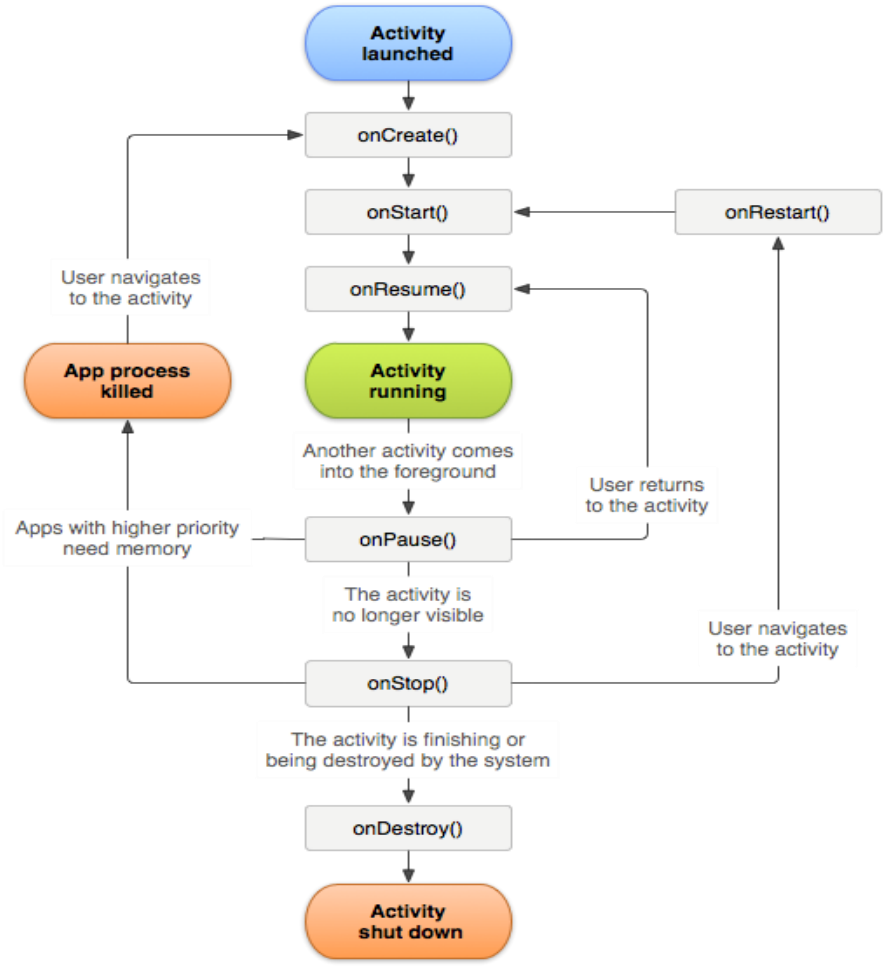
Android – Szkolenie Podstawowe

Future Processing
FP Mobile Division
***„Projekt Jaszczur”* 2015**

Android – Szkolenie Podstawowe

Zad 1 - Utworzenie nowego projektu

Cykl życia aktywności



onCreate() – wywoływana gdy aktywność została utworzona po raz pierwszy. To jest mniejsce gdzie należy wykonywać podstawowe czynności, jak tworzenie widoków, bindowanie danych itp.

onRestart() – wywoływana gdy aktywność została zatrzymana i uruchomiona ponownie

onStart() – wywoływana, gdy aktywność staje się widoczna dla użytkownika.

