PROGRAMOWANIE OBIEKTOWE

UNIWERSYTET PRZYRODNICZO – HUMANISTYCZNY

W SIEDLCACH

SAMOOBSŁUGOWA STACJA BENZYNOWA

Program symulujący pracę samoobsługowej stacji benzynowej.

DOKUMENTACJA TECHNICZNA

PAULINA CZUBAJ
INFORMATYKA I ROK
UPH SIEDLCE

Siedlce, 24 sierpnia 2016r.

1. SPECYFIKACJA PROBLEMU

Zadanie do wykonania brzmi:

Samoobsługowa stacja benzynowa.
Program symulujący pracę samoobsługowej stacji benzynowej.

Jest to zadanie 4 z listy tematów projektu.
Ponadto wiążące są następujące wymogi projektowe:

- Przynajmniej dwa stanowiska do tankowania.
- Tankowanie.
- Obliczanie należności, inkasowanie zapłaty, wypłata reszty.
- Uwzględnij zwykłego użytkownika programu i administratora.
- Uwzględnij sytuację braku awarię sprzętu.
- Uwzględnij klienta i pracownika stacji.

oraz następujące wymogi techniczne:
Rozwiązanie zadania indywidualnego musi wykorzystywać:

- ideę programowania obiektowego,
- mechanizmy JAVY takie, jak dziedziczenie, interfejsy, klasy i metody abstrakcyjne,
- obsługa sytuacji wyjątkowych oraz w niektórych projektach mechanizm wątków,
- graficzny interfejs użytkownika.

2. OMÓWIENIE ISTOTY PROGRAMU

Za najważniejsze cele obrałam sobie spełnienie zadanych wymogów. Obsługa programu opiera się na podziale ról – program opracowany jest zarówno dla klientów naszej Stacji, jak i ich pracowników. Klient ma do użycia dwie funkcje w menu – może on sprawdzić ceny paliwa bądź zatankować samochód poprzez wypełnienie formularza. W tym celu przyjęłam istnienie jednego głównego zbiornika paliwa i połączonych z nim dwóch dystrybutorów – dystrybutora po lewej stronie oraz po prawej stronie, tak aby mógł zostać obsłużony samochód z wlewem paliwa po dowolnej stronie. Możemy zatankować cztery rodzaje paliw: PB95, PB98, LPG oraz ON. Możemy również wskazać, w jaki sposób chcemy określić ilość paliwa – ilościowo (w cm^3) czy wartościowo (w PAU – walucie obowiązującej na Stacji Paliw PAULA). Po złożeniu zamówienia program sprawdza, czy w zbiorniku znajduje się wymagana ilość paliwa oraz czy dany dystrybutor jest w danym momencie czynny. Jeśli wszystko się zgadza, to zostaje wygenerowane zamówienie i następuje moment zapłaty. Program kontroluje, kiedy wpłynie odpowiednia kwota do zapłaty oraz, w razie potrzeby, wydaje resztę. Również po każdym zrealizowanym zamówieniu program odejmuje zużytą ilość paliwa od stanu zbiornika, na bieżąco kontrolując, czy nie trzeba go uzupełnić. Aby dostać się do panelu pracownika wymagane jest podanie hasła. Po poprawnym jego wpisaniu następuje otwarcie panelu. Wśród dostępnych interakcji znajdujemy tam możliwość sprawdzenia stanu zbiornika oraz uzupełnienie jego zasobów do maksimum. Jest także cennik wraz z możliwością zmiany ceny. Każda zmiana ceny jest zapisywana do pliku

tekstowego, aby mogła zostać przesłana do analizy firmom zajmującym się analizą rynku cen paliwa. Jest również opcja zarządzania dystrybutorami – w razie awarii należy je wyłączyć z obiegu. Ostatnią opcją jest podgląd ostatnio zrealizowanych zamówień.

3. KOD POSZCZEGÓLNYCH KLAS WRAZ Z OMÓWIENIEM

3.1. BrakPaliwaException

```
public class BrakPaliwaException extends Exception{

}
```

Jest to fragment, w którym zawarta jest obsługa wyjątku – brak paliwa w zbiorniku, co uniemożliwia zatankowanie samochodu.

3.2. Cennik

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Cennik {

	public interface CennikListener{
		public void cenaZmienilaSie();
	}

	public static final Cennik FAKE_CENNIK = new Cennik(){
		{
			this.ustawStan(RodzajPaliwa.LPG, 500);
			this.ustawStan(RodzajPaliwa.ON, 500);
			this.ustawStan(RodzajPaliwa.PB95, 500);
			this.ustawStan(RodzajPaliwa.PB98, 500);
		}
	};

	private Map<RodzajPaliwa, Integer> stany = new HashMap<>();
	private List<CennikListener> listeners = new ArrayList<>();

	public Cennik() {
		for (RodzajPaliwa p : RodzajPaliwa.values()) {
			stany.put(p, 100);
		}
	}
	public boolean jestPaliwo(RodzajPaliwa p){
		return this.podajCene(p) > 0;
	}

	public int podajCene(RodzajPaliwa p){
		return stany.get(p);
	}

	public void ustawStan(RodzajPaliwa p, int w){
		notifyListeners();
```

```
                stany.put(p, w);
        }

        public void zmienStan(RodzajPaliwa p, int w){
                int s = this.podajCene(p);
                s += w;
                ustawStan(p, s);
        }

