Uniwersytet Technologiczno-Przyrodniczy im. J. i J. Śniadeckich w Bydgoszczy

Wydział Telekomunikacji, Informatyki i Elektrotechniki

Inżynieria Oprogramowania

Projekt – program walut

Autor:
Radosław Bożejewicz
Adrian Wiśniewski
Piotr Podlas
Radosław Szymański

Kierunek:
Teleinformatyka
Grupa: 1
Rok studiów: 3
Rok akademicki: 2017/2018
Tryb studiów: Stacjonarne

Założenia do projektu z „Inżynierii oprogramowania”

1. Oprogramowanie mające wyświetlać kursy walut Narodowego Banku Polskiego w aplikacji desktopowej.
2. Aplikacja desktopowa powinna pobierać dane ze strony Narodowego Banku Polskiego w formacie xml, a następnie interpretować je i wyświetlać w czytelny i przejrzysty sposób.
3. Użytkownik ma możliwość wyboru waluty oraz dnia (lub przedziału dni) z którego chce otrzymać dany kurs.

3.1. W przypadku przedziału dat wyświetlana jest cena dla każdego dnia, oraz cena średnia całego przedziału.

4. Dane wyświetlane powinny być w postaci tabeli, w której będzie zawarta nazwa waluty, cena, data, cena średnia w przypadku przedziału czasowego.

5. Wygenerowana tabela powinna mieć możliwość zapisania jej w formacie pdf.

6. Wygenerowana tabela powinna mieć możliwość wydrukowania jej.

Adnotacje podczas implementacji:

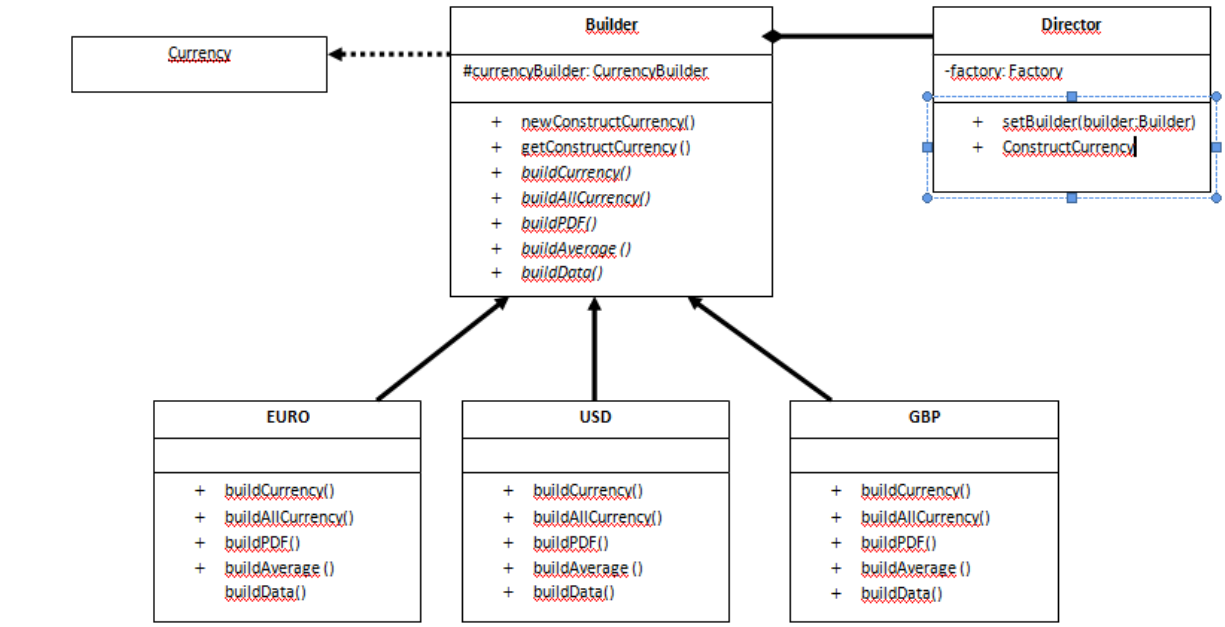
Dodano obsługę wyświetlania informacji o błędnie podanej dacie, o braku podanych informacji. Kompleksowa obsługa błędów.