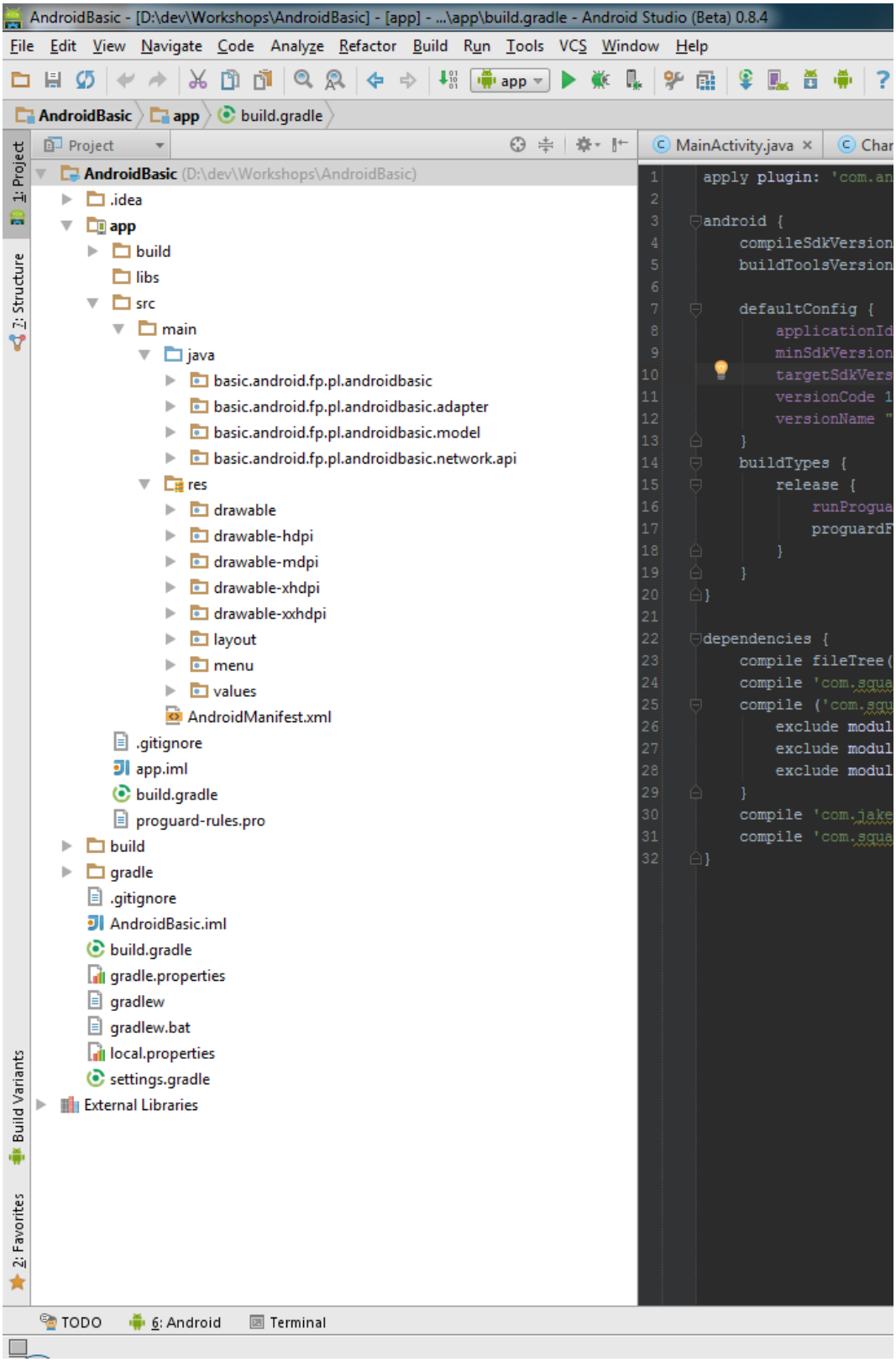
onResume() – wywoływana gdy aktywność rozpoczyna interakcję z użytkownikiem. W tym omencie aktywność jest na szczycie sotsu aktywności.

onPause() - wywoływana, gdy system rozpoczyna przywracanie poprzedniej aktywności. Jest zazwyczaj używana do zapisania trwałych danych, zatrzymania animacji i innych rzeczy, które mogą obciążać procesor.

onStop() – wywoływana gdy aktywność przestaje być widoczna dla użytkownik. Dzieje się tak zazwyczaj gdy otwierana jest nowa aktywność lub aktualna zostaje zniszczona.

onDestroy() – Ostatnia metoda wywoływana przed zniszczeniem aktywności.

