------------------------------------------------------

(31/03/15) - Improvements - Version 0.8 - 0.9

[+] - Fixed fragment issue by correct id and class

[+] - FragmentAudiogram does nothing, and FragmentAudiogram only for inflating view

[\*] - HeaingTestActivity contains implementation for both Audiogram and PlayTone fragments.

[+] - Added StartTest.java and headphone\_info.xml before main HearingTestActivity.java

[\*] - Fix and re-arranged Strings file to properly allocate id's

[+] -

------------------------------------------------------------------------------------------

(30/03/15) - Improvements - Version 0.7

[!] - App was crashing when fragment tag was added.

[!] - When removed it worked, so failure is when app renders fragment element

[+] - Fixed issue by importing correct android support library

[\*] - Files rename to FragmentPlayTone.java and playtone\_fragment.xml

------------------------------------------------------------------------------------------

(25/03/15) - Improvements - Version 0.6

[\*] - Audiogram.java and FrequencyGen.java have seperate xml file,

audiogram\_fragment and frequency\_fragment

[+] - Added hearingtest\_view which contains audiogram and frequency fragments

[\*] - Refactored class name of Audiogram, HearingTest and created MainFragment

class FragmentHearingTest

------------------------------------------------------------------------------------------

(23/03/15) - Improvements - Version 0.5

[+] - Created new classes for Audiogram, FrequencyGen and HearingTest.

[\*] - Frequency Tone now generated in its own class FrequencyGen.java

[\*] - Audiogram graph is now generated by Audiogram.java

[\*] - Test logic is to be designed in HearingTest.java

[\*] - each class should have own function.

[!] - Unable to have both classes acting on one layout

------------------------------------------------------------------------------------------

(23/02/15) - Improvements - Version 0.4

[+] - Implemented AndroidPlot library for XYPlot graph.

[+] - Improve comply to coding standards.

[!] - Unable to get graph to change as user presses button.

[!] -

------------------------------------------------------------------------------------------

(9/2/15) - Improvements - Version 0.3

[+] - Added MainMenu class.

[+] - Added activities for Information on hearing.

[+] - SplashScreen at start.

------------------------------------------------------------------------------------------

(1/2/15) - Improvements - Version 0.2

[+] - Class to generate simple pure tones added.

[+] - added buttons to play different tones.

------------------------------------------------------------------------------------------

(1/2/15) - Improvements - Version 0.1

------------------------------------------------------------------------------------------

[+] - Project created.

[+] - Sample projects tested.

------------------------------------------------------------------------------------------

## Appendix G: The AndroidManifest.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<manifest xmlns:android=**"http://schemas.android.com/apk/res/android"**

package=**"com.example.testtonegenv10"**

android:versionCode=**"10"**

android:versionName=**"10"** >

<uses-sdk

android:minSdkVersion=**"11"**

android:targetSdkVersion=**"16"** />

<application

android:allowBackup=**"true"**

android:icon=**"@drawable/ic\_launcher"**

android:label=**"@string/app\_name"**

android:theme=**"@style/AppTheme"** >

<activity

android:name=**"com.example.testtonegenv10.MainMenu"**

android:label=**"@string/app\_name"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.LAUNCHER"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.StartTest"**

android:label=**"@string/start\_test"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.HearingTestActivity"**

android:label=**"@string/testing\_screen"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.FragmentFrequency"**

android:label=**"@string/testing\_screen"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.FrequencyGenerator"**

android:label=**"@string/testing\_screen"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.FragmentAudiogram"**

android:label=**"@string/testing\_screen"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.LoadTestActivity"**

android:label=**"@string/loadprev\_test"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.MoreInfo"**

android:label=**"@string/moreinfo\_activity"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.MoreInfo2"**

android:label=**"@string/moreinfo2"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.InfoView1"**

android:label=**"@string/Infoview1"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.InfoView2"**

android:label=**"@string/Infoview2"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.InfoView3"**

android:label=**"@string/Infoview3"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

<activity

android:name=**"com.example.testtonegenv10.InfoView4"**

android:label=**"@string/Infoview4"** >

<intent-filter>

<action android:name=**"android.intent.action.MAIN"** />

<category android:name=**"android.intent.category.EMBED"** />

</intent-filter>

</activity>

</application>

</manifest>

## Appendix H: Strings Resource file

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<resources>

<string name=**"app\_name"**>**TestToneGen V10**</string>

<string name=**"splash"**>**SplashScreen**</string>

<string name=**"begin"**>**Begin a test**</string>

<string name=**"load"**>**Load test**</string>

<string name=**"about"**>**About**</string>

<string name=**"frequency\_fragment"**>**Frequency fragment**</string>

<string name=**"audiogram\_fragment"**>**Audiogram fragment**</string>

<string name=**"moreinfo\_activity"**>**Information on hearing**</string>

<string name=**"moreinfo2"**>**Human Auditory System**</string>

<string name=**"Infoview1"**>**What is hearing loss?**</string>

<string name=**"Infoview2"**>**Symptoms of hearing loss**</string>

<string name=**"Infoview3"**>**What is a hearing test?**</string>

<string name=**"Infoview4"**>**About developer**</string>

<string name=**"audiogram\_screen"**>**This screen displays the results**</string>

