Henry Post

ITMD 455

Lab 1: Temperature Converter

Contents

MainActivity.java

```java
package me.henryfbp.temperatureconverter;

import android.content.Context;
import android.graphics.Color;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.LinearLayout;

import java.math.BigDecimal;
import java.math.MathContext;
import java.util.HashMap;
import java.util.Map;

import me.henryfbp.temperatureconverter.lib.EditableTextWatcher;
import me.henryfbp.temperatureconverter.lib.HLib;
import me.henryfbp.temperatureconverter.lib.TemperatureElement;
import me.henryfbp.temperatureconverter.lib.TemperatureSolver;
import me.henryfbp.temperatureconverter.lib.TemperatureUnit;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        final Map<String, TemperatureElement> tempelems = new HashMap<>();
        final TemperatureSolver ts = new TemperatureSolver();
        final Context context = this.getApplicationContext();
        final MainActivity mainActivity = this;

        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.activity_main);

        Toolbar toolbar = this.findViewById(R.id.toolbar);
        this.setSupportActionBar(toolbar);

        LinearLayout templist = this.findViewById(R.id.linearLayoutTemperatureList);


        tempelems.put("fahrenheit", new TemperatureElement(this.getApplicationContext(), new
TemperatureUnit("f")));
        tempelems.put("celsius", new TemperatureElement(this.getApplicationContext(), new
TemperatureUnit("c")));
        tempelems.put("kelvin", new TemperatureElement(this.getApplicationContext(), new
TemperatureUnit("k")));

        // This will update the background color to match the temperature.
        tempelems.get("fahrenheit").editText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {

            }

            @Override
            public void afterTextChanged(Editable s) {

                View v = mainActivity.findViewById(R.id.root);

                // 0 is blue,
                // 100 is red.
                Float percent = tempelems.get("fahrenheit").getTemp().floatValue() / 100;

                // Avoid IllegalArgExceptions.
                if (percent > 1f) {
                    percent = 1f;
                }
                if (percent < 0f) {
                    percent = 0f;
                }

                //Mix the two colors.
                Color c = HLib.mixColors(
                        mainActivity.getColor(R.color.warm),
                        mainActivity.getColor(R.color.cold),
                        percent
                );

                Log.i("bg_color", percent.toString() + "->" + c.toString());

                // Apply the two colors.
                v.setBackgroundColor(
                        c.toArgb()
                );
            }
        });

        //  For each String <---> TemperatureElement, add a listener.
        for (Map.Entry<String, TemperatureElement> entry : tempelems.entrySet()) {

            final String k = entry.getKey();
            final TemperatureElement v = entry.getValue();

            EditableTextWatcher tw = new EditableTextWatcher() {


                @Override
                public void beforeTextChange(CharSequence s, int start, int count, int after) {

                }

                @Override
                protected void onTextChange(CharSequence s, int start, int before, int count) {

                }

                @Override
                public void afterTextChange(Editable s) {
                    Log.i(("main_tempTextEdit_" + k), s.toString());

                    //After text changes, update ALL temperatures that are not the selected one.
                    for (Map.Entry<String, TemperatureElement> entry : tempelems.entrySet()) {
                        String key = entry.getKey();
                        TemperatureElement value = entry.getValue();

                        // If we're not looking at ourselves, solve it!
                        if (!key.equalsIgnoreCase(k)) {

                            try {

                                TemperatureElement otherElem = tempelems.get(key);

                                BigDecimal solution = ts.solve(k, key, v.getTemp());

                                //Do NOT trigger the other element's EditText TextWatcher

otherElem.editText.removeTextChangedListener(otherElem.textWatcher);

                                // Change the text.
                                otherElem.setTemp(solution.round(new MathContext(4)));

                                // Re-register the textChangedListener.
                                otherElem.editText.addTextChangedListener(otherElem.textWatcher);
                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                        }
                    }
                }
            };

            v.editText.addTextChangedListener(tw);
            v.textWatcher = tw;
            templist.addView(v);
        }


        FloatingActionButton fab = this.findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        this.getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

EditableTextWatcher.java

```java
package me.henryfbp.temperatureconverter.lib;

import android.text.Editable;
import android.text.TextWatcher;


/**
 * https://stackoverflow.com/questions/9385081/how-can-i-change-the-edittext-text-without-
triggering-the-text-watcher/42928051#42928051
 */
public abstract class EditableTextWatcher implements TextWatcher {

    private boolean editing;

    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        if (this.editing)
            return;

        this.editing = true;
        try {
            this.beforeTextChange(s, start, count, after);
        } finally {
            this.editing = false;
        }
    }

    public abstract void beforeTextChange(CharSequence s, int start, int count, int after);

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        if (this.editing)
            return;

        this.editing = true;
        try {
            this.onTextChange(s, start, before, count);
        } finally {
            this.editing = false;
        }
    }

    protected abstract void onTextChange(CharSequence s, int start, int before, int count);

    @Override
    public void afterTextChanged(Editable s) {
        if (this.editing)
            return;

        this.editing = true;
        try {
            this.afterTextChange(s);
        } finally {
            this.editing = false;
        }
    }

    public boolean isEditing() {
        return this.editing;
    }
    protected abstract void afterTextChange(Editable s);
}
```

HLib.java