Struktura projektu



Gradle

Plik gradle.build w module:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "basic.android.fp.pl.androidbasic"
        minSdkVersion 15
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
              package="basic.android.fp.pl.androidbasic">

      <uses-permission android:name="android.permission.INTERNET" />

      <application
            android:allowBackup="true"
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme">
            <activity
                  android:name=".MainActivity"
                  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=".ListCurrenciesActivity" />
      </application>

</manifest>
```

Zad 2 - Modyfikujemy pierwszą aktywność

Pamiętaj: Każda aktywność musi być zadeklarowana w manifeście!!

MainActivity:

```
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends Activity {

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		Button changeCurrencyButton = (Button) findViewById(R.id.listCurrenciesButton);
		Button changeCurrencyDialogButton = (Button) findViewById(R.id.rateChangeButton);
	}
}
```

Activity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
			android:layout_width="match_parent"
			android:layout_height="match_parent"
			android:orientation="vertical"
			android:gravity="center">

	<Button
		android:id="@+id/listCurrenciesButton"
		android:layout_width="wrap_content"
		android:layout_height="wrap_content"
		android:text="@string/open_exchange_rates_list" />

	<Button
		android:id="@+id/rateChangeButton"
		android:layout_width="wrap_content"
		android:layout_height="wrap_content"
		android:text="@string/change_rate_manually" />

</LinearLayout>
```

Do pliku strings.xml należy dodać dwa teksty:

```
<string name="change_rate_manually">Zmien kurs ręcznie</string>
<string name="open_exchange_rates_list">Otwórz listę kursów</string>
```

Zad 3 - Odkrywamy magię cyklu życia

```
public class MyActivity extends Activity {


	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		Log.i("LIFECYCLE", "-OnCreate");
	}

	@Override
	protected void onStart() {
		super.onStart();
		Log.i("LIFECYCLE", "--OnStart");
	}

	@Override
	protected void onResume() {
		super.onResume();
		Log.i("LIFECYCLE", "---OnResume");
	}

	@Override
	protected void onPause() {
		super.onPause();
		Log.i("LIFECYCLE", "---OnPause");
	}

	@Override
	protected void onStop() {
		super.onStop();
		Log.i("LIFECYCLE", "--OnStop");
	}

	@Override
	protected void onDestroy() {
		super.onDestroy();
		Log.i("LIFECYCLE", "-OnDestroy");
	}

	@Override
	protected void onRestart() {
		super.onRestart();
		Log.i("LIFECYCLE", "=====>OnRestart");
	}
}
```

Zad 4 - Tworzymy drugą aktywność z listą

Activity_change_currency.xml:

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
```

ListCurrenciesActivity:

```
public class ListCurrenciesActivity extends Activity {

        @InjectView(R.id.list)
        protected ListView currencyListView;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_change_currency);
                ButterKnife.inject(this);
        }
}
```

Dodajemy do build.gradle w tagu dependencies:

```
compile 'com.jakewharton:butterknife:+'
```

Dodajemy do MainActivity otwieranie nowej aktywności po naciśnięciu buttona.
W metodzie onCreate() dodajemy:

```
button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
                Intent i = new Intent(getApplicationContext(), ListCurrenciesActivity.class);
                startActivity(i);
        }
});
```

Tworzymy layout dla pojedyńczego elementu listy:
item_currency_list.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal">

   <ImageView
      android:id="@+id/flag"
      android:layout_width="36dp"
      android:layout_height="36dp"/>

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">

      <TextView
         android:id="@+id/currencyName"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:paddingLeft="6dp"
         android:text="Name"
         android:textAppearance="?android:attr/textAppearanceMedium"/>

      <TextView
         android:id="@+id/averageRate"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:paddingLeft="6dp"
         android:text="1 EUR 4 PLN"
         android:textAppearance="?android:attr/textAppearanceSmall"/>
   </LinearLayout>

</LinearLayout>
```

JSON otrzymywany z serwisu:

```
{
"date" : "2015-03-06",
 "base" : "PLN",
 "rates" : [
        {"currency" : "USD", "rate" : 0.2662},
        {"currency" : "IDR", "rate" : 3444.17}
        ]
}
```

Na podstawie json’a tworzymy klasy modelu:

RatesList**:**

```
public class RatesList {

        private String date;
        private Currency base;
        private List<ExchangeRate> rates;

        public RatesList(String date, Currency base, List<ExchangeRate> exchangeRates) {
                this.date = date;
                this.base = base;
                this.rates = exchangeRates;
        }

        public String getDate() {
                return date;
        }

        public void setDate(String date) {
                this.date = date;
        }

        public Currency getBase() {
                return base;
        }

        public void setBase(Currency base) {
                this.base = base;
        }

        public List<ExchangeRate> getExchangeRates() {
                return rates;
        }

        public void setExchangeRates(List<ExchangeRate> exchangeRates) {
                this.rates = exchangeRates;
        }
}
```

ExchangeRate:

```
public class ExchangeRate implements Serializable {

	private final Currency currency;
	private Float rate;

	public ExchangeRate(Currency currency, Float rate) {
		this.currency = currency;
		this.rate = rate;
	}

	public Currency getCurrency() {
		return currency;
	}

	public Float getRate() {
		return rate;
	}

	public void setRate(Float rate) {
		this.rate = rate;
	}
}
```

Tworzymy adapter dla listy:
CurrencyListAdapter:

```
public class CurrencyListAdapter extends BaseAdapter {

	private final Context context;
	private final List<ExchangeRate> exchangeRates;
	private final LayoutInflater inflater;

	public CurrencyListAdapter(Context context, RatesList currencyTable) {
		this.context = context;
		this.currencyTable = currencyTable;
	inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
	}

	@Override
	public int getCount() {
		return exchangeRates.size();
	}

	@Override
	public ExchangeRate getItem(int position) {
		return exchangeRates.get(position);
	}

	@Override
	public long getItemId(int position) {
		return position;
	}

	@Override
	public View getView(int position, View convertView, ViewGroup parent) {
		return convertView;
	}
}
```

Dodajemy klasę wewnętrzną.
ViewHolder:

```
protected class ViewHolder {
        @InjectView(R.id.currencyName)
        TextView currencyName;
        @InjectView(R.id.averageRate)
        TextView averageRate;

        private ViewHolder(View rootView) {
                ButterKnife.inject(this, rootView);
        }

        protected void populate(ExchangeRate exchangeRate) {
                currencyName.setText(exchangeRate.getCurrency().getCountry() + " " +
                exchangeRate.getCurrency().getCurrencyName());
                averageRate.setText(exchangeRate.getRate().toString());
        }
}
```

Definiujemy jak wypełniany ma być element listy.
W metodzie getView() dodajemy:

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder vh;
        if (convertView == null) {
                convertView = inflater.inflate(R.layout.item_currency_list, parent, false);
                vh = new ViewHolder(convertView);
                convertView.setTag(vh);
        } else {
                vh = (ViewHolder) convertView.getTag();
        }

        ExchangeRate exchangeRate = getItem(position);
        vh.populate(exchangeRate);

        return convertView;
}
```

Ważne, by przy inflatowaniu podać jako parametr attacheToRoot „false”(pogrubione w powyższym listingu), ponieważ adapter pod spodem robi to za nas, więc podpięcie samodzielnie spowalnia cały proces.

Tworzymy dane testowe i dodajemy adapter do listy w ListCurrenciesActivity w metodzie onCreate():

MockData:

```
public class MockData {

	static RatesList ratesList;