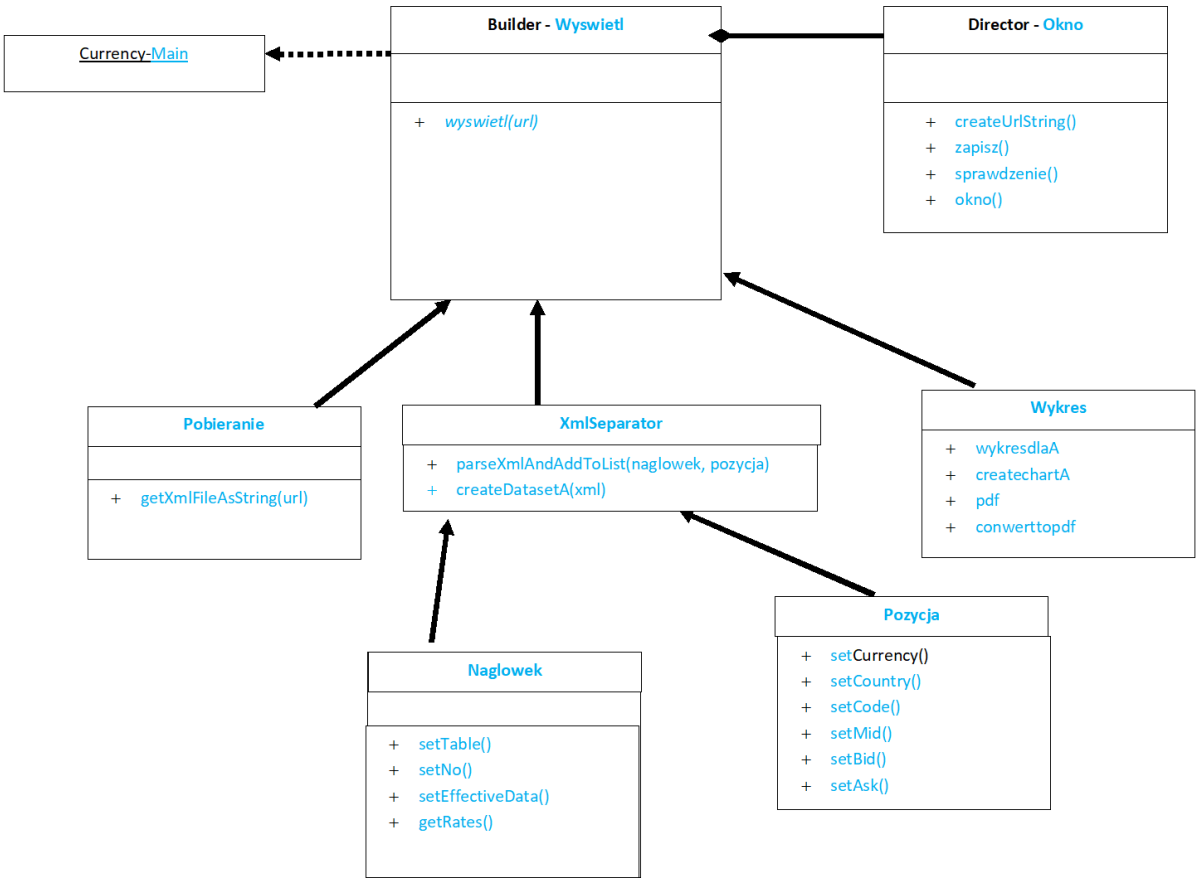
Dodano informacje podstawowe na temat dostępnych tabel NBP, oraz o zakresie dat i dni.

Brak bezpośredniej możliwości drukowania, Istnieje możliwość zapisu wyświetlonych danych do pliku.

Generowane są wykresy dla tabeli typu A, wykresy te można zapisać do pliku PDF. Wykres jest generowany dla 4 najważniejszych walut na świecie (EUR, USD, CHF, GBP).

Wzorzec projektowy





Implementacja

Program został napisany w języku Java wspierany językiem Groovy. Groovy jest to skryptowy język wzorowany na składni Javy

Program został podzielony na 7 klas. Naglowek, Pozycja, Pobieranie, XmlSeparator, Wyswietl, Wykres, Okno oraz Main. Klasy Naglowek oraz Pozycja to klasy, które reprezentują wewnętrzną strukturę pliku xml pobranego ze strony NBP. Pozostałe klasy są odpowiedzialne za odczyt oraz wyświetlenie danych, stworzenie okna aplikacji, pobranie danych z Internetu. Klasa main steruje całością programu.

Program działa tylko z połączeniem internetowym. Kod programu oraz dokumentacja znajduje się w repozytorium GitHub.