        private void notifyListeners(){
                for(CennikListener l:this.listeners){
                        l.cenaZmienilaSie();
                }
        }
        public boolean addListener(CennikListener e) {
                return listeners.add(e);
        }
}
```

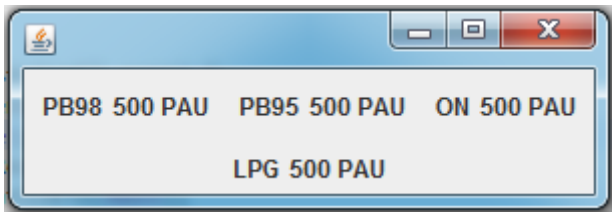
W tym miejscu jest utworzony interfejs, który mówi co klasa implementująca może robić, ale oczywiście nie jak ma to robić. Jest utworzony FAKE_CENNIK, który wykorzystałam do testowania programu w innych klasach. Jest utworzona mapa, czyli klasa implementująca interfejs java.util.Map<K,V>. Mapa reprezentuje związki obiektów z kluczami, czyli w tym wypadku zależność rodzaju paliwa z jego ilością. Są tu także tworzone zmienne i metody wykorzystywane w dalszych częściach projektu.

3.3. CennikOkienko

```
import java.awt.Rectangle;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class CennikOkienko extends PodstawoweOkienkoCena {

        public CennikOkienko(Cennik c) {
                super(c);
        }

        protected void dodajAkcjeWWierszu(RodzajPaliwa p, JPanel panel) {

        }

        public static CennikOkienko odtworz(Cennik c) {
                CennikOkienko frame = new CennikOkienko(c);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 300, 300));
                frame.setLocationRelativeTo(null);
                return frame;
        }

        public static void main(String[] args) {
                JFrame frame = new CennikOkienko(Cennik.FAKE_CENNIK);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 300, 300));
                frame.setLocationRelativeTo(null);
```

```
	}
}
```

Jest to miejsce, w którym następuje dziedziczenie po klasach, a dokładnie po klasie PodstawoweOkienkoCena. Jest tu utworzone okno cennika bez możliwości edytowania cen, czyli Cennik wyświetlający po wybraniu odpowiedniej opcji w panelu użytkownika.

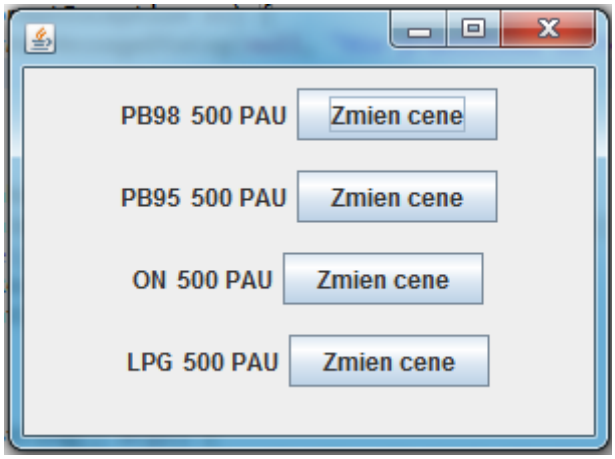


3.4. CennikOkienkoZKasowaniem

```
import java.awt.Rectangle;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class CennikOkienkoZKasowaniem extends PodstawoweOkienkoCena {

	public CennikOkienkoZKasowaniem(Cennik c) {
		super(c);
	}

	protected void dodajAkcjeWWierszu(RodzajPaliwa p, JPanel panel) {
		JButton bWypelnij;
		panel.add(bWypelnij = new JButton("Zmien cene"));
		bWypelnij.addActionListener(e -> {
			try {
				int cena =
Integer.parseInt(JOptionPane.showInputDialog("Podaj nowa cene"));

	CennikOkienkoZKasowaniem.this.cennik.ustawStan(p, cena);

				setVisible(false);
				dispose();
			} catch (NumberFormatException ex) {
				JOptionPane.showMessageDialog(null, "Nie
prawidlowa wartosc");
			}
		});
	}

	public static CennikOkienkoZKasowaniem odtworz(Cennik c) {
		CennikOkienkoZKasowaniem frame = new
CennikOkienkoZKasowaniem(c);
		frame.setVisible(true);
		frame.setBounds(new Rectangle(0, 0, 300, 300));
		frame.setLocationRelativeTo(null);
		return frame;
	}

	public static void main(String[] args) {
```

```
                JFrame                frame                =                new
CennikOkienkoZKasowaniem(Cennik.FAKE_CENNIK);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 300, 300));
                frame.setLocationRelativeTo(null);
        }