<string name=**"hello\_world"**>**\t\tWelcome to the \n\t\tHearing Test App**</string>

<string name=**"testing\_screen"**>**Test Screen**</string>

<string name=**"start\_test"**>**Test starting.**</string>

<string name=**"calibrate\_title"**>**Calibration**</string>

<string name=**"calibrate\_info"**>**For best results, its highly recommended that you use headphones for the remainder of this test.**</string>

<string name=**"calibrate\_button"**>**Click here to continue**</string>

<string name=**"loadprev\_test"**>**Load a test**</string>

<string name=**"loadtest\_title\_hint"**>**Enter a title for the test**</string>

<string name=**"load\_title\_label"**>**Title**</string>

<string name=**"load\_test\_date"**>**Date:**</string>

<string name=**"load\_test\_checked"**>**Checked:**</string>

<string name=**"load\_test\_title\_hint"**>**Enter a name for the test.**</string>

<string name=**"action\_settings"**>**Settings**</string>

<string name=**"button1"**>**500hz**</string>

<string name=**"button2"**>**1kHz**</string>

<string name=**"button3"**>**2kHz**</string>

<string name=**"button4"**>**3kHz**</string>

<string name=**"button5"**>**4kHz**</string>

<string name=**"button6"**>**Begin a test**</string>

<string name=**"leftEar"**>**Left**</string>

<string name=**"rightEar"**>**Right**</string>

<string name=**"infobutton1"**>**What is hearing loss?**</string>

<string name=**"infobutton2"**>**Symptoms of hearing loss**</string>

<string name=**"infobutton3"**>**What is the test?**</string>

<string name=**"infobutton4"**>**About developer**</string>

<string name=**"okbutton"**>**OK**</string>

<string name=**"infoView1"**>**\n•Hearing is the ability to perceive sounds. Sound occurs over a wide spectrum of frequencies.**

**The human ear is sensitive to a frequency band within that spectrum expressed in decibels**

**(dB).**

**\n\n•Frequencies capable of being heard by humans are called audio or sonic. The range is**

**typically considered to be between 20 Hz and 20 ,000 Hz (Hertz).**

**\n\n•Frequencies higher than**

**audio are referred to as ultrasonic, while frequencies below audio are referred to as**

**infrasonic. Loss of the ability to hear sound frequencies in the normal range of hearing is**

**called hearing impairment.\n\n**</string>

<string name=**"infoView2"**>**\nYou may have a hearing loss if:**

**\n\n•You hear people speaking but you have difficulty understanding what they are saying**

**\n\n•You are frequently having to ask people to repeat what they have said**

**\n\n•You feel left out of conversations**

**\n\n•You complain that others are mumbling**

**\n\n•You need to ask colleagues about the details of a meeting or presentation you attended**

**\n\n•You miss jokes because you didn\'t hear all of what was said**

**\n\n•Friends and family need you to turn the TV or radiodown**

**\n\n•You do not hear the telephone and or doorbell**

**\n\n•You have difficulty working out where sounds are coming from**

**\n\n•You have a history of exposure to loud noise e.g. working in the construction industry**

**\n\n•You understand more of what is being said if you are looking at the person speaking to you**

**\n\n•Soft sounds seem to have disappeared altogether in speech**</string>

<string name=**"infoView3"**>**\n•Pure tone audiometry is the standard assesment of an individual\'s hearing.\n\n•The results of a pure tone audiometry test are recorded onto a chart called an audiogram.**

**\n\n•It is an easy method to obtain and provide information about an individuals hearing across a range of frequencies.**

**\n\n•A pure tone is a tone that has a single specific frequency.This tone is determined by the rate, or speed at which the sound waves vibrate.**

**\n\n•A 4000 Hz tone has a higher frequency than the 500 Hz tone.**</string>

<string name=**"infoView4"**>**The system of the body that lets a person hear sound is called the auditory system. It is made up of four main parts.**

**The visible part of the ear and the ear canal make up the outer ear.**

**The area behind the eardrum is called the middle ear. It is made up of the three smallest bones in the body (malleus, incus and stapes) and the Eustachian tube, which leads from the middle ear to the back of the throat.**

**\n\nThe main organ for hearing, the cochlea, is in the inner ear. The cochlea is the sense organ for hearing like the nose is for smelling. The middle ear bones are connected to the cochlea and the cochlea is connected to the auditory nerve.**

