ASSIGNMENT 2
CODE REPORT

Keegan George / 301310019

RECYCLER VIEW – ALL SENSORS

The RecyclerView is comprised of many elements. It requires a Data Model, Layout Manager, Adapter, View Holder and some other components. The XML contains code for the basic recycler view. The DataItemAdapter class is responsible for pulling the views of items in the Data Item class (data set) and presenting the items on the screen through the view holder. Inside the DataItemAdapter class is an inner class called the ViewHolder class. This is responsible for the inflated single recycler view item that is converted to Java code. There are two methods in the DataItemAdapter class that are responsible for the click events. One is for long clicks, which on a long press is set to display a toast indicating the item name. The onClickListener, is for short clicks in which it creates a new explicit intent directing the user to another activity. Passed into this as extras is the sensorKey and the item-id in order to allow the user to view the specific sensor they clicked on. All the sensors of the device are added to the recycler view in the Home Fragment class. There is an ArrayList consisting of all the sensor names that are added to the list after being taken from the SensorManager. After this, the Data item adapter is instantiated, and the recycler view is set to it, which displays the Sensor ArrayList in the recycler view. All these sensors are **not** hardcoded because it is searched for in the device when the activity is created, and it is placed in the recycler view based on what sensors are available on the device. This prevents any errors from occurring because of missing sensors that could have been hardcoded.

SENSOR DETAILS

The sensor details are all handled in the Sensor Details activity. This is called through the explicit intent when the user clicks on a specific sensor in the Home Fragment. As explained above, the sensors activity is displayed when an onClick event occurs in the recycler view. The sensor details class is responsible for expanding on the details of the enumerated sensors from the Home Fragment. It contains TextView fields representing both the sensor information and sensor values. A link is made from XML to Java for these TextViews. Also, sensors are acquired from all the list and a reference is made to get the current sensor being clicked on using the key. A string of all the sensorDetails is available for displaying information including the sensor's name, type, version, vendor, power, resolution, minimum delay, and maximum range. The string is set in an HTML format in order to bold the titles, and the string is set to the TextView from HTML. Another method is included called

onSensorChanged. This is responsible for updating the sensor's values and updating the TextView. An array of floats takes in the values, a string for the default value is instantiated, and a for loop cycles through changes in values and updates the TextView by converted the float to a String and setting the TextView as the current value.

LIGHT SENSOR

Details regarding the light sensor is all contained mostly in the LightFragment class. This class is responsible for updating the current light sensor values, and it is also responsible for playing a beep sound when the light sensor is completely covered. Other than the on resume method that updates the TextView, the primary class responsible for most of the work is the onSensorChanged(SensorEvent event) method. This class ensures the current sensor is the light sensor and sets the TextView as the current light sensor value, as well as updates it. It uses an array of the values to check to play a beep from the default notification sound from the Ringtone Manager, only if the sensor value is 0.

VIBRATION

The VibrationFragment class is the class that is responsible for the Vibration events. It instantiates the XML of a vibrating image and a non-vibrating phone image and updates them based on if the phone is flat on the table or not. The onSensorChanged method is responsible for checking if the sensor type is the accelerometer and has many conditional statements that check for certain accelerometer values which are common when a device is flat on the table, and if the condition is true then a vibration from the vibrator occurs for 5 seconds.

DEVICE MOVEMENT

The MoveFragment class is the primary class responsible for the device movement event. This fragment implements the onClickListener and SensorEventListener interfaces. It contains variables for linking XML code to Java. However, more importantly, there are variables linking to the accelerometer sensor, float values for the final and initial movements of the device, dampener values, and a Boolean to check if the device is in current movement. The onSensorChanged method will check if the

accelerometer values are being changed to clarify if the device is in motion. If the device is in motion it will update the text from the button press event, which is taken care of in the onClick method.

ENVIRONMENTAL NOISE

The EnviroFragment class is the fragment class that is responsible for updating the environmental noise event. The class implements the View.OnClickListener interface and mainly has variables referring to the XML TextView and the MediaRecorder. OnActivityCreated will create a new instance of the media recorder and link the XML to Java for the Button and TextView. The onClick method contains most of the code. It has code for setting the Media Recorder's audio source, output format, audio encoder, and output file. It prepares the media recorder which is safely kept under a try-catch loop. It will then start recording upon button press at maximum amps. It then will record the audio checking for a certain amount of time and if the maximum amplitude heard is greater or less than a certain amount, then it will update the TextView accordingly.