}
```

Jest to miejsce w którym utworzone jest okienko z kasowaniem, czyli okienko dostępne dla pracownika Stacji. Trzeba tu pilnować, aby wprowadzona wartość liczbowa została zapisana rzeczywiście jako liczba. Jest tu również zastosowana obsługa wyjątków, ponieważ w momencie, gdy nie zostanie wprowadzona liczba, następuje wyświetlenie się komunikatu nakazującego powtórne wprowadzenie.

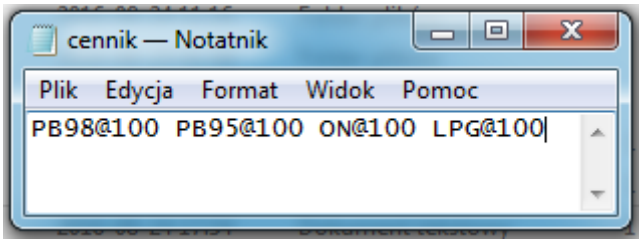


3.5. CennikZapisOdczyt

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class CennikZapisOdczyt {
        private Cennik cennik;

        public CennikZapisOdczyt(Cennik cennik) {
                super();
                this.cennik = cennik;
        }

        public void load() {
                try {
                        System.out.println(new
File("cennik.txt").getAbsolutePath());
                        BufferedReader    br    =    new    BufferedReader(new
FileReader("cennik.txt"));
                        String line;
                        while ((line = br.readLine()) != null) {
                                String[] parts = line.split("@");
```

```
                                RodzajPaliwa p =
RodzajPaliwa.valueOf(parts[0]);
                                int cena = Integer.parseInt(parts[1]);
                                cennik.ustawStan(p, cena);
                        }
                        br.close();
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

        public void save() {
                try {
                        BufferedWriter bw = new BufferedWriter(new
FileWriter("cennik.txt"));
                        for (RodzajPaliwa paliwo : RodzajPaliwa.values()) {
                                bw.write(paliwo.name());
                                bw.write("@");
                                bw.write(""+cennik.podajCene(paliwo));
                                bw.write("\n");
                        }
                        bw.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

}
```

W tym miejscu następuje zapis zmian cen paliwa do pliku oraz odczyt ich z pliku. W pierwszej kolejności tworzę plik tekstowy cennik.txt. Klasa BufferedReader umożliwia odczytywanie tekstu liniami – udostępnia metodę readLine(), którą wywołujemy w kolejnej linii. Poszczególne ceny podczas zapisywania oddzielam znakiem @ aby nie pomylić się później w odczytywaniu danych. Następuje także kontrola wyjątków, czy dany plik jest utworzony oraz czy nie nastąpiła nieudana próba odczytu lub zapisu do pliku.



3.6. DYSTRYBUTOR

```
import java.util.Arrays;
import java.util.List;

public class Dystrybutor {

        public static final List<Dystrybutor> FAKE_DYSTRYBUTORY;
        static {
```

```
                FAKE_DYSTRYBUTORY = Arrays.asList(//
                                new                         Dystrybutor("Prawy",
Zbiornik.FAKE_ZBIORNIK, MenadzerSprzedazy.FAKE_MENADZER), //
                                new                          Dystrybutor("Lewy",
Zbiornik.FAKE_ZBIORNIK, MenadzerSprzedazy.FAKE_MENADZER)//
                );
        }
        private final String name;
        private final Zbiornik zbiornik;
        private final MenadzerSprzedazy menadzer_sprzedazy;
        private boolean wlaczony = true;

        public      Dystrybutor(String      name,      Zbiornik      zbiornik,
MenadzerSprzedazy menadzer_sprzedazy) {
                this.name = name;
                this.zbiornik = zbiornik;
                this.menadzer_sprzedazy = menadzer_sprzedazy;
        }

        public boolean isWlaczony() {
                return wlaczony;
        }

        public void setWlaczony(boolean wlaczony) {
                this.wlaczony = wlaczony;
        }

        public void przelaczWlaczony() {
                setWlaczony(!isWlaczony());
        }

        public String getName() {
                return name;
        }

        @Override
        public String toString() {
                return this.getName();
        }

        public    void    nalej(RodzajPaliwa    p,    int    ilosc)    throws
BrakPaliwaException{
                zbiornik.zmienStan(p, -1 * ilosc);
        }
}
```

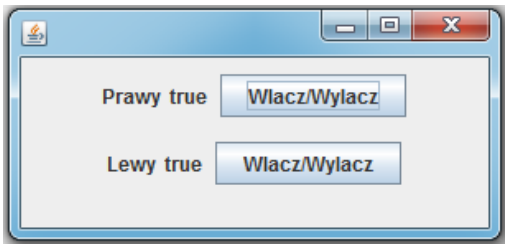
Jest tu tworzony FAKE_DYSTRYBUTOR, który wykorzystałam do testowania programu w innych klasach. Mamy zmienne wchodzące w skład klasy Dystrybutor, jak i również zarządzanie dystrybutorami w przypadku awarii.

3.7. DystrybutoryOkienko

```
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.JButton;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class DystrybutoryOkienko extends JFrame {

	private List<Dystrybutor> dystrybutory;

	public DystrybutoryOkienko(List<Dystrybutor> dystrybutory) {
		this.dystrybutory = dystrybutory;

		JPanel lista = new JPanel();
		for (Dystrybutor d : dystrybutory) {
			JPanel panel = new JPanel();
			panel.add(new JLabel(d.getName()));
			panel.add(new JLabel(d.isWlaczony() + " "));
			JButton bPrzelacz;
			panel.add(bPrzelacz = new JButton("Wlacz/Wylacz"));
			bPrzelacz.addActionListener(new ActionListener() {

				@Override
				public void actionPerformed(ActionEvent e) {
					d.przelaczWlaczony();
					setVisible(false);
					dispose();
				}
			});
			lista.add(panel);
		}
		add(lista);
	}

	public static DystrybutoryOkienko odtworz(List<Dystrybutor>
dystrybutory) {
		DystrybutoryOkienko frame = new
DystrybutoryOkienko(dystrybutory);
		frame.setVisible(true);
		frame.setBounds(new Rectangle(0, 0, 300, 300));
		frame.setLocationRelativeTo(null);
		return frame;
	}

	public static void main(String[] args) {
		odtworz(Dystrybutor.FAKE_DYSTRYBUTORY);
	}

}
```

Jest to tworzenie okienka związanego z zarządzaniem dystrybutorami.

