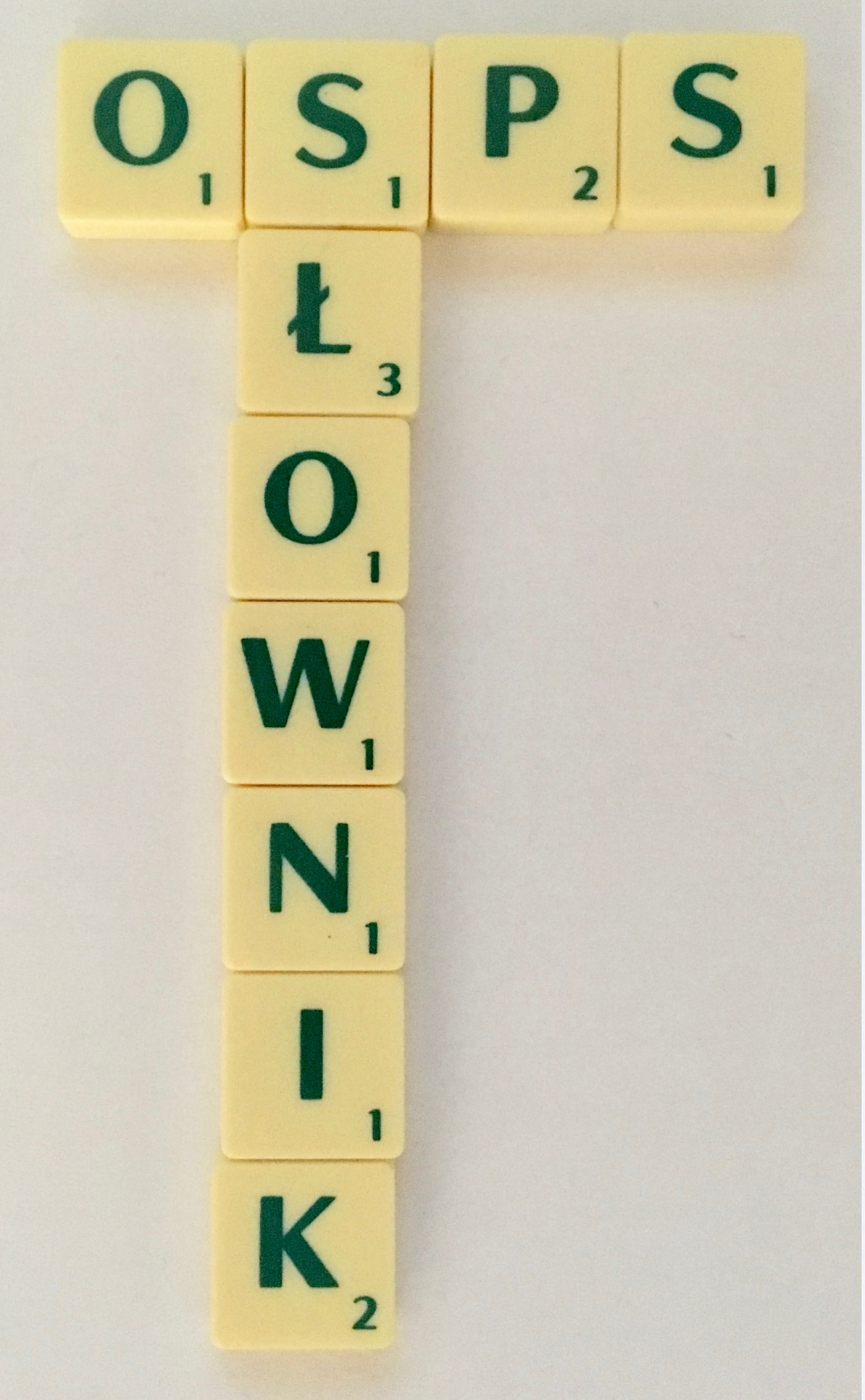
WSTĘP DO ANDROIDA

Laboratorium 6
Systemy i