Klasa Naglowek:

```
package Waluty

class Naglowek {
    private String table = null
    private String no = null
    private String effectiveDate = null
    private List<Pozycja> rates = new ArrayList<>()

    void setTable(String table) {
        this.table = table
    }

    void setNo(String no) {
        this.no = no
    }

    void setEffectiveDate(String effectiveDate) {
        this.effectiveDate = effectiveDate
    }

    List<Pozycja> getRates() {
        return rates
    }

    @Override
    String toString() {
        return "Tabela '" + table + "' Walut dla dnia: '"+ effectiveDate +
"'."
    }
}
```

Klasa odczytuje rodzaj tabeli oraz datę (daty) z xml

Klasa Pozycja:

```
package Waluty

class Pozycja {
    private String country = null
    private String currency = null
    private String code = null
    private String mid = null
    private String bid = null
    private String ask = null

    void setCountry(String country) {
        this.country = country
    }

    void setCurrency(String currency) {
        this.currency = currency
    }

    void setCode(String code) {
        this.code = code
    }

    void setMid(String mid) {
        this.mid = mid
    }

    void setBid(String bid) {
        this.bid = bid
    }

    void setAsk(String ask) {
        this.ask = ask
    }

    @Override
    String toString() {
        StringBuilder builder = new StringBuilder()

        if (country) {
            builder.append("Państwo: ")
            builder.append(country).append("\n")
        }

        if (currency){
            builder.append("Waluta: ")
            builder.append(currency).append("\n")
        }

        if (code) {
            builder.append("Kod: ")
            builder.append(code).append("\n")
        }

        if (mid) {
            builder.append("Wartość: ")
            builder.append(mid).append(" zł\n")
        }
        if (bid) {
            builder.append("Sprzedaż: ")
```

```
            builder.append(bid).append(" zł\n")
        }
        if (ask) {
            builder.append("Kupno: ")
            builder.append(ask).append(" zł\n")
        }

        return builder.toString()
    }
}
```

Klasa ta odczytuje dokładne dane dotyczące walut oraz ich wartości z XML

Klasa Pobieranie:

```
package Waluty

class Pobieranie {
    static def getXmlFileAsString(String urlString) {
        def builder = new StringBuilder()
        def inputStream = null
        def reader
        def line
        def url

        try {
            url = new URL(urlString)
            inputStream = url.openStream()
            reader = new BufferedReader(new InputStreamReader(inputStream))

            while ((line = reader.readLine()) != null) {
                builder.append(line)
            }
        } catch (MalformedURLException mue) {
            mue.printStackTrace()
        } catch (IOException ioe) {
            ioe.printStackTrace()
        } finally {
            try {
                if (inputStream != null) inputStream.close()
            } catch (IOException ioe) {
                ioe.printStackTrace()
            }
        }

        return builder.toString()
    }
}
```

Klasa odczytuje po kolei każdy wiersz z podanego pliku dostepnego pod adresem url oraz zwraca go w postaci StringBuildera. Zwracany jest cały plik xml.

Klasa XmlSeparator:

```
package Waluty

import org.jfree.data.category.CategoryDataset
import org.jfree.data.category.DefaultCategoryDataset

class XmlSeparator {
    static def parseXmlAndAddToList(String xml, List<Naglowek>
exchangeRatesTables) {
        def document = new XmlParser().parseText(xml)
        Naglowek table = null
        Pozycja rate = null
        document.ExchangeRatesTable.each {
            bk ->
                table = new Naglowek()

                table.setTable("${bk.Table.text()}")
                table.setNo("${bk.No.text()}")
                table.setEffectiveDate("${bk.EffectiveDate.text()}")
                bk.Rates.Rate.each {
                    bt ->
                        rate = new Pozycja()

                        rate.setCountry("${bt.Country.text()}")
                        rate.setCurrency("${bt.Currency.text()}")
                        rate.setCode("${bt.Code.text()}")
                        rate.setMid("${bt.Mid.text()}")
                        rate.setAsk("${bt.Ask.text()}")
                        rate.setBid("${bt.Bid.text()}")

                        table.getRates().add(rate)
                }
                exchangeRatesTables.add(table)

        }
    }
     CategoryDataset createDatasetA(String xml) {
         def document = new XmlParser().parseText(xml)
         Naglowek table = null
         Pozycja rate = null
         String []daty =new String[100]
         String []eur =new String[100]
         String []usd =new String[100]
         String []chf =new String[100]
         String []gbp =new String[100]
         double wartosceur
         double wartoscusd
         double wartoscchf
         double wartoscgbp
         int i=0
         int j=0
         document.ExchangeRatesTable.each {

             bk ->
                 table = new Naglowek()
                 daty[i]=bk.EffectiveDate.text()
                 bk.Rates.Rate.each {
                     bt ->
                         rate = new Pozycja()
```

```
                        table.getRates().add(rate)
                        if (bt.Code.text()=="EUR")
                            eur[i]=bt.Mid.text()
                        if (bt.Code.text()=="USD")
                            usd[i]=bt.Mid.text()
                        if (bt.Code.text()=="CHF")
                            chf[i]=bt.Mid.text()
                        if (bt.Code.text()=="GBP")
                            gbp[i]=bt.Mid.text()
                }
                i++
        }


        String series1 = "USD";
        String series2 = "EUR";
        String series3 = "CHF";
        String series4 = "GBP";
        DefaultCategoryDataset dataset = new DefaultCategoryDataset()
        for (j==0;j<=i;j++){
        if(daty[j]!=null) {
            wartosceur = Double.parseDouble(eur[j])
            wartoscchf = Double.parseDouble(chf[j])
            wartoscgbp = Double.parseDouble(gbp[j])
            wartoscusd = Double.parseDouble(usd[j])

            dataset.addValue(wartoscusd, series1, daty[j])
            dataset.addValue(wartosceur, series2, daty[j])
            dataset.addValue(wartoscchf, series3, daty[j])
            dataset.addValue(wartoscgbp, series4, daty[j])
           }
        }
       return dataset

    }
}
```

