**Dominik’s Code - Reviewed by Irusha, Derek, and Amr**

**OverviewPageFragment.java**

Overall Comments:

* Add Descriptions for Method Names
* No comments

package com.lifeofpi.lifeband;  
  
import android.os.Bundle;  
import android.support.v4.widget.SwipeRefreshLayout;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import org.json.JSONObject;  
  
/\*\*  
 \* Created by dominikschmidtlein on 11/4/2015.  
 \*/  
public class OverviewPageFragment extends PageFragment {  
 public static final String NAME = "Overview";  
  
 private SwipeRefreshLayout swipeRefreshLayout;  
 private TextView currentHeartbeatTextView;  
 private TextView currentRespirationTextView;  
 private TextView currentAccelerationTextView;

// Add comments to describe the function of the method..  
 public static OverviewPageFragment newInstance(int page) {  
 return (OverviewPageFragment) PageFragment.newInstance(page, new OverviewPageFragment());  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.fragment\_overview\_tab, container, false);  
 swipeRefreshLayout = (SwipeRefreshLayout) view;  
 currentHeartbeatTextView = (TextView) view.findViewById(R.id.currentHeartbeatTextView);  
 currentRespirationTextView = (TextView) view.findViewById(R.id.currentRespirationTextView);  
 currentAccelerationTextView = (TextView) view.findViewById(R.id.currentAccelerationTextView);  
  
 swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {  
 @Override  
 public void onRefresh() {  
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 JSONObject latestDataJSON = new UDPHelper().getLatestDataJSON;  
 JSONObject jSONData = null;  
 if (UDPHelper.sendUDP((MainActivity) getActivity(), latestDataJSON, getIP(), getPort()))  
 jSONData = UDPHelper.receiveUDP((MainActivity) getActivity(), getPort(), MainActivity.RECEIVE\_PERIOD);  
 if (jSONData != null) {  
 try {  
 JSONObject data = jSONData.getJSONObject("data");  
 final double pulse = data.getDouble("pulse");  
 final double resp = data.getDouble("resp");  
 final double acc = data.getDouble("accell");  
 updateOverview(pulse, resp, acc);  
 } catch (Exception e) {  
 Log.e(MainActivity.TAG, "data extraction failed", e);  
 }  
 }  
 endRefresh();  
 }  
 }).start();  
 }  
 });  
  
 return view;  
 }  
  
 private String getStringFromResources(int id){  
 return getActivity().getResources().getString(id);  
 }  
  
 private int getPort(){  
 Log.d("TAG", ((MainActivity) getActivity()).sharedPreferences.getString(getStringFromResources(R.string.port\_key), getStringFromResources(R.string.port\_default)));  
 return Integer.valueOf(((MainActivity) getActivity()).sharedPreferences.getString(getStringFromResources(R.string.port\_key), getStringFromResources(R.string.port\_default)));  
 }  
  
 private String getIP(){  
 return ((MainActivity)getActivity()).sharedPreferences.getString(getStringFromResources(R.string.ip\_key), getStringFromResources(R.string.ip\_default));  
 }   
  
 public void updateOverview(final double pulse, final double resp, final double acc) {  
 getActivity().runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 currentHeartbeatTextView.setText(Math.round(pulse) + "");  
 currentRespirationTextView.setText(Math.round(resp) + "");  
 currentAccelerationTextView.setText(Math.round(acc) + "");  
 }  
 });  
 }  
  
 public void endRefresh() {  
 getActivity().runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 swipeRefreshLayout.setRefreshing(false);  
 }  
 });  
 }  
}

//please have more comments to explain

* Functionality
  + Overall functionality works according to the project specifications
  + Input and Output network communications conform to established data protocol
  + Code straightforward to understand
  + No sections of code incomplete
* Comments
  + Meta data at the top (Name, Date Modified)
  + Comments are comprehensible and add something to the maintainability of the code
  + Comments are neither too numerous nor verbose
  + Comments are in the right place and they are useful
* Code smells
  + Tabs and brackets are consistent
  + Code as modular as possible
  + Repetitive code has been factored out
  + Command classes have been designed to undertake one task only
  + The code does not use unjustifiable static methods/blocks
  + N/A Loops have a set length and correct termination conditions
  + Any unusual behavior or edge-case handling described
  + No hardcoded or Magic numbers present within reason
* Performance
  + No unnecessary loops
  + No possible replacement of recursive functions with sequential functions
  + the code was designed to perform as fast as possible
* Scope
  + No possible replacement of global variables to function variables
  + Variable types have been generalized where possible
  + Down casting used properly
* Unit Tests
  + Unit tests are present and correct
  + The code is unit testable
  + Testing the part that is implemented and checked for right performance
* Error/Exception handling
  + Common errors have been checked for
  + No zombie threads running
  + Any security concerns have been addressed
  + N/A data goes through many filters that makes sure the data does not contain errors
* Logging/Debugging Info
  + N/A Logging used appropriately (proper logging level and details)
  + N/A Are all data inputs checked (for the correct type, length, format, and range) and encoded?
* Code Library Usage
  + Frameworks have been used appropriately
  + Can any of the code be replaced with library functions?
  + the code was designed using the libraries and there is no code to be replaced