	public static RatesList getListOfRates() {
		if (ratesList == null) {
			List<ExchangeRate> rates = new ArrayList<ExchangeRate>();
			rates.add(new ExchangeRate(Currency.AUD, 0.3431f));
			rates.add(new ExchangeRate(Currency.BGN, 0.4724f));
			rates.add(new ExchangeRate(Currency.BRL, 0.7974f));
			rates.add(new ExchangeRate(Currency.CAD, 0.3326f));
			rates.add(new ExchangeRate(Currency.CHF, 0.2584f));
			rates.add(new ExchangeRate(Currency.CNY, 1.676f));
			rates.add(new ExchangeRate(Currency.CZK, 6.6242f));
			rates.add(new ExchangeRate(Currency.DKK, 1.8007f));
			rates.add(new ExchangeRate(Currency.GBP, 0.1752f));
			rates.add(new ExchangeRate(Currency.HKD, 2.0737f));
			rates.add(new ExchangeRate(Currency.HRK, 1.85f));
			rates.add(new ExchangeRate(Currency.HUF, 73.764f));
			rates.add(new ExchangeRate(Currency.IDR, 3469.59f));
			rates.add(new ExchangeRate(Currency.ILS, 1.0694f));
			rates.add(new ExchangeRate(Currency.INR, 16.646f));
			rates.add(new ExchangeRate(Currency.JPY, 32.152f));
			rates.add(new ExchangeRate(Currency.KRW, 294.43f));
			rates.add(new ExchangeRate(Currency.MXN, 4.0236f));
			rates.add(new ExchangeRate(Currency.MYR, 0.9763f));
			rates.add(new ExchangeRate(Currency.NOK, 2.0644f));
			rates.add(new ExchangeRate(Currency.NZD, 0.357f));
			rates.add(new ExchangeRate(Currency.PHP, 11.8f));
			rates.add(new ExchangeRate(Currency.RON, 1.0738f));
			rates.add(new ExchangeRate(Currency.RUB, 16.332f));
			rates.add(new ExchangeRate(Currency.SEK, 2.2258f));
			rates.add(new ExchangeRate(Currency.SGD, 0.3661f));
			rates.add(new ExchangeRate(Currency.THB, 8.6685f));
			rates.add(new ExchangeRate(Currency.TRY, 0.6924f));
			rates.add(new ExchangeRate(Currency.USD, 0.2674f));
			rates.add(new ExchangeRate(Currency.ZAR, 3.1443f));
			rates.add(new ExchangeRate(Currency.EUR, 0.2416f));
			ratesList = new RatesList("2015-03-07", Currency.PLN, rates);
		}
		return ratesList;
	}
```

ListCurrenciesActivity -> onCreate():

```
CurrencyListAdapter adapter = new CurrencyListAdapter(this, MockData. getListOfRates());
currencyListView.setAdapter(adapter);
```

Zad 5 - Zapisanie wybranej waluty do pamięci trwałej

Dodajemy klasę pomocniczą do zapisu i odczytu z pamięci trwałej.

```
public class SharedPreferencesSupporter {

	private static final String CURRENCY_MAIN_KEY = SharedPreferencesSupporter.class.getName() +
".currency";
	private static final String NAME = ".name";
	private static final String AVERAGE_RATE = ".averageRate";

	public static ExchangeRate loadCurrentRate(Context context) {
		SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);
		float averageRate = preferences.getFloat(CURRENCY_MAIN_KEY + AVERAGE_RATE,
3.73f);
		String name = preferences.getString(CURRENCY_MAIN_KEY + NAME,
Currency.PLN.toString());
		return new ExchangeRate(Currency.valueOf(name), averageRate);
	}

	public static void saveCurrentRate(ExchangeRate exchangeRate, Context context) {
		SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);
		SharedPreferences.Editor editor = preferences.edit();
		editor.putFloat(CURRENCY_MAIN_KEY + AVERAGE_RATE, exchangeRate.getRate());
		editor.putString(CURRENCY_MAIN_KEY + NAME, exchangeRate.getCurrency().toString());
		editor.apply();
	}
}
```

Następnie obsługujemy kliknięcie na elemencie listy.
W klasie ListCurrenciesActivity w metodzie onCreate dodajemy:

```
currencyListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
	@Override
	public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
		CurrencyListAdapter currencyAdapter = (CurrencyListAdapter) parent.getAdapter();
		ExchangeRate exchangeRate = currencyAdapter.getItem(position);
		SharedPreferencesSupporter.saveCurrentRate(exchangeRate, ListCurrenciesActivity.this);
		Toast.makeText(this, "Currency saved to SharedPreferences", Toast.LENGTH_SHORT).show();
	}
});
```

Lub używając Butterknifa:

```
@OnItemClick(R.id.list)
void onListItemClick(AdapterView<?> parent, View view, int position){
	CurrencyListAdapter currencyAdapter = (CurrencyListAdapter) parent.getAdapter();
	ExchangeRate exchangeRate = currencyAdapter.getItem(position);
	SharedPreferencesSupporter.saveCurrentRate(exchangeRate, ListCurrenciesActivity.this);
	Toast.makeText(this, "Currency saved to SharedPreferences", Toast.LENGTH_SHORT).show();
}
```

Zad 6 - Odczyt z pamięci

By zobaczyć rezultat zapisu w pamięci dodamy pole tekstowe w MainActivity z aktualnie wybraną walutą. Edytujemy activity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:orientation="vertical"
                        android:gravity="center">

        <TextView
                android:id="@+id/currentCurrency"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20sp"
                android:textStyle="bold" />

        <Button
                android:id="@+id/listCurrenciesButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/open_exchange_rates_list" />

        <Button
                android:id="@+id/rateChangeButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/change_rate_manually" />

</LinearLayout>
```

Oraz obsługujemy pobieranie z pamięci w MainActivity:

W metodzie onCreate:

```
currentCurrency = (TextView) findViewById(R.id.currentCurrency);
```

Dodajemy metodę:

```
@Override
protected void onResume() {
        super.onResume();
        currentExchangeRate = SharedPreferencesSupporter.loadCurrentRate(this);
        final Currency currency = currentExchangeRate.getCurrency();
        currentCurrency.setText("Twoja waluta to: " + currency.getCountry() + " " +
currency.getCurrencyName() + "\nKurs: " + currentExchangeRate.getRate());
}
```

Zad 7 - Asynchroniczne ładowanie bitmap

Do pliku build.gradle w tagu dependecsies dodajemy:

compile 'com.squareup.picasso:picasso:+'

Następnie dodajemy klasę pomocniczą dostarczającą adresy poszczególnych flag:

```
public class FlagAddress {

	public static String obtainAddress(Currency currency) {
		return "http://192.168.0.11:8087/" + currency.getCountry().toLowerCase().replace(" ",
"") + ".png";
	}
}
```

Uaktualniamy ViewHolder

```
protected class ViewHolder {
	@InjectView(R.id.currencyName)
	TextView currencyName;
	@InjectView(R.id.averageRate)
	TextView averageRate;
	@InjectView(R.id.flag)
	ImageView flag;

	private ViewHolder(View rootView) {
		ButterKnife.inject(this, rootView);
	}
}
```

W metodzie populate dodajemy pobranie obrazków za pomocą Picasso:

Picasso.with(context).load(FlagAddress.obtainAddress(exchangeRate.getCurrency())).placeholder(R.drawable.money).into(flag);

Zad 8 - Pobieranie walut z internetu i parsowanie json’a

Tworzymy API które definiuje punkty dostępu do serwisu.

```
public interface JsonRatesService {

	@GET("/list/USD")
	RatesList getCurrencyTable();
}
```

W ListCurrenciesActivity w metodzie onCreate() konfigurujemy adapter serwisu.

```
Gson gson = new GsonBuilder().
	setFieldNamingPolicy(FieldNamingPolicy.LOWER_CASE_WITH_UNDERSCORES).
	registerTypeAdapter(Currency.class, new CurrencyTypeAdapter()).
	create();

RestAdapter restAdapter = new RestAdapter.Builder().
	setEndpoint(getString(R.string.webservice_url)).
	setConverter(new GsonConverter(gson)).
	build();

service = restAdapter.create(JsonRatesService.class);
```

Używamy biblioteki GSON do parsowania JSON.
Dodajemy klasę adaptera dla GSON’a wspomagającą parsowanie:

```
public class CurrencyTypeAdapter extends TypeAdapter<Currency> {
	@Override
	public void write(JsonWriter out, Currency value) throws IOException {
		if (value == null) {
			out.nullValue();
			return;
		}
		out.value(value.toString());
	}

	@Override
	public Currency read(JsonReader in) throws IOException {
		if (in.peek() == JsonToken.NULL) {
			in.nextNull();
			return null;
		}
		return Currency.valueOf(in.nextString());
	}
}
```