Klasa odczytuje z naszych pobranych danych interesujące nas dane, które są dodawane do listy. Jest tu użyte Groovy closure. Do wyciągnięcia poszczególnych danych używa zintegrowanego w Groovy języka wyrażeń ścieżek GPath. Odwołuje się do poszczególnych pól w pliku xml a nie bezpośrednio z poziomu kodu. W klasie znajduje się funkcja createdatasetA, która generuje zastaw danych, które są potrzebne do stworzenia wykresu.

Klasa Wyświetl:

```
package Waluty

class Wyswietl {
    String wyswietl(String url){
        def builder = new StringBuilder()
        String urlString = url
        String xml = Pobieranie.getXmlFileAsString(urlString)

        List<Naglowek> tables = new ArrayList<>()
        XmlSeparator.parseXmlAndAddToList(xml, tables)

        for (ex in tables) {
            builder.append(ex)
            builder.append("\n")

        for (rx in ex.getRates()) {
            builder.append(rx)
            builder.append("\n")
        }

    }
        return builder.toString()
}
}
```

Klasa, która pobiera dokładnie dane które nas interesują I zwraca je w postaci StringBuildera.

Klasa Wykres:

```
package Waluty

import javax.swing.JFrame
import java.awt.Graphics2D
import java.awt.geom.Rectangle2D
import com.lowagie.text.Document
import com.lowagie.text.DocumentException
import com.lowagie.text.Rectangle
import com.lowagie.text.pdf.DefaultFontMapper
import com.lowagie.text.pdf.PdfContentByte
import com.lowagie.text.pdf.PdfTemplate
import com.lowagie.text.pdf.PdfWriter
import org.jfree.chart.axis.CategoryAxis
import org.jfree.chart.axis.CategoryLabelPositions
import org.jfree.chart.ChartFactory
import org.jfree.chart.ChartPanel
import org.jfree.chart.JFreeChart
import org.jfree.chart.plot.CategoryPlot
import org.jfree.chart.plot.PlotOrientation
import org.jfree.data.category.CategoryDataset
import org.jfree.ui.RefineryUtilities


class Wykres {
    void wykresDlaA(String url) {
        JFrame frame = new JFrame("Waluty")

        String urlString = url
        String xml = Pobieranie.getXmlFileAsString(urlString)
        XmlSeparator wykres = new XmlSeparator()
        CategoryDataset dataset = wykres.createDatasetA(xml)
        JFreeChart chart = createChartA(dataset)
        ChartPanel chartPanel = new ChartPanel(chart)
        frame.setSize(400, 300)
        frame.setContentPane(chartPanel)
        RefineryUtilities.centerFrameOnScreen(frame)
        frame.setVisible(true)
    }
    private JFreeChart createChartA(final CategoryDataset dataset) {

        final JFreeChart chart = ChartFactory.createBarChart3D(
                "Wykres walut z tabeli A",         // chart title
                "Data",               // domain axis label
                "Wartość",                  // range axis label
                dataset,                  // data
                PlotOrientation.VERTICAL, // orientation
                true,                     // include legend
                true,                     // tooltips?
                false                     // URLs?
        );
        final CategoryPlot plot = chart.getCategoryPlot()
        final CategoryAxis domainAxis = plot.getDomainAxis()
        domainAxis.setCategoryLabelPositions(

CategoryLabelPositions.createUpRotationLabelPositions(Math.PI / 5.0)
        );
        return chart

    }
```

```
   void pdf(String url) {

       String urlString = url
       String xml = Pobieranie.getXmlFileAsString(urlString)
       XmlSeparator wykres = new XmlSeparator()
       final CategoryDataset dataset = wykres.createDataset(xml)
       convertToPdf(createChart(dataset), 600, 400, "wykres.pdf")

    }


    void convertToPdf(JFreeChart chart, int width, int height, String
filename) {

        Document document = new Document(new Rectangle(width, height))
        try {
            PdfWriter writer
            writer = PdfWriter.getInstance(document, new
FileOutputStream(filename))
            document.open()
            PdfContentByte cb = writer.getDirectContent()
            PdfTemplate tp = cb.createTemplate(width, height)
            Graphics2D g2d = tp.createGraphics(width, height, new
DefaultFontMapper())
            Rectangle2D r2d = new Rectangle2D.Double(0, 0, width, height)
            chart.draw(g2d, r2d)
            g2d.dispose()
            cb.addTemplate(tp, 0, 0)
        }
        catch (DocumentException de) {
            de.printStackTrace()
        }
        catch (FileNotFoundException e) {
            e.printStackTrace()
        }
        document.close()
    }
}
```