**The auditory nerve is the nerve that runs from the inner ear (cochlea) to the brainstem. The cochlea sends signals to the auditory nerve; then the auditory nerve sends these signals to the brain.**

**Hearing loss can happen because of problems in any part of the auditory system.**</string>

<string name=**"infoView5"**>**Hearing is different than other senses like vision, smell and taste. The hearing system is about changing movement into a signal that the brain can understand. Sound travels through the air as vibrations in the air pressure (i.e. sound waves). It takes four steps through the auditory system before the brain can understand sounds.**

**\n\nStep 1: Sound moves into the ear through the outer ear and the ear canal. Once the sound has moved down the ear canal, the sound waves hit the eardrum, which makes the eardrum vibrate.**

**\n\nStep 2: When the eardrum vibrates, it moves the bones of the middle part of the ear (malleus, incus and stapes). The movement of the malleus and incus makes the stapes move in and out of the entrance of the inner ear, the oval window.**

**\n\nStep 3: The other side of the oval window, in the inner ear, is filled with fluid. As the stapes moves back and forth on the oval window, it makes waves. This changes the vibrations of the middle ear into hydraulic energy.**

**\n\nStep 4: The hydraulic energy of the waves makes tiny hair cells in the cochlea sway back and forth. The swaying action creates signals, which move through the auditory nerve then up the levels of the brainstem until they reach the brain.**</string>

<string name=**"rta\_dialog\_title"**>**Rate this app**</string>

<string name=**"rta\_dialog\_message"**>**If you enjoy this app, please take a moment to rate this app. It won\'t take more than a minute. Thank you for your support!**</string>

<string name=**"rta\_dialog\_ok"**>**Rate now**</string>

<string name=**"rta\_dialog\_cancel"**>**Remind later**</string>

<string name=**"rta\_dialog\_no"**>**No, thanks**</string>

</resources>

## Appendix I: Layout Files

### calibration\_info.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<RelativeLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

android:background=**"#FFFFFF"**

android:gravity=**"center"** >

<TextView

android:id=**"@+id/textView2"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentTop=**"true"**

android:layout\_centerHorizontal=**"true"**

android:layout\_marginTop=**"23dp"**

android:gravity=**"center"**

android:text=**"@string/calibrate\_title"**

android:textAppearance=**"?android:attr/textAppearanceLarge"**

android:textSize=**"50sp"** />

<TextView

android:id=**"@+id/textView1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentLeft=**"true"**

android:layout\_alignParentRight=**"true"**

android:layout\_below=**"@+id/textView2"**

android:layout\_marginTop=**"32dp"**

android:gravity=**"center"**

android:text=**"@string/calibrate\_info"**

android:textAppearance=**"?android:attr/textAppearanceLarge"** />

<Button

android:id=**"@+id/button1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignLeft=**"@+id/textView2"**

android:layout\_alignParentBottom=**"true"**

android:layout\_alignRight=**"@+id/textView2"**

android:layout\_marginBottom=**"32dp"**

android:background=**"@drawable/nice\_button"**

android:gravity=**"center"**

android:onClick=**"startTest"**

android:shadowColor=**"@android:color/black"**

android:text=**"@string/calibrate\_button"**

android:textColor=**"@android:color/white"** />

<ImageView

android:id=**"@+id/imageView1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignRight=**"@+id/button1"**

android:layout\_below=**"@+id/textView1"**

android:background=**"#FFFFFF"**

android:src=**"@drawable/head\_phones\_img"** />

</RelativeLayout>

### infoview1.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<ScrollView xmlns:android=**"http://schemas.android.com/apk/res/android"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"** >

<RelativeLayout

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

>

<TextView

android:id=**"@+id/infoView1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentLeft=**"true"**

android:layout\_alignParentTop=**"true"**

android:gravity=**"center\_vertical"**

android:text=**"@string/infoView1"**

android:textSize=**"20sp"** />

</RelativeLayout>

</ScrollView>

### infoview2.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<ScrollView xmlns:android=**"http://schemas.android.com/apk/res/android"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"** >

<RelativeLayout

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"**

>

<TextView

android:id=**"@+id/infoView2"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentTop=**"true"**

android:text=**"@string/infoView2"**

android:textSize=**"20sp"** />

</RelativeLayout>

</ScrollView>

### infoview3.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<ScrollView xmlns:android=**"http://schemas.android.com/apk/res/android"**

xmlns:tools=**"http://schemas.android.com/tools"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

tools:context=**".InfoView3"** >

<RelativeLayout

android:layout\_width=**"match\_parent"**

android:layout\_height=**"wrap\_content"** >

<TextView

android:id=**"@+id/infoView3"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentTop=**"true"**

android:text=**"@string/infoView3"**

android:textSize=**"20sp"** />

</RelativeLayout>

</ScrollView>

### infoview4.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<RelativeLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