Pobieranie z internet wywoływane musi być na osobnym wątku, inaczej rzucony zostanie NetworkOnMainThreadException. Dlatego oddelegujemy wywołanie operacji sieciowej do osobnego wątku. Zdefiniujemy wewnętrzną klasę rozszerzającą AsyncTask.

```
private class GetCurrencyTableTask extends AsyncTask<Currency, Void, RatesList> {

	private final ProgressDialog dialog;

	public GetCurrencyTableTask(Context context) {
		dialog = new ProgressDialog(context);
		dialog.setMessage(getString(R.string.please_wait));
	}

	@Override
	protected void onPreExecute() {
		super.onPreExecute();
		dialog.show();
	}

	@Override
	protected RatesList doInBackground(Currency... params) {
		return service.getCurrencyTable(params[0]);
	}

	@Override
	protected void onPostExecute(RatesList currencies) {
		super.onPostExecute(currencies);
		dialog.dismiss();
		currencyListView.setAdapter(new CurrencyListAdapter(ListCurrenciesActivity.this,
currencies));
	}
}
```

Dodajemy metodę:

```
private void loadData() {
	new GetCurrencyTableTask(this).execute();
}
```

Pozostaje w metodzie onCreate wywołać metodę loadData:

Zad 9 - Dodawanie menu kontekstowego oraz akcji do ActionBar’a

By zdefiniować menu kontekstowe oraz akcje w ActionBarze, należy utworzyć nowy plik w folderze menu.
Change_currency.xml:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item
      android:id="@+id/action_refresh"
      android:title="@string/action_refresh"
      android:showAsAction="always"
      android:icon="@drawable/ic_menu_refresh"/>
   <item
      android:id="@+id/menu_refresh"
      android:title="@string/action_refresh"
      android:showAsAction="never"
      android:icon="@drawable/ic_menu_refresh"/>
</menu>
```

Wracamy do ChangeCurrencyActivity i dodajemy metody:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.change_currency, menu);
        return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_refresh || id == R.id.menu_refresh) {
                loadData();
                return true;
        }
        return super.onOptionsItemSelected(item);
}
```

Zad 10 - Dialog do edycji aktualnej waluty

Dodajemy klasę definiującą dialog do edycji waluty:

```
public class RateChangeDialogFragment extends DialogFragment {

	private static final String CURRENCY_BUNDLE_KEY = "CURRENCY_BUNDLE_KEY";

	private OnCurrencyChangedListener onCurrencyChangedListener;
	private ExchangeRate currencyRate;

	private EditText inputEditText;

	public static RateChangeDialogFragment getInstance(ExchangeRate rate) {
		Bundle bundle = new Bundle();
		bundle.putSerializable(CURRENCY_BUNDLE_KEY, rate);

		RateChangeDialogFragment fragment = new RateChangeDialogFragment();
		fragment.setArguments(bundle);

		return fragment;
	}

	@Override
	public void onAttach(Activity activity) {
		super.onAttach(activity);

		// This makes sure that the container activity has implemented
		// the callback interface. If not, it throws an exception
		try {
			onCurrencyChangedListener = (OnCurrencyChangedListener) activity;
		} catch (ClassCastException e) {
			throw new ClassCastException(activity.toString() + " must implement
OnCurrencyChangedListener");
		}

		currencyRate = (ExchangeRate)
getArguments().getSerializable(CURRENCY_BUNDLE_KEY);
	}

	@Override
	public Dialog onCreateDialog(Bundle savedInstanceState) {
		inputEditText = createInputEditText();

		return new AlertDialog.Builder(getActivity()) //
				.setIcon(R.drawable.ic_launcher) //
```

```
                .setTitle(R.string.dialog_title) //
                .setMessage(R.string.dialog_message) //
                .setPositiveButton(R.string.ok, new PositiveOnClickListener()) //
                .setNegativeButton(R.string.cancel, null) //
                .setView(inputEditText) //
                .create();
    }

    private EditText createInputEditText() {
        EditText input = new EditText(getActivity());
        input.setInputType(InputType.TYPE_CLASS_NUMBER |
InputType.TYPE_NUMBER_FLAG_DECIMAL);
        input.addTextChangedListener(new CurrencyTextWatcher());
        input.setText(String.valueOf(currencyRate.getRate()));
        return input;
    }

    private boolean isValid(String text) {
        try {
            Float.parseFloat(text);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }

    private class PositiveOnClickListener implements DialogInterface.OnClickListener {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            changeExchangeRate();
        }
    }

    private void changeExchangeRate() {
        if (isValid(inputEditText.getText().toString())) {
            onCurrencyChangedListener.onRateChanged(currencyRate);
        } else {
            Toast.makeText(getActivity(), R.string.invalid, Toast.LENGTH_SHORT).show();
        }
    }

    private class CurrencyTextWatcher implements TextWatcher {

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