Klasa wykres jest odpowiedzialna za inicjaję wykresu, ustawienie jego parametrów oraz wygenerowanie pliku PDF z wykresem.

Klasa Main:

```
package Waluty

import org.jfree.ui.RefineryUtilities

import javax.swing.*

class Main {
    static void main(String[] args) {
        def frame = new JFrame("Waluty")
        frame.setContentPane(new Okno().plane)
        frame.setSize(800,600)
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
        RefineryUtilities.centerFrameOnScreen(frame)
        frame.setVisible(true)
        JOptionPane.showMessageDialog(null,"Program
walutowy(beta)\nObowiązkowym polem jest pole rodzaju tabeli.\n" +
```

```
                "W przypadku nie podania daty, wyświetlane są najświeższe
dane.\nMaksymalny przedział to 91dni\nDane archiwalne są dostępne od 2
stycznia 2002r. (2002-01-02)",
                "Program",1
        )
    }
    }
```

Klasa inicjująca okno, oraz jego parametry.

Klasa Okno.

```
package Waluty;

import javax.swing.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.*;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Okno {
    JPanel plane;
    JTextArea textArea1;
    JPanel wybor;
    JTextField datapocz;
    JTextField datakon;
    JButton infoButton;
    JButton pokażCenyButton;
    ButtonGroup button;
    JButton wyczyśćPoleButton;
    JButton zapiszButton;
    JPanel przyciski;
    JRadioButton aRadioButton;
    JRadioButton bRadioButton;
    JRadioButton cRadioButton;
    private JButton infoOProgramieButton;
    String dane = null;

    //  f
    public Okno() {

        button = new ButtonGroup();
        button.add(aRadioButton);
        button.add(bRadioButton);
        button.add(cRadioButton);

        infoButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                JOptionPane.showMessageDialog(null, "a - tabela kursów
średnich walut obcych \n" +
                        "b - tabela kursów średnich walut
niewymienialnych\n" +
                        "c - tabela kursów kupna i sprzedaży",
"Informacje", 1);
            }
        });

        pokażCenyButton.addMouseListener(new MouseAdapter() {
            @Override
```

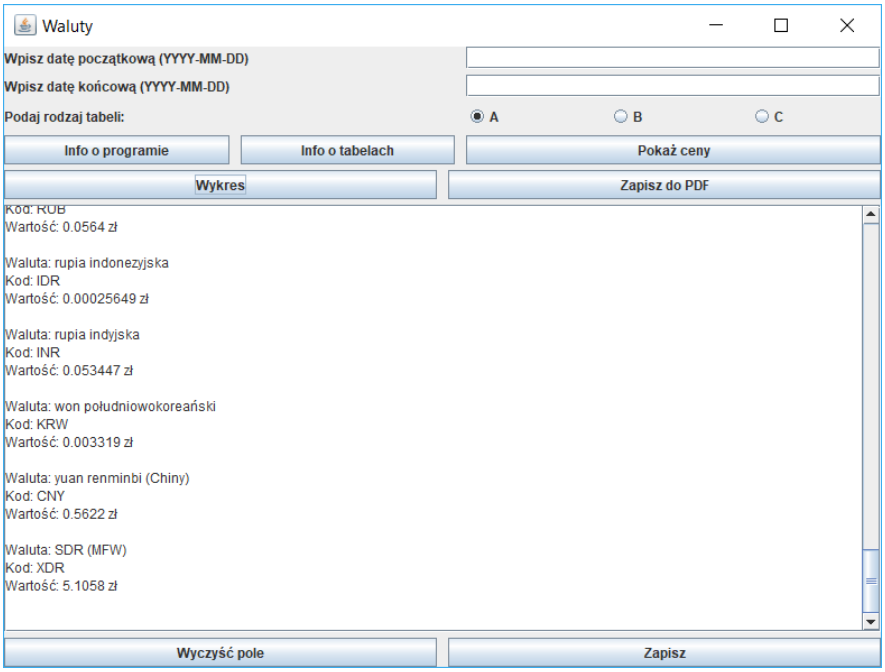
```
            public void mouseClicked(MouseEvent e) {
                Wyswietl wy = new Wyswietl();
                textArea1.setEditable(false);
                String urlString = createUrlString();
                String spr =
("http://api.nbp.pl/api/exchangerates/tables/null///?format=xml");
                if (urlString.equals(spr)) {
                    JOptionPane.showMessageDialog(null, "BŁĄD! Nie podano
żądnych danych do wyszukiwania\n" +
                            "Musisz podać przynajmniej rodzaj", "Błąd", 2);
                } else {
                    if (datakon.getText().isEmpty() &&
datapocz.getText().isEmpty()) {
                        textArea1.setText(wy.wyswietl(urlString));
                    } else if (sprawdzanie() == true) {
                        textArea1.setText(wy.wyswietl(urlString));
                    }
                }
            }
        });
        wyczyśćPoleButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                textArea1.setText("");
                datapocz.setText("");
                datakon.setText("");
            }
        });
        zapiszButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                zapisz();
            }
        });
        infoOProgramieButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                JOptionPane.showMessageDialog(null, "Program
walutowy(beta)\nObowiązkowym polem jest pole rodzaju tabeli.\n" +
                                "W przypadku nie podania daty, wyświetlane
są najświeższe dane.\nMaksymalny przedział to 91dni\nDane archiwalne są
dostępne od 2 stycznia 2002r. (2002-01-02)",
                        "Program", 1
                );
            }
        });
    }
    // funkcja tworząca adres url do naczego xml

    String createUrlString() {

        String rodzaj = null;
        if (aRadioButton.isSelected())
            rodzaj = "a";
        if (bRadioButton.isSelected())
            rodzaj = "b";
        if (cRadioButton.isSelected())
            rodzaj = "c";

        StringBuilder builder = new StringBuilder();
        builder.append("http://api.nbp.pl/api/exchangerates/tables/");
```