android:layout\_width=**"match\_parent"**

android:layout\_height=**"match\_parent"**

android:orientation=**"vertical"** >

<ImageView

android:id=**"@+id/imageView1"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"match\_parent"**

android:scaleType=**"centerCrop"**

android:src=**"@drawable/infoview4"** />

<TextView

android:id=**"@+id/TextView03"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_alignParentBottom=**"true"**

android:layout\_centerHorizontal=**"true"**

android:layout\_marginBottom=**"37dp"**

android:text=**"junmalik109@gmail.com"**

android:textColor=**"#FFFFFF"**

android:textSize=**"20dp"** />

<TextView

android:id=**"@+id/textview"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_above=**"@+id/TextView02"**

android:layout\_centerHorizontal=**"true"**

android:text=**"Hearing Test App"**

android:textColor=**"#FFFFFF"**

android:textSize=**"25dp"** />

<TextView

android:id=**"@+id/TextView01"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_above=**"@+id/textview"**

android:layout\_centerHorizontal=**"true"**

android:text=**"Version 1.0"**

android:textColor=**"#FFFFFF"**

android:textSize=**"25dp"** />

<TextView

android:id=**"@+id/TextView02"**

android:layout\_width=**"wrap\_content"**

android:layout\_height=**"wrap\_content"**

android:layout\_above=**"@+id/TextView03"**

android:layout\_centerHorizontal=**"true"**

android:layout\_marginBottom=**"49dp"**

android:text=**"© 2015 - Junaid Malik"**

android:textColor=**"#FFFFFF"**

android:textSize=**"25dp"** />

</RelativeLayout>

### loadtest.xml

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<LinearLayout xmlns:android=**"http://schemas.android.com/apk/res/android"**

android:layout\_width=**"fill\_parent"**

android:layout\_height=**"fill\_parent"** >

<com.androidplot.xy.XYPlot

android:id=**"@+id/xyplot"**

android:layout\_width=**"fill\_parent"**

android:layout\_height=**"250dp"**

title=**"an xy plot"** />

</LinearLayout>

## Appendix J: Open Source Libraries

### SimpleXYSeries.java by AndroidPlot developers

/\*

\* Copyright 2012 AndroidPlot.com

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

// this class has been modified by Junaid Malik

// checking of value length to be the same size has been removed

package com**.**example**.**testtonegenv10**;**

**import** android**.**graphics**.**Canvas**;**

**import** android**.**util**.**Log**;**

**import** android**.**util**.**Pair**;**

**import** com**.**androidplot**.**Plot**;**

**import** com**.**androidplot**.**PlotListener**;**

**import** com**.**androidplot**.**xy**.**XYSeries**;**

**import** java**.**util**.**LinkedList**;**

**import** java**.**util**.**List**;**

**import** java**.**util**.**NoSuchElementException**;**

**import** java**.**util**.**concurrent**.**locks**.**ReentrantReadWriteLock**;**

/\*\*

\* A convenience class used to create instances of XYPlot generated from Lists of Numbers.

\*/

public class SimpleXYSeries **implements** XYSeries**,** PlotListener **{**

private static final String TAG **=** SimpleXYSeries**.**class**.**getName**();**

@Override

public void onBeforeDraw**(**Plot source**,** Canvas canvas**)** **{**

lock**.**readLock**().**lock**();**

**}**

@Override

public void onAfterDraw**(**Plot source**,** Canvas canvas**)** **{**

lock**.**readLock**().**unlock**();**

**}**

public enum ArrayFormat **{**

Y\_VALS\_ONLY**,**

XY\_VALS\_INTERLEAVED

**}**

private volatile LinkedList**<**Number**>** xVals **=** **new** LinkedList**<**Number**>();**

private volatile LinkedList**<**Number**>** yVals **=** **new** LinkedList**<**Number**>();**

private volatile String title **=** **null;**

private ReentrantReadWriteLock lock **=** **new** ReentrantReadWriteLock**(true);**

public SimpleXYSeries**(**String title**)** **{**

**this.**title **=** title**;**

**}**

/\*\*

\* Generates an XYSeries instance from the List of numbers passed in. This is a convenience class

\* and should only be used for static data models; it is not suitable for representing dynamically

\* changing data.

\*

\* @param model A List of Number elements comprising the data model.

\* @param format Format of the model. A format of Y\_VALS\_ONLY means that the array only contains y-values.

\* For this format x values are autogenerated using values of 0 through n-1 where n is the size of the model.