```java
package me.henryfbp.temperatureconverter.lib;

import android.graphics.Color;

import java.util.Random;

public class HLib {

    public static Color randomColor() {
        Random r = new Random();
        return Color.valueOf(r.nextFloat(), r.nextFloat(), r.nextFloat());
    }


    /*
     * Adapted from https://stackoverflow.com/a/17544748/4262535.
     *
     * Mixes two colors together.
     */
    public static Color mixColors(Color c1, Color c2, Float percent) {

        if (percent < 0f) {
            throw new IllegalArgumentException(percent.toString() + " < 0!");
        }
        if (percent > 1f) {
            throw new IllegalArgumentException(percent.toString() + " > 1!");
        }

        float inv_percent = 1.0f - percent;

        float r = (c1.red() * percent +
                c2.red() * inv_percent);

        float g = (c1.green() * percent +
                c2.green() * inv_percent);

        float b = (c1.blue() * percent +
                c2.blue() * inv_percent);

        return Color.valueOf(r, g, b);
    }

    public static Color mixColors(int c1, int c2, float percent) {
        return mixColors(Color.valueOf(c1), Color.valueOf(c2), percent);
    }


}
```

TemperatureElement.java

```java
package me.henryfbp.temperatureconverter.lib;

import android.content.Context;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.math.BigDecimal;

import me.henryfbp.temperatureconverter.R;

public class TemperatureElement extends LinearLayout {

    public TemperatureUnit unit;
    public EditText editText;
    public TextWatcher textWatcher;
    public TextView textView;

    public TemperatureElement(Context context, TemperatureUnit t) {
        super(context);

        this.unit = t;

        //inflate all layout contents from single_temperature.xml
        View v = LayoutInflater.from(this.getContext()).inflate(R.layout.single_temperature, null);

        this.addView(v); //add inflated view

        this.editText = v.findViewById(R.id.editTextTemperatureUnit);
        this.setTemp(new BigDecimal("-1"));

        this.textView = v.findViewById(R.id.textViewTemperatureUnit);
        this.textView.setText(this.unit.unit);

    }

    public BigDecimal getTemp() {
        try {
            return new BigDecimal(this.editText.getText().toString());

        } catch (NumberFormatException nfe) {
            return new BigDecimal(0);
        }
    }

    public void setTemp(BigDecimal t) {
        this.editText.setText(t.toPlainString());
    }

}
```

TemperatureSolver.java

```java
package me.henryfbp.temperatureconverter.lib;

import com.google.common.collect.ImmutableList;

import java.math.BigDecimal;
import java.util.HashMap;
import java.util.Map;

/***
 * Solve any temperature.
 */
public class TemperatureSolver {

    public Map<ImmutableList, TemperatureSolverSingle> map = new HashMap<>();

    public TemperatureSolver() { //TODO: Use linear algebra to automatically fill-in missing
formulas.
        this.addTemp(
                new TemperatureSolverSingle("celsius", "fahrenheit", "F(x) = ((9/5) * x) + 32"),
                new TemperatureSolverSingle("celsius", "kelvin", "F(x) = x + 273.15"),
                new TemperatureSolverSingle("fahrenheit", "kelvin", "F(x) = (x + 459.67) * (5/9)"),
                new TemperatureSolverSingle("fahrenheit", "celsius", "F(x) = (x - 32) * (5/9)"),
                new TemperatureSolverSingle("kelvin", "celsius", "F(x) = x - 273.15"),
                new TemperatureSolverSingle("kelvin", "fahrenheit", "F(x) = ((9/5) * x) - 459.67")
        );
    }

    public void addTemp(TemperatureSolverSingle... tsss) {
        for (TemperatureSolverSingle tss : tsss) {
            this.map.put(ImmutableList.of(tss.from, tss.to), tss);
        }
    }

    public BigDecimal solve(String from, String to, BigDecimal temp) {
        return this.map.get(ImmutableList.of(from, to)).solve(temp);
    }
}
```

TemperatureSolverSingle.java

```java
package me.henryfbp.temperatureconverter.lib;

import org.mariuszgromada.math.mxparser.Argument;
import org.mariuszgromada.math.mxparser.Expression;
import org.mariuszgromada.math.mxparser.Function;

import java.math.BigDecimal;

/***
 * Turn one temperature into another one.
 */
public class TemperatureSolverSingle {

    public String from;
    public String to;
    public Function f;

    public TemperatureSolverSingle(String from, String to, String func) {
        this.from = from;
        this.to = to;
        this.f = new Function(func);
    }

    /**
     * Default constructor, defaults to celsius->fahrenheit.
     */
    public TemperatureSolverSingle() {
        this.f = new Function("F(x) = ((9/5) * x) + 32");
        this.from = "celsius";
        this.to = "fahrenheit";
    }

    public BigDecimal solve(BigDecimal temp) {
        return BigDecimal.valueOf(new Expression("F(x)", this.f,
                new Argument("x", temp.doubleValue())).calculate());

    }
}
```

TemperatureUnit.java

```java
package me.henryfbp.temperatureconverter.lib;

public class TemperatureUnit {

    public String unit;

    public TemperatureUnit(String unit) {
        this.unit = unit;
    }
}
```