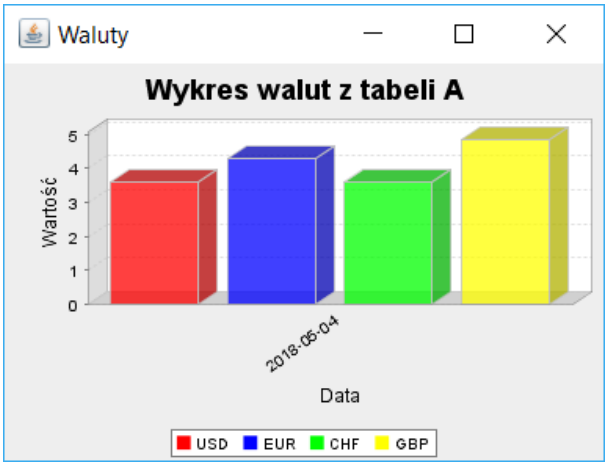
```
        builder.append(rodzaj);
        builder.append("/");
        builder.append(datapocz.getText());
        builder.append("/");
        builder.append(datakon.getText());
        builder.append("/?format=xml");
        return builder.toString();
    }

    // funkcja sprawdzająca poprawność wpisywanej daty
    boolean sprawdzanie() {
        Pattern data = Pattern.compile("^((?:20)\\d\\d)[- /.](0[1-
9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])$");
        int a = 0;
        int b = 0;
        String datain = datapocz.getText();
        String dataout = datakon.getText();
        Matcher spr_datapocz = data.matcher(datain);
        Matcher spr_datakon = data.matcher(dataout);
        if (datain.isEmpty()) {
        } else {
            if (spr_datapocz.matches() == false) {
                JOptionPane.showMessageDialog(null, "Podany został zły
format daty początkowej.\n" +
                        "Prawidłowy format: YYYY-MM-DD", "Błąd", 2);
                a = 1;
            }
        }

        if (dataout.isEmpty()) {
        } else {
            if (spr_datakon.matches() == false) {
                JOptionPane.showMessageDialog(null, "Podany został zły
format daty końcowej.\n" +
                        "Prawidłowy format: YYYY-MM-DD", "Błąd", 2);
                b = 1;
            }
        }
        if (a == 1 || b == 1)
            return false;
        else
            return true;
    }

    // funkcja zapisująca dane do pliku
    void zapisz() {
        JFileChooser fc = new JFileChooser();

        if (fc.showSaveDialog(null) == JFileChooser.APPROVE_OPTION) {
            File plik = fc.getSelectedFile();

            try {
                PrintWriter pw = new PrintWriter(plik);
                Scanner skaner = new Scanner(textArea1.getText());

                while (skaner.hasNextLine()) {
                    pw.println(skaner.nextLine());

                }
                pw.close();
            } catch (FileNotFoundException ex) {
```

```
                    ex.printStackTrace();
                }
            }
        }
}
```

Najbardziej rozbudowana klasa. Posiada metodę zapisz, która zapisuje dane wyświetlane w polu tekstowym do pliku. Metoda sprawdzanie jest odpowiedzialna za sprawdzenie poprawności wprowadzonej daty. Sprawdza czy podana data jest taka jak podane wyrażenie regularne. Metoda createUrlString tworzy url z którego będzie pobierany xml. Zwraca gotowy string z url'em. Metoda Okno, która jest konstruktorem obsługuje całe okno aplikacji. Tam znajdują się komponenty okna oraz obsługa przycisków.