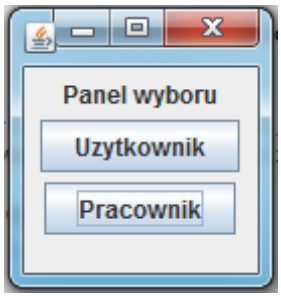


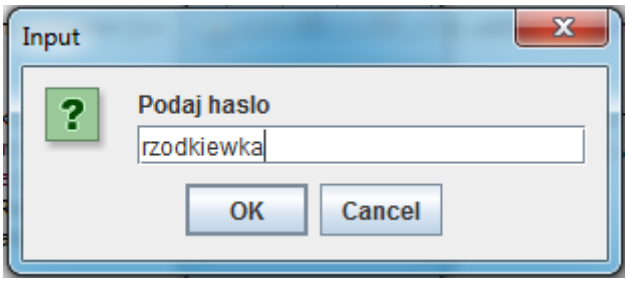
3.8. LogowanieOkienko

```
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class LogowanieOkienko extends JFrame {

	private JButton bUzytkownik;
	private JButton bPracownik;
	private final MenadzerSprzedazy menadzer;
	private final Zbiornik zbiornik;
	private final List<Dystrybutor> dystrybutory;

	public LogowanieOkienko(Zbiornik zbiornik, MenadzerSprzedazy
sprzedaz, List<Dystrybutor> dystrybutory) {
		this.zbiornik = zbiornik;
		this.menadzer = sprzedaz;
		this.dystrybutory = dystrybutory;

		stworzWyglad();
	}

	private void stworzWyglad() {
		JPanel panel = new JPanel();
		add(panel);
		panel.add(new JLabel("Panel wyboru"));
		panel.add(bUzytkownik = new JButton("Uzytkownik"));
		panel.add(bPracownik = new JButton("Pracownik"));

		bPracownik.addActionListener(ev -> {
			String haslo = JOptionPane.showInputDialog("Podaj
haslo");
			if("rzodkiewka".equals(haslo)){
				PracownikOkienko.odtworz(menadzer, zbiornik,
dystrybutory);
			}else{
				JOptionPane.showMessageDialog(null, "Bledne
haslo");
			}
		});
		bUzytkownik.addActionListener(ev ->
UzytkownikOkienko.odtworz(menadzer, zbiornik, dystrybutory));
	}

	public static LogowanieOkienko odtworz(Zbiornik zbiornik,
MenadzerSprzedazy sprzedaz, List<Dystrybutor> dystrybutory) {
		LogowanieOkienko frame = new LogowanieOkienko(zbiornik,
sprzedaz, dystrybutory);
		frame.setVisible(true);
		frame.setBounds(new Rectangle(0, 0, 100, 140));
		frame.setLocationRelativeTo(null);
		return frame;
```

```
	}

	public static void main(String[] args) {
		odtworz(Zbiornik.FAKE_ZBIORNIK,
MenadzerSprzedazy.FAKE_MENADZER, Dystrybutor.FAKE_DYSTRYBUTORY);
	}

}
```

Jest to tworzenie okienka związanego z logowaniem. Jest tu zawarte między innymi wpisywanie hasła pozwalającego na wejście do panelu pracownika.





3.9. MenadzerSprzedazy

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.List;

public class MenadzerSprzedazy {

	public static final MenadzerSprzedazy FAKE_MENADZER = new
MenadzerSprzedazy(Cennik.FAKE_CENNIK);
	static{
		FAKE_MENADZER.zamowienia.add(new
Zamowienie(Dystrybutor.FAKE_DYSTRYBUTORY.get(0), RodzajPaliwa.LPG, 100,
100));
		FAKE_MENADZER.zamowienia.add(new
Zamowienie(Dystrybutor.FAKE_DYSTRYBUTORY.get(0), RodzajPaliwa.ON, 120,
10));
		FAKE_MENADZER.zamowienia.add(new
Zamowienie(Dystrybutor.FAKE_DYSTRYBUTORY.get(0), RodzajPaliwa.PB95, 150,
20));
		FAKE_MENADZER.zamowienia.add(new
Zamowienie(Dystrybutor.FAKE_DYSTRYBUTORY.get(0), RodzajPaliwa.LPG, 160,
50));
```

```
                FAKE_MENADZER.zamowienia.add(new
Zamowienie(Dystrybutor.FAKE_DYSTRYBUTORY.get(0),    RodzajPaliwa.ON,    200,
70));
        }

        private final Cennik cennik;

        private final List<Zamowienie> zamowienia = new ArrayList<>();

        public MenadzerSprzedazy(Cennik cennik) {
                this.cennik = cennik;
        }

        public Zamowienie sprzedajIlosc(Dystrybutor d, RodzajPaliwa p, int
ilosc) {
                int cena = this.cennik.podajCene(p);
                int wartosc = ilosc * cena;
                Zamowienie z = new Zamowienie(d, p, wartosc, ilosc);
                zamowienia.add(z);
                return z;
        }

        public Zamowienie sprzedajWartosc(Dystrybutor d, RodzajPaliwa p,
int wartosc) {
                int cena = this.cennik.podajCene(p);
                int ilosc = wartosc / cena;
                Zamowienie z = new Zamowienie(d, p, wartosc, ilosc);
                zamowienia.add(z);
                return z;
        }

        public Cennik getCennik() {
                return cennik;
        }

        public List<Zamowienie> getZamowienia() {
                return Collections.unmodifiableList(zamowienia);
        }

        public void wycofajZamowienie(Zamowienie z){
                this.zamowienia.remove(z);
        }