\* @param title Title of the series

\*/

public SimpleXYSeries**(**List**<?** **extends** Number**>** model**,** ArrayFormat format**,** String title**)** **{**

**this(**title**);**

setModel**(**model**,** format**);**

**}**

public SimpleXYSeries**(**List**<?** **extends** Number**>** xVals**,** List**<?** **extends** Number**>** yVals**,** String title**)** **{**

**this(**title**);**

**if(**xVals **==** **null** **||** yVals **==** **null)** **{**

**throw** **new** IllegalArgumentException**(**"Neither the xVals nor the yVals parameters may be null."**);**

**}**

// if(xVals.size() != yVals.size()) {

// throw new IllegalArgumentException("xVals and yVals List parameters must be of the same size.");

// }

**this.**xVals**.**addAll**(**xVals**);**

**this.**yVals**.**addAll**(**yVals**);**

**}**

/\*\*

\* Use index value as xVal, instead of explicit, user provided xVals.

\*/

public void useImplicitXVals**()** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

xVals **=** **null;**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

/\*\*

\* Use the provided list of Numbers as yVals and their corresponding indexes as xVals.

\* @param model A List of Number elements comprising the data model.

\* @param format Format of the model. A format of Y\_VALS\_ONLY means that the array only contains y-values.

\* For this format x values are autogenerated using values of 0 through n-1 where n is the size of the model.

\*/

public void setModel**(**List**<?** **extends** Number**>** model**,** ArrayFormat format**)** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

// empty the current values:

//xVals.clear();

xVals **=** **null;**

yVals**.**clear**();**

// make sure the new model has data:

**if** **(**model **==** **null** **||** model**.**size**()** **==** 0**)** **{**

**return;**

**}**

**switch** **(**format**)** **{**

// array containing only y-vals. assume x = index:

**case** Y\_VALS\_ONLY**:**

**for(**Number n **:** model**)** **{**

yVals**.**add**(**n**);**

**}**

/\*for (int i = 0; i < model.size(); i++) {

//xVals.add(i);

yVals.add(model.get(i));

}\*/

**break;**

// xy interleaved array:

**case** XY\_VALS\_INTERLEAVED**:**

**if** **(**xVals **==** **null)** **{**

xVals **=** **new** LinkedList**<**Number**>();**

**}**

**if** **(**model**.**size**()** **%** 2 **!=** 0**)** **{**

**throw** **new** IndexOutOfBoundsException**(**"Cannot auto-generate series from odd-sized xy List."**);**

**}**

// always need an x and y array so init them now:

int sz **=** model**.**size**()** **/** 2**;**

**for** **(**int i **=** 0**,** j **=** 0**;** i **<** sz**;** i**++,** j **+=** 2**)** **{**

xVals**.**add**(**model**.**get**(**j**));**

yVals**.**add**(**model**.**get**(**j **+** 1**));**

**}**

**break;**

**default:**

**throw** **new** IllegalArgumentException**(**"Unexpected enum value: " **+** format**);**

**}**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

/\*\*

\* Sets individual x value based on index

\* @param value

\* @param index

\*/

public void setX**(**Number value**,** int index**)** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

xVals**.**set**(**index**,** value**);**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

/\*\*

\* Sets individual y value based on index

\* @param value

\* @param index

\*/

public void setY**(**Number value**,** int index**)** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

yVals**.**set**(**index**,** value**);**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

/\*\*

\* Sets xy values based on index

\* @param xVal

\* @param yVal

\* @param index

\*/

public void setXY**(**Number xVal**,** Number yVal**,** int index**)** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

yVals**.**set**(**index**,** yVal**);**

xVals**.**set**(**index**,** xVal**);**

**}** **finally** **{**lock**.**writeLock**().**unlock**();}**

**}**

public void addFirst**(**Number x**,** Number y**)** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

**if** **(**xVals **!=** **null)** **{**

xVals**.**addFirst**(**x**);**

**}**

yVals**.**addFirst**(**y**);**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

/\*\*

\*

\* @return Pair<Number, Number> with first equal to x-val and second equal to y-val.

\*/

public Pair**<**Number**,** Number**>** removeFirst**()** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

**if** **(**size**()** **<=** 0**)** **{**

**throw** **new** NoSuchElementException**();**

**}**

**return** **new** Pair**<**Number**,** Number**>(**xVals **!=** **null** **?** xVals**.**removeFirst**()** **:** 0**,** yVals**.**removeFirst**());**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

public void addLast**(**Number x**,** Number y**)** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

**if** **(**xVals **!=** **null)** **{**

xVals**.**addLast**(**x**);**

**}**

yVals**.**addLast**(**y**);**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

/\*\*

\*

\* @return Pair<Number, Number> with first equal to x-val and second equal to y-val.