```
                @Override
                public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                }

                @Override
                public void afterTextChanged(Editable s) {
                        if (isValid(s.toString())) {
                                currencyRate.setRate(Float.parseFloat(s.toString()));
                        }
                }
        }

        public interface OnCurrencyChangedListener {

                void onRateChanged(ExchangeRate currency);
        }
}
```

W MainActivity dodajemy obsługę klikania na drugi button:

```
changeCurrencyDialogButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        RateChangeDialogFragment.getInstance(currentExchangeRate).show(getFragmentManager(),
"tag");
        }
});
```

Oraz implementacje interfejsu OnCurrencychangedListener

```
@Override
public void onRateChanged(ExchangeRate exchangeRate) {
        currentExchangeRate = exchangeRate;
        final Currency currency = exchangeRate.getCurrency();
        currentCurrency.setText("Twoja waluta to: " + currency.getCountry() + " " +
currency.getCurrencyName() + "\nKurs: " + exchangeRate.getRate());
        SharedPreferencesSupporter.saveCurrentRate(exchangeRate, this);
}
```

Zad 11 - Przeliczanie walut

Tworzymy nowe activity:

```
public class ExchangeActivity extends Activity {
```

```
	public static final String CURRENCY_BUNDLE_KEY = "CURRENCY_BUNDLE_KEY";

	@InjectView(R.id.newCurrency)
	protected TextView newCurrencyTextView;
	@InjectView(R.id.currency)
	protected TextView currencyTextView;

	private ExchangeRate exchangeRate;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_exchange);
		ButterKnife.inject(this);

		exchangeRate = (ExchangeRate)
getIntent().getSerializableExtra(CURRENCY_BUNDLE_KEY);
		currencyTextView.setText("Aktualny kurs to:\t" + exchangeRate.getRate());
	}

	@OnTextChanged(value = R.id.currencyEditText, callback =
	OnTextChanged.Callback.AFTER_TEXT_CHANGED)
	protected void onTextChange(Editable text) {
		if (isValid(text.toString())) {
			float value = Float.parseFloat(text.toString()) * exchangeRate.getRate();
			newCurrencyTextView.setText("To\t" + value + "\t" +
exchangeRate.getCurrency());
		} else {
			newCurrencyTextView.setText(R.string.invalid);
		}
	}

	private boolean isValid(String text) {
		try {
			Float.parseFloat(text);
			return true;
		} catch (NumberFormatException e) {
			return false;
		}
	}
}
```

Dodajemy layout:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical"
```

```
                android:layout_margin="30dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="USD:"
        android:textSize="12sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/currencyEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number|numberDecimal" />

    <TextView
        android:id="@+id/newCurrency"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp" />

    <TextView
        android:id="@+id/currency"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp" />

</LinearLayout>
```

Edytujemy metodę onCreate() w MainActivity – dodajemy:

```
Button calculateCurrencyButton = (Button) findViewById(R.id.calculateCurrency);
calculateCurrencyButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                        Intent i = new Intent(getApplicationContext(), ExchangeActivity.class);
                        i.putExtra(ExchangeActivity.CURRENCY_BUNDLE_KEY,
currentExchangeRate);
                        startActivity(i);
                }
        });
```

Przydatne linki

- http://square.github.io/retrofit/
- http://jakewharton.github.io/butterknife/
- http://square.github.io/picasso/

- http://gradleplease.appspot.com/
- http://developer.android.com/index.html