}
```

Jest tu tworzony FAKE_MENADZER, który wykorzystałam do testowania programu w innych klasach. Mamy zmienne wchodzące w skład klasy MenadzerSprzedazy, jak również między innymi rozwiązany problem podawania ilości/wartości zamawianego paliwa.

3.10. PodstawoweOkienkoCena

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public abstract class PodstawoweOkienkoCena extends JFrame {
```

```
        protected Cennik cennik;

        public PodstawoweOkienkoCena(Cennik cennik) {
                this.cennik = cennik;

                stworzWyglad();
        }

        protected void stworzWyglad() {
                JPanel lista = new JPanel();
                for (RodzajPaliwa p : RodzajPaliwa.values()) {
                        JPanel panel = stworzWiersz(p);
                        lista.add(panel);
                }
                add(lista);
        }

        protected JPanel stworzWiersz(RodzajPaliwa p) {
                JPanel panel = new JPanel();
                panel.add(new JLabel(p.name()));
                panel.add(new JLabel(this.cennik.podajCene(p) + " PAU"));
                dodajAkcjeWWierszu(p, panel);
                return panel;
        }

        protected abstract void dodajAkcjeWWierszu(RodzajPaliwa p, JPanel
panel);
}
```

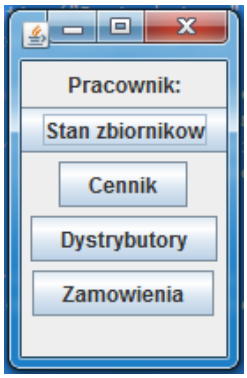
Jest tu tworzone podstawowe okienko cennika dziedziczące po klasie JFrame. Są tu utworzone funcie stworzWyglad() oraz stworzWiersz().

3.11.PracownikOkienko

```
import java.awt.Rectangle;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class PracownikOkienko extends JFrame {
        private MenadzerSprzedazy sprzedaz;
        private Zbiornik zbiornik;
        private List<Dystrybutor> dystrybutory;

        private JButton bSprawdz;
        private JButton bCeny;
        private JButton bDystrybutory;
        private JButton bZamowienia;

        public    PracownikOkienko(MenadzerSprzedazy    sprzedaz,    Zbiornik
zbiornik, List<Dystrybutor> dystrtbutory) {
                this.sprzedaz = sprzedaz;
                this.zbiornik = zbiornik;
                this.dystrybutory = dystrtbutory;

                stworzWyglad();
        }
```

```
        private void stworzWyglad() {
                JPanel panel = new JPanel();
                add(panel);
                panel.add(new JLabel("Pracownik:"));
                panel.add(bSprawdz = new JButton("Stan zbiornikow"));
                panel.add(bCeny = new JButton("Cennik"));
                panel.add(bDystrybutory = new JButton("Dystrybutory"));
                panel.add(bZamowienia = new JButton("Zamowienia"));

                bCeny.addActionListener(ev                                    ->
CennikOkienkoZKasowaniem.odtworz(sprzedaz.getCennik()));
                bSprawdz.addActionListener(ev                                 ->
ZbiornikOkienko.odtworz(zbiornik));
                bDystrybutory.addActionListener(ev                            ->
DystrybutoryOkienko.odtworz(dystrybutory));
                bZamowienia.addActionListener(ev                              ->
ZamowieniaOkienko.odtworz(sprzedaz));

        }

        public static PracownikOkienko odtworz(MenadzerSprzedazy menadzer,
Zbiornik fakezbiornik, List<Dystrybutor> dystrybutor) {
                PracownikOkienko frame = new PracownikOkienko(menadzer,
fakezbiornik,
                                dystrybutor);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 100, 300));
                frame.setLocationRelativeTo(null);
                return frame;
        }

        public static void main(String[] args) {
                MenadzerSprzedazy                menadzer                =
MenadzerSprzedazy.FAKE_MENADZER;
                Zbiornik fakezbiornik = Zbiornik.FAKE_ZBIORNIK;
                List<Dystrybutor>               dystrybutor             =
Dystrybutor.FAKE_DYSTRYBUTORY;
                odtworz(menadzer, fakezbiornik, dystrybutor);
        }


}
```

Jest tu tworzone okienko pracownika, czyli panel pojawiający się po zalogowaniu się jako pracownik.



3.12.RodzajPaliwa

```
public enum RodzajPaliwa {
        PB98, PB95, ON, LPG
}
```

Następuje tu utworzenie typu enum. Pozwala on na definiowane wybranego zbioru możliwych wartości. Tu: zbioru rodzajów paliwa.

3.13.StacjaPaliw

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class StacjaPaliw {

        public static void main(String[] args) {

                Cennik cennik = new Cennik();
                CennikZapisOdczyt       cennik_zapis_odczyt       =       new
CennikZapisOdczyt(cennik);
                cennik_zapis_odczyt.load();
                cennik.addListener(() -> cennik_zapis_odczyt.save());