\*/

public Pair**<**Number**,** Number**>** removeLast**()** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

**if** **(**size**()** **<=** 0**)** **{**

**throw** **new** NoSuchElementException**();**

**}**

**return** **new** Pair**<**Number**,** Number**>(**xVals **!=** **null** **?** xVals**.**removeLast**()** **:** yVals**.**size**()** **-** 1**,** yVals**.**removeLast**());**

**}** **finally** **{**

lock**.**writeLock**().**unlock**();**

**}**

**}**

@Override

public String getTitle**()** **{**

**return** title**;**

**}**

public void setTitle**(**String title**)** **{**

lock**.**writeLock**().**lock**();**

**try** **{**

**this.**title **=** title**;**

**}** **finally** **{**lock**.**writeLock**().**unlock**();}**

**}**

@Override

public int size**()** **{**

**return** yVals **!=** **null** **?** yVals**.**size**()** **:** 0**;**

**}**

@Override

public Number getX**(**int index**)** **{**

**return** xVals **!=** **null** **?** xVals**.**get**(**index**)** **:** index**;**

**}**

@Override

public Number getY**(**int index**)** **{**

**return** yVals**.**get**(**index**);**

**}**

**}**

### RateThisApp.java by Keisuke Kobayashi

/\*

\* Copyright 2013 Keisuke Kobayashi

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package com**.**kskkbys**.**rate**;**

**import** java**.**util**.**Date**;**

**import** com**.**example**.**testtonegenv10**.**R**;**

**import** android**.**app**.**AlertDialog**;**

**import** android**.**content**.**Context**;**

**import** android**.**content**.**DialogInterface**;**

**import** android**.**content**.**DialogInterface**.**OnCancelListener**;**

**import** android**.**content**.**DialogInterface**.**OnClickListener**;**

**import** android**.**content**.**Intent**;**

**import** android**.**content**.**SharedPreferences**;**

**import** android**.**content**.**SharedPreferences**.**Editor**;**

**import** android**.**net**.**Uri**;**

**import** android**.**util**.**Log**;**

/\*\*

\* RateThisApp<br>

\* A library to show the app rate dialog

\* @author Keisuke Kobayashi <k.kobayashi.122@gmail.com>

\*

\*/

public class RateThisApp **{**

private static final String TAG **=** RateThisApp**.**class**.**getSimpleName**();**

private static final String PREF\_NAME **=** "RateThisApp"**;**

private static final String KEY\_INSTALL\_DATE **=** "rta\_install\_date"**;**

private static final String KEY\_LAUNCH\_TIMES **=** "rta\_launch\_times"**;**

private static final String KEY\_OPT\_OUT **=** "rta\_opt\_out"**;**

private static Date mInstallDate **=** **new** Date**();**

private static int mLaunchTimes **=** 0**;**

private static boolean mOptOut **=** **false;**

/\*\*

\* Days after installation until showing rate dialog

\*/

public static final int INSTALL\_DAYS **=** 7**;**

/\*\*

\* App launching times until showing rate dialog

\*/

public static final int LAUNCH\_TIMES **=** 10**;**

/\*\*

\* If true, print LogCat

\*/

public static final boolean DEBUG **=** **false;**

/\*\*

\* Call this API when the launcher activity is launched.<br>

\* It is better to call this API in onStart() of the launcher activity.

\*/

public static void onStart**(**Context context**)** **{**

SharedPreferences pref **=** context**.**getSharedPreferences**(**PREF\_NAME**,** Context**.**MODE\_PRIVATE**);**

Editor editor **=** pref**.**edit**();**

// If it is the first launch, save the date in shared preference.

**if** **(**pref**.**getLong**(**KEY\_INSTALL\_DATE**,** 0**)** **==** 0L**)** **{**

Date now **=** **new** Date**();**

editor**.**putLong**(**KEY\_INSTALL\_DATE**,** now**.**getTime**());**

log**(**"First install: " **+** now**.**toString**());**

**}**

// Increment launch times

int launchTimes **=** pref**.**getInt**(**KEY\_LAUNCH\_TIMES**,** 0**);**

launchTimes**++;**

editor**.**putInt**(**KEY\_LAUNCH\_TIMES**,** launchTimes**);**

log**(**"Launch times; " **+** launchTimes**);**

editor**.**commit**();**

mInstallDate **=** **new** Date**(**pref**.**getLong**(**KEY\_INSTALL\_DATE**,** 0**));**

mLaunchTimes **=** pref**.**getInt**(**KEY\_LAUNCH\_TIMES**,** 0**);**

mOptOut **=** pref**.**getBoolean**(**KEY\_OPT\_OUT**,** **false);**

printStatus**(**context**);**

**}**

/\*\*

\* Show the rate dialog if the criteria is satisfied

\* @param context

\*/

public static void showRateDialogIfNeeded**(**final Context context**)** **{**

**if** **(**shouldShowRateDialog**())** **{**

showRateDialog**(**context**);**

**}**

**}**

/\*\*

\* Check whether the rate dialog shoule be shown or not

\* @return

\*/

private static boolean shouldShowRateDialog**()** **{**

**if** **(**mOptOut**)** **{**

**return** **false;**

**}** **else** **{**

**if** **(**mLaunchTimes **>=** LAUNCH\_TIMES**)** **{**

**return** **true;**

**}**

long threshold **=** INSTALL\_DAYS **\*** 24 **\*** 60 **\*** 60 **\*** 1000L**;** // msec