aplikacje bez granic

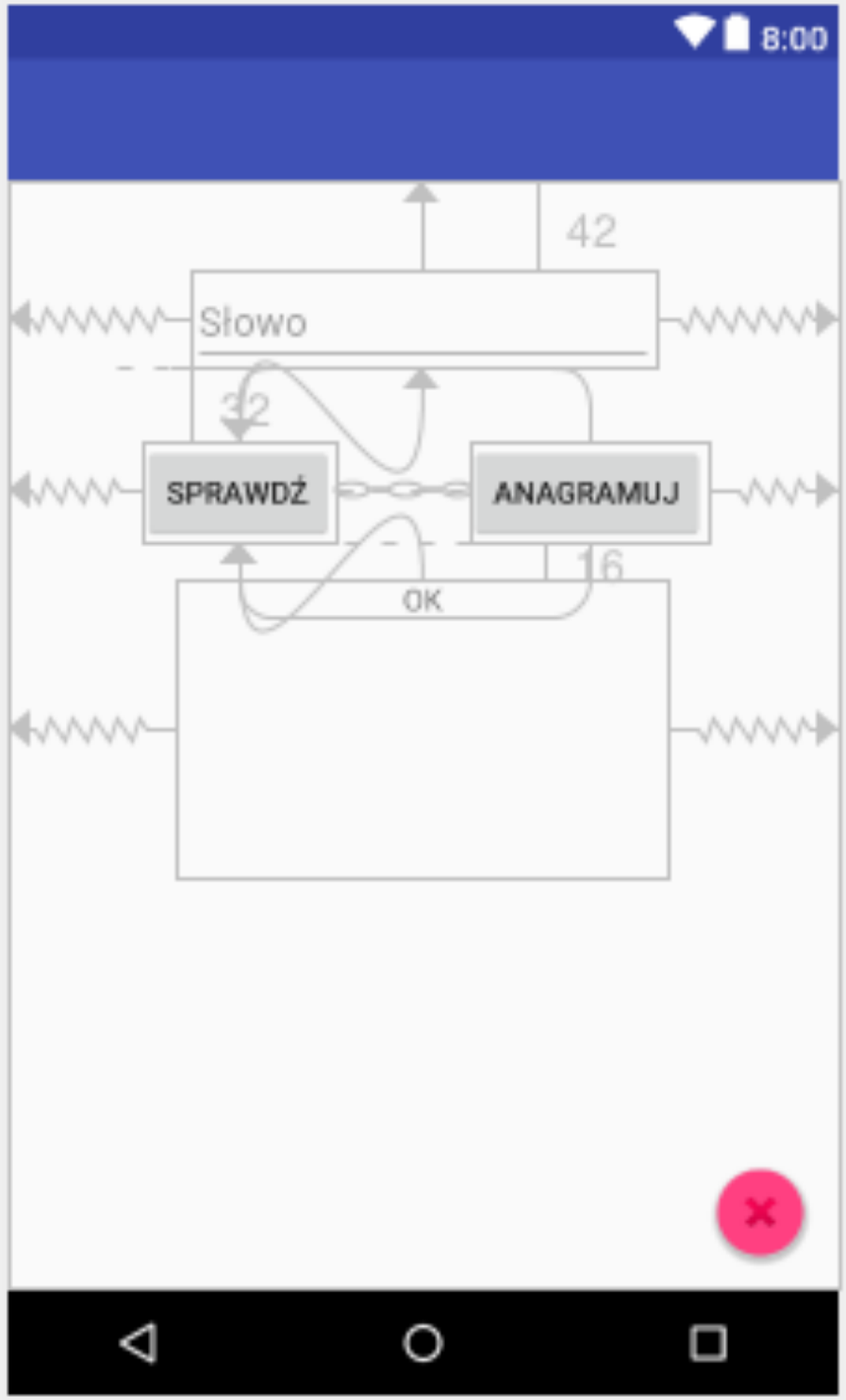


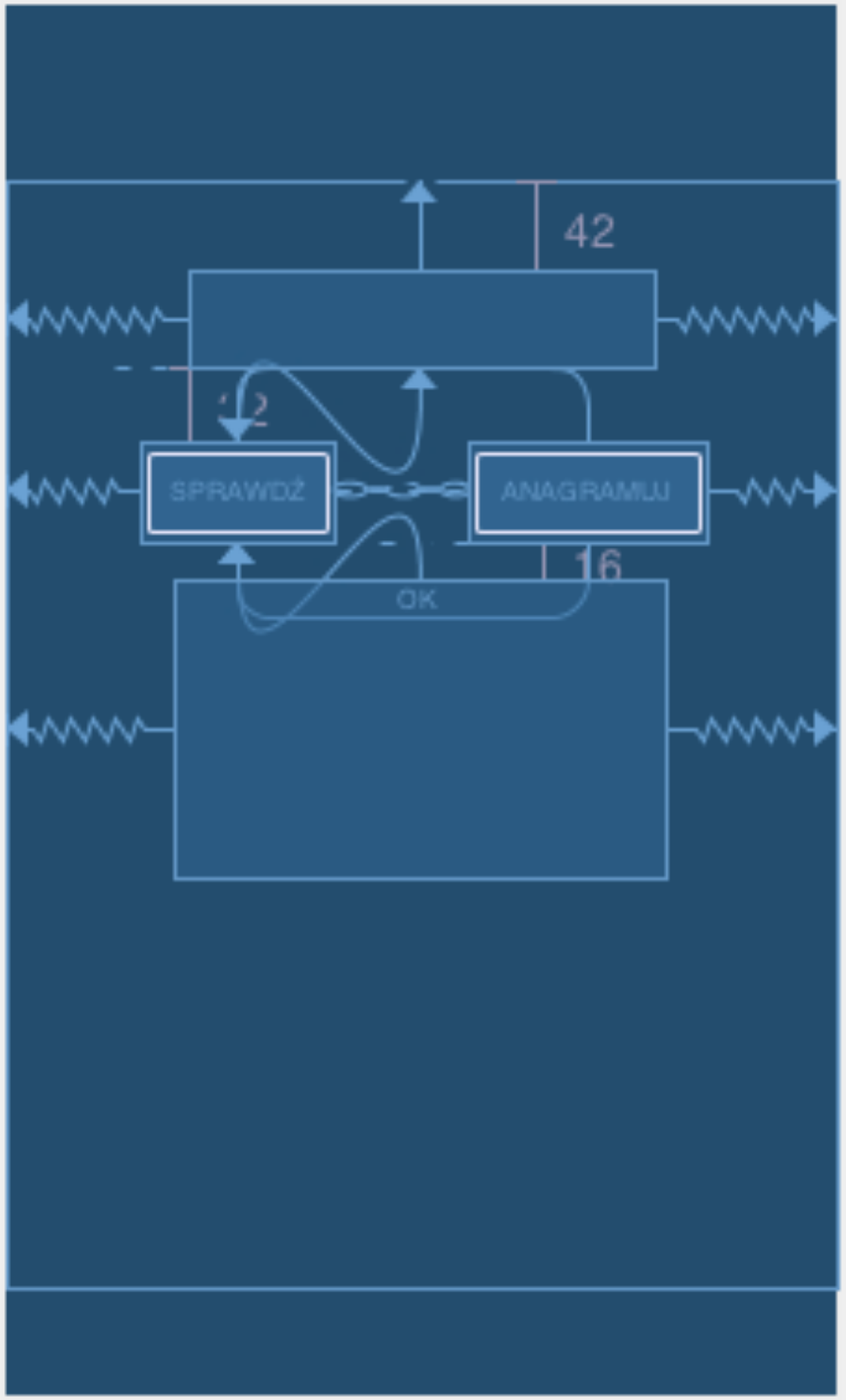
Stworzymy aplikację - interfejs do Oficjalnego Słownika Polskiego Scrabblisty