                Zbiornik zbiornik = new Zbiornik();
                MenadzerSprzedazy sprzedaz = new MenadzerSprzedazy(cennik);
                List<Dystrybutor> dystrybutory = Arrays.asList(//
                                new Dystrybutor("Prawy", zbiornik, sprzedaz),
//
                                new Dystrybutor("Lewo", zbiornik, sprzedaz)//
                );
                LogowanieOkienko.odtworz(zbiornik, sprzedaz, dystrybutory);
        }
}
```

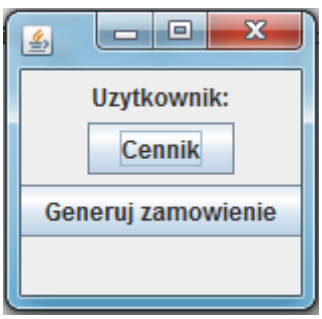
Znajduje się tu główny main programu oraz utworzenie poszczególnych elementów. Wywoływane jest także okienko logowania.

3.14.UzytkownikOkienko

```
import java.awt.Rectangle;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class UzytkownikOkienko extends JFrame {
        private MenadzerSprzedazy sprzedaz;
        private Zbiornik zbiornik;
        private List<Dystrybutor> dystrybutory;

        private JButton bCennik;
```

```
        private JButton bGenerujZamowienie;

        public   UzytkownikOkienko(MenadzerSprzedazy   sprzedaz,   Zbiornik
zbiornik, List<Dystrybutor> dystrtbutory) {
                this.sprzedaz = sprzedaz;
                this.zbiornik = zbiornik;
                this.dystrybutory = dystrtbutory;

                stworzWyglad();
        }

        private void stworzWyglad() {
                JPanel panel = new JPanel();
                add(panel);
                panel.add(new JLabel("Uzytkownik:"));
                panel.add(bCennik = new JButton("Cennik"));
                panel.add(bGenerujZamowienie    =    new    JButton("Generuj
zamowienie"));

                bCennik.addActionListener(ev                                ->
CennikOkienko.odtworz(sprzedaz.getCennik()));
                bGenerujZamowienie.addActionListener(ev                     ->
ZamowienieOkienko.odtworz(sprzedaz,dystrybutory));

        }

        public static UzytkownikOkienko odtworz(MenadzerSprzedazy menadzer,
Zbiornik zbiornik, List<Dystrybutor> dystrybutory) {
                UzytkownikOkienko  frame  =  new  UzytkownikOkienko(menadzer,
zbiornik,
                                dystrybutory);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 150, 300));
                frame.setLocationRelativeTo(null);
                return frame;
        }

        public static void main(String[] args) {
                MenadzerSprzedazy                menadzer                =
MenadzerSprzedazy.FAKE_MENADZER;
                Zbiornik fakezbiornik = Zbiornik.FAKE_ZBIORNIK;
                List<Dystrybutor>               dystrybutor               =
Dystrybutor.FAKE_DYSTRYBUTORY;
                odtworz(menadzer, fakezbiornik, dystrybutor);
        }


}
```

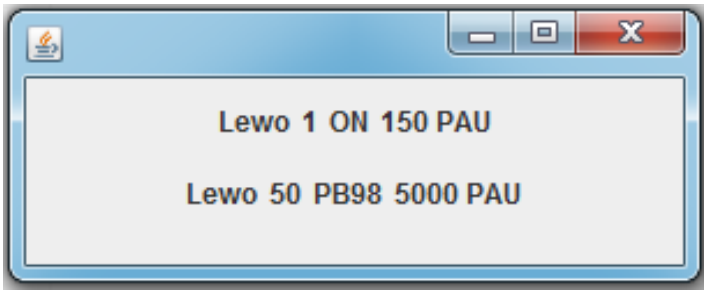
Jest tu tworzone okienko zwykłego użytkownika.



3.15.ZamowieniaOkienko

```
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class ZamowieniaOkienko extends JFrame {

	private MenadzerSprzedazy sprzedaz;

	public ZamowieniaOkienko(MenadzerSprzedazy sprzedaz) {
		this.sprzedaz = sprzedaz;

		JPanel lista = new JPanel();
		for (Zamowienie p : sprzedaz.getZamowienia()) {
			JPanel panel = new JPanel();
			panel.add(new JLabel(p.getDystrybutor().getName()));
			panel.add(new JLabel(p.getIlosc() + ""));
			panel.add(new JLabel(p.getPaliwo().name()));
			panel.add(new JLabel(p.getWartosc() + " PAU"));

			lista.add(panel);
		}
		add(lista);
	}

	public static ZamowieniaOkienko odtworz(MenadzerSprzedazy sprzedaz)
{
		ZamowieniaOkienko frame = new ZamowieniaOkienko(sprzedaz);
		frame.setVisible(true);
		frame.setBounds(new Rectangle(0, 0, 300, 300));
		frame.setLocationRelativeTo(null);
		return frame;
	}

	public static void main(String[] args) {
		odtworz(MenadzerSprzedazy.FAKE_MENADZER);
	}

}
```

Jest tu tworzone okienko wyświetlania pracownikowi listy ostatnich zamówień.



3.16.Zamowienie

```
public class Zamowienie {
	private final Dystrybutor dystrybutor;
	private final RodzajPaliwa paliwo;
	private final int wartosc;
	private final int ilosc;

	public Zamowienie(Dystrybutor dystrybutor, RodzajPaliwa paliwo, int
wartosc, int ilosc) {
		this.dystrybutor = dystrybutor;
		this.paliwo = paliwo;
		this.wartosc = wartosc;
		this.ilosc = ilosc;
	}

	public Dystrybutor getDystrybutor() {
		return dystrybutor;
	}

	public RodzajPaliwa getPaliwo() {
		return paliwo;
	}

	public int getWartosc() {
		return wartosc;
	}

	public int getIlosc() {
		return ilosc;
	}
}
```

Jest to klasa, w której następuje pobranie danych wpisywanych przez użytkownika do okienka zamówienia.

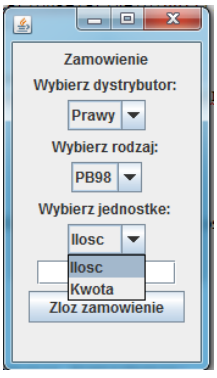
3.17.ZamowienieOkienko