Gotowa aplikacja:

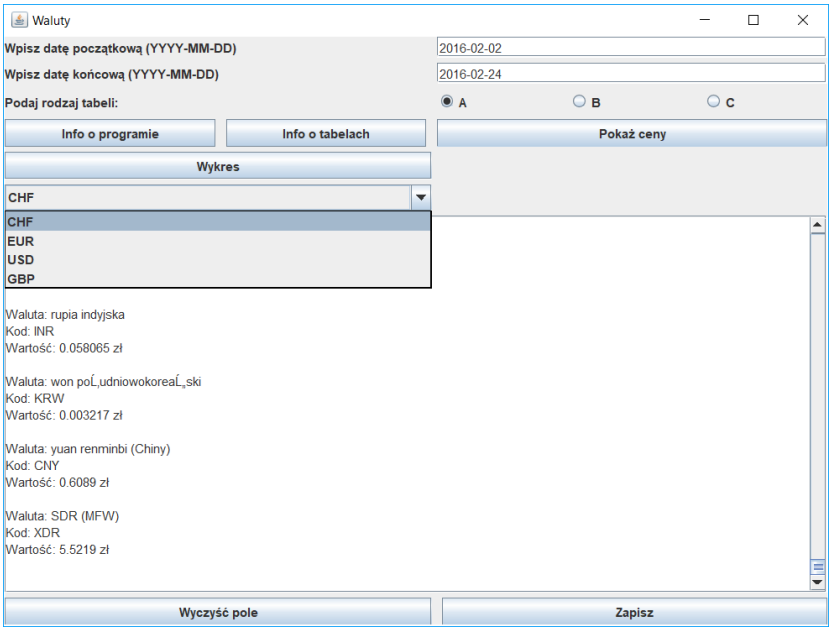




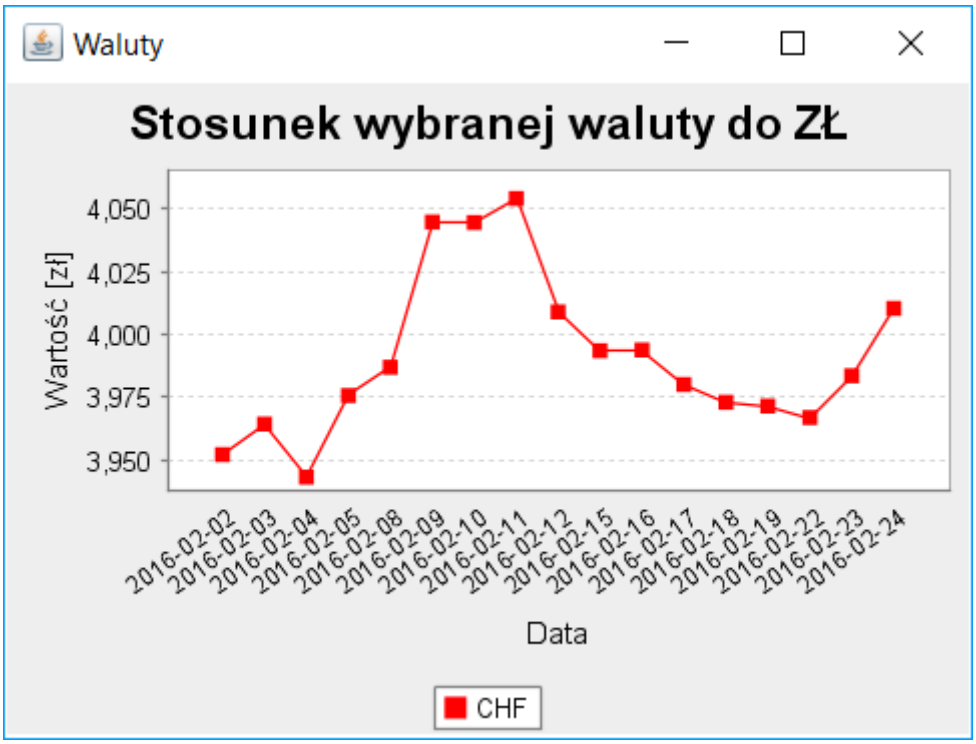
Dodatkowa funkcjonalność – Michał Cholewinski, Marcin Czarkowski Piort Rudof Daniel Kwaśniewski.