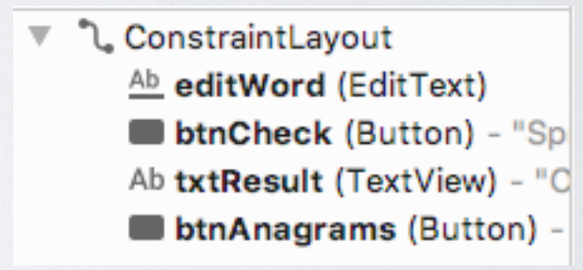
I

- Tworzymy nowy projekt typu Empty Activity o nazwie OSPS - góra Android 8

- Projektujemy aktywność aby wyglądała jak na następnym slajdzie - zawiera Plain Text, 2 przyciski i wieloliniowy TextView


8:00
42
Słowo
32
SPRAWDŹ
ANAGRAMUJ
16
OK


42
SPRAWDŹ
ANAGRAMUJ
16
OK


ConstraintLayout
editWord (EditText)
btnCheck (Button) - "Sp
txtResult (TextView) - "C
btnAnagrams (Button) -

- Do pliku build.gradle (Module: app) dodajemy na koniec linijkę

```
implementation 'khttp:khttp:0.1.0'
```

- W aktywności dodajemy metody obsługi przycisków

```
fun checkWord(v: View) {
    val tast=BgTask()
    tast.execute( ...params: "Check")
}

fun doAnagrams(v: View) {
    val tast=BgTask()
    tast.execute( ...params: "Anagrams")
}
```

- W klasie aktywności definiujemy pole

```
private var word:String? = null
```

- Ponieważ transmisja sieciowa musi odbywać się w wątku asychronicznym, musimy zdefiniować klasę dziedziczącą po AsyncTask

- Wewnątrz klasy aktywności dodajemy deklarację

```
private inner class BgTask:AsyncTask<String,Int,String> () {

    override fun onPreExecute() {
        super.onPreExecute()
        word = editWord.text.toString().trim()
    }

    override fun onPostExecute(result: String?) {
        super.onPostExecute(result)
        txtResult.text=result
    }
```

```
override fun doInBackground(vararg p0: String?): String {
    if (word!=null) {
        val w=word!!
        if (w.length > 1) {

            if (p0[0]=="Check") {

                try {
                    val url = "http://www.pfs.org.pl/files/php/osps_funkcje2.php"

                    val response = khttp.get(
                        url = url,
                        params = mapOf("s" to "spr", "slowo_arbiter2" to w))

                    val r=response.content[0].toInt()
                    if (r==49)
                        return "Słowo jest poprawne"
                    else
                        return "Słowo jest niepoprawne"

                } catch (e: Exception) {
                    return e.message.toString()
                }
            } else { //anagramy
```

```
        try {
            val url = "http://www.pfs.org.pl/files/php/osps_funkcje2.php"

            val response = khttp.get(
                    url = url,
                    params = mapOf("s" to "ana", "slowo2" to w))

            val r=String(response.content)
            val items=r.split( ...delimiters: "|")
            val sb=StringBuilder()

            for (i in items) if (i!="") {
                sb.appendln(i)
                }
                return sb.toString()

            } catch (e: Exception) {
                return e.message.toString()
            }
        }
    } else {
        return "Wprowadź słowo mające 2 lub więcej liter"
    }
}
return "Wprowadź słowo mające 2 lub więcej liter"
}
```

II

- Tworzymy aplikację wyświetlającą bieżące kursy walut z kilku źródeł:
  - umowny kurs średni Forex
  - Kurs w kantorze internetowy Alior Banku
  - Kurs dewiz w banku BZWBK

II

- Za udostępnienie kursów odpowiedzialna jest aplikacja-serwis, która parsuje odpowiednie źródła i agreguje je do jednego pliku w formacie RSS

- Nasza aplikacja ma parsować ten plik i wyświetlać jego zawartość na ekranie jako dynamicznie tworzoną tabelę