```
import java.awt.HeadlessException;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class ZamowienieOkienko extends JFrame {

	private final MenadzerSprzedazy sprzedaz;
	private final List<Dystrybutor> dystrybutory;
	private JComboBox<Dystrybutor> comboDystrybutory;
	private JComboBox<RodzajPaliwa> comboRodzaje;
```

```
        private JComboBox<String> comboJednostka;
        private JButton bZloz;
        private JTextField bWartosc;

        public          ZamowienieOkienko(MenadzerSprzedazy          sprzedaz,
List<Dystrybutor> dystrybutory) throws HeadlessException {
                super();
                this.sprzedaz = sprzedaz;
                this.dystrybutory = dystrybutory;

                JPanel panel;
                add(panel = new JPanel());
                panel.add(new JLabel("Zamowienie"));

                panel.add(new JLabel("Wybierz dystrybutor: "));
                comboDystrybutory = new JComboBox<>(dystrybutory.toArray(new
Dystrybutor[dystrybutory.size()]));
                panel.add(comboDystrybutory);

                panel.add(new JLabel("Wybierz rodzaj: "));
                comboRodzaje = new JComboBox<>(RodzajPaliwa.values());
                panel.add(comboRodzaje);

                panel.add(new JLabel("Wybierz jednostke: "));
                comboJednostka   =   new   JComboBox<>(new   String[]{"Ilosc",
"Kwota"});
                panel.add(comboJednostka);

                panel.add(bWartosc = new JTextField(10));

                panel.add(bZloz = new JButton("Zloz zamowienie"));
                bZloz.addActionListener(ev -> {
                        Dystrybutor      dystrybutor      =      (Dystrybutor)
comboDystrybutory.getSelectedItem();
                        RodzajPaliwa       rodzaj       =       (RodzajPaliwa)
comboRodzaje.getSelectedItem();
                        String           akcja           =           (String)
comboJednostka.getSelectedItem();

                        if(!dystrybutor.isWlaczony()){
                                JOptionPane.showMessageDialog(null,
"Dystrybutor wylaczony");
                                return;
                        }

                        Zamowienie zamowienie = null;
                        try{
                                int                 wartosc                 =
Integer.parseInt(bWartosc.getText());

                                if("Ilosc".equals(akcja)){
                                        zamowienie                          =
sprzedaz.sprzedajIlosc(dystrybutor, rodzaj, wartosc);
                                }else{
                                        zamowienie                          =
sprzedaz.sprzedajWartosc(dystrybutor, rodzaj, wartosc);
                                }
                                dystrybutor.nalej(rodzaj,
zamowienie.getIlosc());
                                int wartosc_z = zamowienie.getWartosc();
```

```
                                while(wartosc_z > 0){
                                        int                 a                 =
Integer.parseInt(JOptionPane.showInputDialog("Do    zaplaty    jest:    "    +
wartosc z + " PAU"));
                                        wartosc_z -= a;
                                }
                                if(wartosc_z != 0){
                                        JOptionPane.showMessageDialog(null,
"Reszty zostalo: " + -1 * wartosc_z + " PAU");
                                }

                        }catch(NumberFormatException ex){
                                JOptionPane.showMessageDialog(null,
"Nieprawidlowa liczba");
                                return;
                        } catch (BrakPaliwaException e) {
                                JOptionPane.showMessageDialog(null,      "Brak
paliwa");
                                sprzedaz.wycofajZamowienie(zamowienie);
                                return;
                        }

                        JOptionPane.showMessageDialog(null,       "Zamowienie
zapisano.");


                });

        }

        public static ZamowienieOkienko odtworz(MenadzerSprzedazy sprzedaz,
List<Dystrybutor> dystrybutory) {

                ZamowienieOkienko  frame  =  new  ZamowienieOkienko(sprzedaz,
dystrybutory);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 170, 300));
                frame.setLocationRelativeTo(null);
                return frame;
        }
}
```

Tu odbywa się generowanie formularza zamówienia, uzupełnianie go, obsługa wyjątków, złożenie zamówienia, obliczenie należnej kwoty, przyjęcie zapłaty, obliczenie i wydanie reszty oraz wykonanie usługi.



3.18.Zbiornik

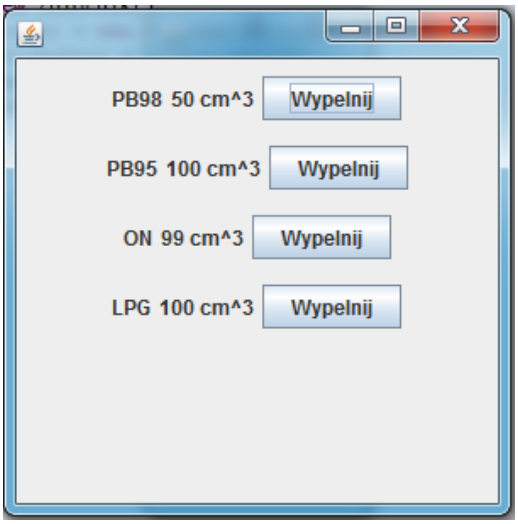
```
import java.util.HashMap;
import java.util.Map;