**if** **(new** Date**().**getTime**()** **-** mInstallDate**.**getTime**()** **>=** threshold**)** **{**

**return** **true;**

**}**

**return** **false;**

**}**

**}**

/\*\*

\* Show the rate dialog

\* @param context

\*/

public static void showRateDialog**(**final Context context**)** **{**

AlertDialog**.**Builder builder **=** **new** AlertDialog**.**Builder**(**context**);**

builder**.**setTitle**(**R**.**string**.**rta\_dialog\_title**);**

builder**.**setMessage**(**R**.**string**.**rta\_dialog\_message**);**

builder**.**setPositiveButton**(**R**.**string**.**rta\_dialog\_ok**,** **new** OnClickListener**()** **{**

@Override

public void onClick**(**DialogInterface dialog**,** int which**)** **{**

String appPackage **=** context**.**getPackageName**();**

Intent intent **=** **new** Intent**(**Intent**.**ACTION\_VIEW**,** Uri**.**parse**(**"https://play.google.com/store/apps/details?id=" **+** appPackage**));**

context**.**startActivity**(**intent**);**

setOptOut**(**context**,** **true);**

**}**

**});**

builder**.**setNeutralButton**(**R**.**string**.**rta\_dialog\_cancel**,** **new** OnClickListener**()** **{**

@Override

public void onClick**(**DialogInterface dialog**,** int which**)** **{**

clearSharedPreferences**(**context**);**

**}**

**});**

builder**.**setNegativeButton**(**R**.**string**.**rta\_dialog\_no**,** **new** OnClickListener**()** **{**

@Override

public void onClick**(**DialogInterface dialog**,** int which**)** **{**

setOptOut**(**context**,** **true);**

**}**

**});**

builder**.**setOnCancelListener**(new** OnCancelListener**()** **{**

@Override

public void onCancel**(**DialogInterface dialog**)** **{**

clearSharedPreferences**(**context**);**

**}**

**});**

builder**.**create**().**show**();**

**}**

/\*\*

\* Clear data in shared preferences.<br>

\* This API is called when the rate dialog is approved or canceled.

\* @param context

\*/

private static void clearSharedPreferences**(**Context context**)** **{**

SharedPreferences pref **=** context**.**getSharedPreferences**(**PREF\_NAME**,** Context**.**MODE\_PRIVATE**);**

Editor editor **=** pref**.**edit**();**

editor**.**remove**(**KEY\_INSTALL\_DATE**);**

editor**.**remove**(**KEY\_LAUNCH\_TIMES**);**

editor**.**commit**();**

**}**

/\*\*

\* Set opt out flag. If it is true, the rate dialog will never shown unless app data is cleared.

\* @param context

\* @param optOut

\*/

private static void setOptOut**(**final Context context**,** boolean optOut**)** **{**

SharedPreferences pref **=** context**.**getSharedPreferences**(**PREF\_NAME**,** Context**.**MODE\_PRIVATE**);**

Editor editor **=** pref**.**edit**();**

editor**.**putBoolean**(**KEY\_OPT\_OUT**,** optOut**);**

editor**.**commit**();**

**}**

/\*\*

\* Print values in SharedPreferences (used for debug)

\* @param context

\*/

private static void printStatus**(**final Context context**)** **{**

SharedPreferences pref **=** context**.**getSharedPreferences**(**PREF\_NAME**,** Context**.**MODE\_PRIVATE**);**

log**(**"\*\*\* RateThisApp Status \*\*\*"**);**

log**(**"Install Date: " **+** **new** Date**(**pref**.**getLong**(**KEY\_INSTALL\_DATE**,** 0**)));**

log**(**"Launch Times: " **+** pref**.**getInt**(**KEY\_LAUNCH\_TIMES**,** 0**));**

log**(**"Opt out: " **+** pref**.**getBoolean**(**KEY\_OPT\_OUT**,** **false));**

**}**

/\*\*

\* Print log if enabled

\* @param message

\*/

private static void log**(**String message**)** **{**

**if** **(**DEBUG**)** **{**

Log**.**v**(**TAG**,** message**);**

**}**

**}**

**}**

# Bibliography

[1] Pure tone air and bone conduction threshold audiometry with and without masking - British Society of Audiology. 2015. Available at: <http://www.thebsa.org.uk/resources/pure-tone-air-bone-conduction-threshold-audiometry-without-masking/>

[Accessed 20 May 2015]