II

- Tworzymy nowy projekt typu Empty Activity o nazwie Waluty

- Do manifestu dodajemy na wszelki wypadek linijkę

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

II

- Projektujemy wyg

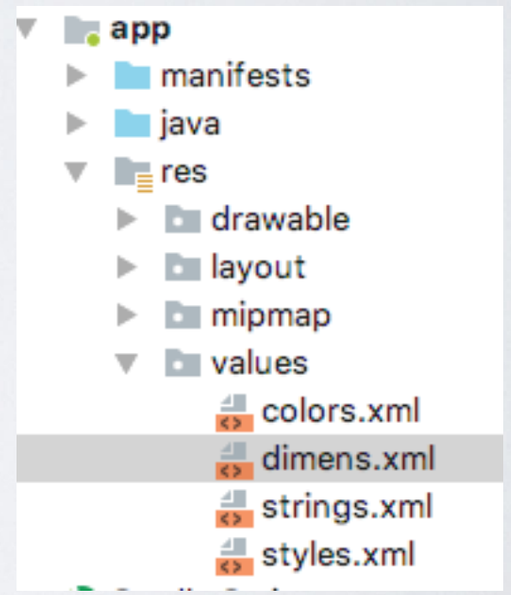
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

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

        <Button
            android:id="@+id/btnRefresh"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="refresh"
            android:text="Odśwież" />
    </LinearLayout>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TableLayout
            android:id="@+id/tableCurrencies"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

        </TableLayout>
    </ScrollView>

</LinearLayout>
```

```
class Currency(var code:String,var rate:Double,var change:Double,var date:Date) {

    fun formatChange():String {
        val ch=rate-change
        val sb=StringBuilder()
        if (ch>0)
            sb.append("+")
        sb.append(String.format("%.4f", ch))
        return sb.toString()
    }

    fun percentChange():String {
        val ch=rate-change
        var p=(ch/change)*100
        val sb=StringBuilder()
        if (ch>0)
            sb.append("+")
        sb.append(String.format("%.2f",p))
        sb.append("%")
        return sb.toString()
    }

    fun getColor():String {
        val ch=rate-change
        if (ch>0) return "#00FF00"
        if (ch<0) return "#FF0000"
        return "#aaaaaa"
    }
}
```

II

- W projekcie na ścieżce app/res/values dodamy nowy plik XML o nazwie dimens.xml i poniższej zawartości

```
<resources>
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="font_size_verysmall">12sp</dimen>
    <dimen name="font_size_small">14sp</dimen>
    <dimen name="font_size_medium">24sp</dimen>
</resources>
```



II

- W klasie aktywności dodajemy jako pola 3 listy, które będą przechowywały kursy walut

```
var forex:MutableList<Currency>? = null
var alior:MutableList<Currency>? = null
var bzwbk:MutableList<Currency>? = null
```

- Ściąganie pliku musi odbywać się asynchronicznie, dodajemy więc klasę wewnętrzną

||

```
private inner class CurrencyDownloader: AsyncTask<String, Int, String>() {

    override fun onPreExecute() {
        super.onPreExecute()
    }

    override fun onPostExecute(result: String?) {
        super.onPostExecute(result)
        loadData()
        showData()
    }
```

```kotlin
override fun doInBackground(vararg params: String?): String {
    try {
        val url = URL( spec: "http://fcds.cs.put.poznan.pl/MyWeb/waluty.xml")
        val connection = url.openConnection()
        connection.connect()
        val lenghtOfFile = connection.contentLength
        val isStream = url.openStream()
        val testDirectory = File( pathname: "$filesDir/XML")
        if (!testDirectory.exists()) testDirectory.mkdir()
        val fos = FileOutputStream( name: "$testDirectory/waluty.xml")
        val data = ByteArray( size: 1024)
        var count = 0
        var total: Long = 0
        var progress = 0
        count = isStream.read(data)
        while (count != -1) {
            total += count.toLong()
            val progress_temp = total.toInt() * 100 / lenghtOfFile
            if (progress_temp % 10 == 0 && progress != progress_temp) {
                progress = progress_temp
            }
            fos.write(data, off: 0, count)
            count = isStream.read(data)
        }
        isStream.close()
        fos.close()
    } catch (e: MalformedURLException) {
        return "Malformed URL"
    } catch (e: FileNotFoundException) {
        return "File not found"
    } catch (e: IOException) {
        return "IO Exception"
    }
    return "success"
}
```

II

- Do aktywności dodajemy metody