Naszym zadaniem było dodanie nowego wykresu liniowego, który wyświetli wartość wybranej waluty obcej względem zł.

Wykres liniowy powstaje tylko i wyłącznie gdy podany zostanie zakres dat. Po wybraniu opcji „pokaż ceny” przy założeniu, że wybrało się tabelę typu ‘A’ oraz podano przedział datowy zgodny z wymaganiami, pojawia się dodatkokwe polr typu ComboBox z możliwością wyboru dla jakiej waluty chcemy wygenerować nasz wykres liniowy. Są to waluty takie same jak w przypadku wykresu słupkowego (EUR, USD, CHF, GBP)



Po wyborze odpowiedniej waluty oraz wybraniu opcji „Wykres” pojawi się domyślny wykres słupkowy oraz dodany przez nas wykres liniowy.



Zmiany w kodzie.

Do klasy XmlSeparator dodana została funkcja tworząca dane potrzebne do wykresu. Pobiera je z XML.. Wykres korzysta z biblioteki JFreeChart.

```
CategoryDataset createDataset_linowy(String xml, String seria) {
    def document = new XmlParser().parseText(xml)
    Naglowek table = null
    Pozycja rate = null
    String []daty =new String[100]
    String []war =new String[100]
    double wartosc
    int i=0
    int j=0
    document.ExchangeRatesTable.each {

        bk ->
            table = new Naglowek()
            daty[i]=bk.EffectiveDate.text()
            bk.Rates.Rate.each {
                bt ->
                    rate = new Pozycja()
                    if (bt.Code.text()==seria)
                        war[i]=bt.Mid.text()
            }
            i++
    }

    String series1 = seria
    DefaultCategoryDataset dataset = new DefaultCategoryDataset()
    for (j==0;j<=i;j++){
        if(daty[j]!=null) {
            wartosc = Double.parseDouble(war[j])
            dataset.addValue(wartosc, series1, daty[j])
        }
    }
    return dataset

}
```

W klasie Wykres.grovy dodane zostały 2 funkcje. Funkcja createChart_liniowy tworzy wykres z naszych danych, dodaje odpowiednie etylkiety wykresu oraz zakres skali osi X. Funkcja wykres_liniowy jest odpowiedzialna za inicjalizację okna z wtkresem.

```
private JFreeChart createChart_liniowy(final CategoryDataset dataset) {

    final JFreeChart chart = ChartFactory.createLineChart(
            "Stosunek wybranej waluty do ZŁ",         // chart title
            "Data",               // domain axis label
            "Wartość [zł]",                  // range axis label
            dataset,                  // data
            PlotOrientation.VERTICAL, // orientation
            true,                     // include legend
            true,                     // tooltips?
            false                     // URLs?
    );
    final CategoryPlot plot = chart.getCategoryPlot()
    final CategoryAxis domainAxis = plot.getDomainAxis()
    domainAxis.setCategoryLabelPositions(
            CategoryLabelPositions.createUpRotationLabelPositions(Math.PI /
```

```
5.0)
    )
    final LineAndShapeRenderer renderer = (LineAndShapeRenderer)
plot.getRenderer();
      renderer.setDrawShapes(true);
    final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
    rangeAxis.setAutoRangeIncludesZero(false);
    rangeAxis.setUpperMargin(0.10);
    return chart

}
```

```
void wykres_liniowy(String url, String seria) {
    JFrame frame = new JFrame("Waluty")

    String urlString = url
    String xml = Pobieranie.getXmlFileAsString(urlString)
    XmlSeparator wykres = new XmlSeparator()
    CategoryDataset dataset = wykres.createDataset_linowy(xml, seria)
    JFreeChart chart = createChart_liniowy(dataset)
    ChartPanel chartPanel = new ChartPanel(chart)
    frame.setSize(400, 300)
    frame.setContentPane(chartPanel)
    RefineryUtilities.centerFrameOnScreen(frame)
    frame.setVisible(true)
}
```

Do klasy Okno.java w miejscu obsługi lisenera przycisku “Wykres” dodaliśmy warunek sprawdzający czy podano wartości w polach dat oraz wyświetlenie dodatkowego wykresu. Dodane zastało równiej pole ComboBox za pomocą formularza graficznego Okno.form.

Założenia, dokumentacja oraz gotowy program, który otrzymaliśmy od grupy poprzedniej pozwoliły na wporawadzenie dodatkowej funkcjonalności. Po zapoznaniu się z kondem oraz dokumentacją wprowadzenie dodatkowego wykres nie było trodnością. Musieliśmy zapoznać się z dokumentacją biblioteki JFreeChart oraz jej możliwościami. Ostatecznie udało się dodać wykres, opcję wyboru waluty, która nas interesuje.