[2] Suzanne Purdy Head of Speech Science University of Auckland, Guideline for diagnosing occupational noise induced hearing loss Part 3: Audiometric standards Available at: <http://www.acc.co.nz/PRD_EXT_CSMP/groups/external_communications/documents/reference_tools/wpc091007.pdf>

[Accessed 20 May 2015]

[3] Hearing Tests (Audiometry) | Health | Patient.co.uk. 2015. Hearing Tests (Audiometry) | Health | Patient.co.uk*.* Available at: <http://www.patient.co.uk/health/hearing-tests-audiometry>

[Accessed 20 May 2015]

[4] Types of Tests Used to Evaluate Hearing in Children and Adults. Available at: <http://www.asha.org/public/hearing/Types-of-Tests-Used-to-Evaluate-Hearing/>

[Accessed 20 May 2015]

[5] Androidplot — Bitbucket. Available at: <https://bitbucket.org/androidplot/androidplot>

[Accessed 20 May 2015]

[6] Marble Mice: Generate And Play A Tone In Android. Available at: <http://www.marblemice.blogspot.ie/2010/04/generate-and-play-tone-in-android.html>

[Accessed 26 May 2015]

[7] Dr Béatrice Duthey, Ph.D., 20th February2013, Priority Medicines for Europe and the World, “A Public Health Approach to Innovation", Update on 2004 Background Paper, 6.21 Hearing Loss

Available at: <http://www.who.int/medicines/areas/priority_medicines/BP6_21Hearing.pdf>

[Accessed 26 May 2015]

[8] Application Development for Android Available at: <http://www.who.int/medicines/areas/priority_medicines/BP6_21Hearing.pdf>

[Accessed 28 May 2015]

[9] Prevalence of age-related hearing loss in Europe: a review – Springer. Available at: <http://link.springer.com/article/10.1007%2Fs00405-011-1597-8>.

[Accessed 28 May 2015]

[10] Evaluation of the Social and Economic Costs of Hearing Impairment by Shield (2006 - Google Scholar. 2015. Available at: https://scholar.google.com/scholar?q=Evaluation+of+the+Social+and+Economic+Costs+of+Hearing+Impairment%E2%80%9D+by+Shield+

[Accessed 28 May 2015]

[11] Human Ear Diagram Available at: <http://www.rudyard.org/wpcontent/uploads/2013/09/human-ear-diagram.jpg>

[Accessed 28 May 2015]

[12] Pete Goodliffe, 2014. Becoming a Better Programmer: A Handbook for People Who Care About Code. 1 Edition. O'Reilly Media.

[13] Marko Gargenta, 2011. Learning Android. 1 Edition. O'Reilly Media.

[14] Portable Digital Audiometer Supplier Hearing Aid Professional Physician's Office <http://www.alibaba.com/product-detail/Portable-Digital-Audiometer-Supplier-Hearing-Aid_1233690528/showimage.html>

[Accessed 30 May 2015]

[15] Noise-induced Hearing <https://www.nidcd.nih.gov/staticresources/health/hearing/NIDCD-Noise-Induced-Hearing-Loss.pdf>

[Accessed 30 May 2015]

[16] Git client for Mac and Windows | Atlassian. Available at: <https://www.atlassian.com/software/sourcetree/overview?_mid=e0734c30de09bb01fe8ee8625dc35ca4&gclid=CIeZgeqR6sUCFYJZ2wodDFIAxg>

[Accessed 30 May 2015]

[17] Guide to Agile Practices. 2015. Guide to Agile Practices. Available at: <http://guide.agilealliance.org/guide/storymap.html>

[Accessed 31 May 2015]

[18] Pivotal Tracker | Agile Project Management. Available at: <http://www.pivotaltracker.com/>

[Accessed 31 May 2015]

[19] Arafura Audiology & Alice Audiology | Interpreting an audiogram. [ONLINE] Available at: <http://www.arafuraaudiology.com.au/interpretinganaudiogram.html>.

[Accessed 31 May 2015]

[20] Available at: <http://www.chimehealth.co.uk/web/data/audiogram-hearing-loss-examples-2.pdf>. [Accessed 31 May 2015]

[21] Pencil Project. Available at: <http://pencil.evolus.vn/>. [Accessed 31 May 2015]

[22] Paul Tobin, 2007, PSpice for Digital Signal Processing, Edition 1, Morgan & Claypool Publishers.

[23] AudioTrack | Android Develop. Available at: <http://developer.android.com/reference/android/media/AudioTrack.html>.

[Accessed 31 May 2015]

[24] Graph View - Summary & Features. [ONLINE] Available at: <http://www.android-graphview.org/>.

[Accessed 31 May 2015]

[25] Online Tone Generator - Free, Simple and Easy to Use.. [ONLINE] Available at: <http://www.onlinetonegenerator.com/>.

[Accessed 01 June 2015]