```
fun downloadData() {
    val cd=CurrencyDownloader()
    cd.execute()
}

fun refresh(v: View) {
    downloadData()
}
```

```
fun loadData() {
    forex= mutableListOf()
    alior= mutableListOf()
    bzwbk= mutableListOf()

    val filename = "waluty.xml"
    val path = filesDir
    val inDir = File(path,  child: "XML")

    if (inDir.exists()) {
        val file = File(inDir, filename)
        if (file.exists()) {
            val xmlDoc: Document = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(file)

            xmlDoc.documentElement.normalize()

    val items: NodeList = xmlDoc.getElementsByTagName( tagname: "item")


    for (i in 0..items.length - 1) {
        val itemNode: Node = items.item(i)

        if (itemNode.getNodeType() == Node.ELEMENT_NODE) {

            val elem = itemNode as Element
            val children=elem.childNodes

            var currentCategory:MutableList<Currency>? = null
            var currentCode:String? = null
            var currentRate:String? = null
            var currentDate:String? = null
```

```
                for (j in 0..children.length - 1) {
                    val node=children.item(j)
                    if (node is Element) {
                        when (node.nodeName) {
                            "category" -> {
                                when (node.textContent) {
                                    "Forex" -> currentCategory = forex
                                    "Alior" -> currentCategory = alior
                                    "BZWBK" -> currentCategory = bzwbk
                                    else -> currentCategory = null
                                }
                            }
                            "title" -> { currentCode = node.textContent
                            }
                            "description" -> { currentRate = node.textContent
                            }
                            "pubDate" -> { currentDate = node.textContent
                            }
                        }
                    }
                }

                if (currentCategory!=null && currentCode!=null && currentDate!=null && currentRate!=null)
                {
                    val rates=currentRate.split( ...delimiters: " ")
                    val r=rates[0].toDouble()
                    val ch=rates[1].toDouble()

                    val pattern = "EEE, dd MMM yyyy HH:mm:ss Z"
                    //val format = SimpleDateFormat(pattern, Locale.ENGLISH)
                    val sdf=SimpleDateFormat(pattern, Locale.ENGLISH)
                    val d=sdf.parse(currentDate)


                    val c=Currency(currentCode,r,ch,d)
                    currentCategory.add(c)
                }
            }
        }
    }
}
```

II

- Oraz metodę showCurrencies, którą ze względu na długość można pobrać stąd: https://pastebin.com/2M0Yb919

- Proszę ją dobrze przeanalizować

II

- I jeszcze metodę

```
fun showData() {
    tableCurrencies.removeAllViews()
    if (forex!=null)
        showCurrencies(forex!!, name: "Forex")
    if (alior!=null)
        showCurrencies(alior!!, name: "Alior")
    if (bzwbk!=null)
        showCurrencies(bzwbk!!, name: "BZWBK")
}
```

- Do onCreate dodajemy jeszcze linijki

```
loadData()
showData()
downloadData()
```

II

- Właściwe importy

```
import android.graphics.Color
import android.os.AsyncTask
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.TypedValue
import android.view.Gravity
import android.view.View
import android.widget.*
import kotlinx.android.synthetic.main.activity_main.*
import org.w3c.dom.Document
import org.w3c.dom.Element
import org.w3c.dom.Node
import org.w3c.dom.NodeList
import java.io.File
import java.io.FileNotFoundException
import java.io.FileOutputStream
import java.io.IOException
import java.net.MalformedURLException
import java.net.URL
import java.text.SimpleDateFormat
import java.util.*
import javax.xml.parsers.DocumentBuilderFactory
```

- Oczekiwany efekt