public class Zbiornik {
	private Map<RodzajPaliwa, Integer> stany = new HashMap<>();

	public static final Zbiornik FAKE_ZBIORNIK = new Zbiornik(){
	};

	public Zbiornik() {
		for (RodzajPaliwa p : RodzajPaliwa.values()) {
			stany.put(p, 100);
		}
	}
	public boolean jestPaliwo(RodzajPaliwa p){
		return this.podajStan(p) > 0;
	}

	public int podajStan(RodzajPaliwa p){
		return stany.get(p);
	}

	public void ustawStan(RodzajPaliwa p, int w) throws
BrakPaliwaException{
		if(w <= 0){
			throw new BrakPaliwaException();
		}

		stany.put(p, w);
	}

	public void zmienStan(RodzajPaliwa p, int w) throws
BrakPaliwaException{
		int s = this.podajStan(p);
		s += w;
		ustawStan(p, s);
	}
}
```

Tu następuje tworzenie klasy zbiornik a także charakterystyka wyjątku BrakPaliwaException. Jest tu także zawarta usługa napełniania zbiornika przez pracownika Stacji.

3.19.ZbiornikOkienko

```
import java.awt.Rectangle;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class ZbiornikOkienko extends JFrame{

	private Zbiornik zbiornik;

	public ZbiornikOkienko(Zbiornik zbiornik) {
```

```
                this.zbiornik = zbiornik;

                JPanel lista = new JPanel();
                for(RodzajPaliwa p: RodzajPaliwa.values()){
                        JPanel panel = new JPanel();
                        panel.add(new JLabel(p.name()));
                        panel.add(new  JLabel(this.zbiornik.podajStan(p)  +  "
cm^3"));
                        JButton bWypelnij;
                        panel.add(bWypelnij = new JButton("Wypelnij"));
                        bWypelnij.addActionListener(e -> {
                                try {
                                        zbiornik.ustawStan(p, 1000);
                                } catch (BrakPaliwaException e1) {
                                        e1.printStackTrace();
                                }
                                setVisible(false);
                                dispose();
                        });
                        lista.add(panel);
                }
                add(lista);
        }

        public static ZbiornikOkienko odtworz(Zbiornik z) {
                ZbiornikOkienko frame = new ZbiornikOkienko (z);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 300, 300));
                frame.setLocationRelativeTo(null);
                return frame;
        }

        public static void main(String[] args) {
                JFrame frame= new ZbiornikOkienko(Zbiornik.FAKE_ZBIORNIK);
                frame.setVisible(true);
                frame.setBounds(new Rectangle(0, 0, 100, 300));
                frame.setLocationRelativeTo(null);
        }

}
```



Tu następuje tworzenie okienka Zbiornik dostępnego z panelu pracownika w celu uzupełnienia stanu paliwa.

4. INSTRUKCJA DLA UŻYTKOWNIKA

1. Uruchom program oraz wybierz opcję „użytkownik”.
2. Jeśli chcesz sprawdzić aktualne ceny paliw, wybierz cennik.
3. Jeśli chcesz zatankować samochód, złóż zamówienie. W tym celu uzupełnij formularz, zatwierdź go, jeśli nie będzie przeszkód technicznych zapłać za usługę, odbierz resztę i ciesz się zatankowanym samochodem! ☺

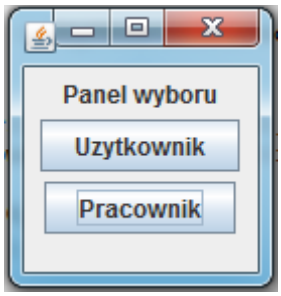
4. INSTRUKCJA DLA PRACOWNIKA

1. Uruchom program, wybierz opcję „pracownik” i wpisz hasło „rzodkiewka”.
2. Jeśli chcesz sprawdzić aktualny poziom paliwa w zbiorniku bądź uzupełnić zbiornik, wybierz Stan Zbiorników.
3. Jeśli chcesz sprawdzić aktualne ceny paliw i/lub je edytować, wybierz Cennik.
4. Jeśli chcesz sprawdzić włączyć/wyłączyć dystrybutor, wybierz Dystrybutory.
5. Jeśli chcesz zobaczyć ostatnie zamówienia, wybierz Zamówienia.

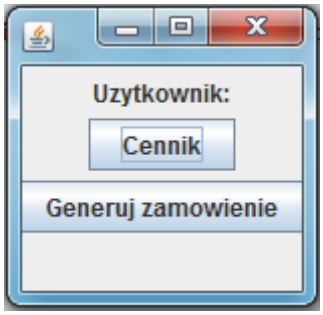
5. PRZYKŁADOWE UŻYCIE

5.1. UŻYTKOWNIK

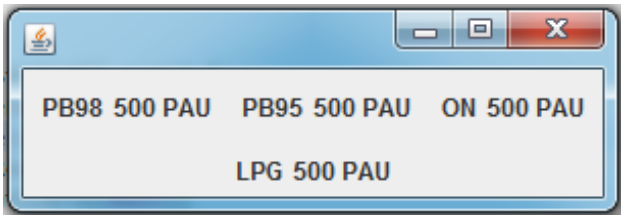
1. Wskazanie użytkownika.



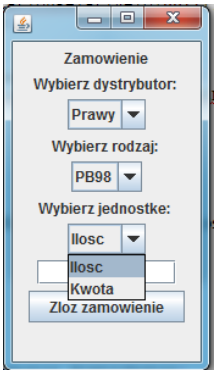
2. Uruchamia się menu użytkownika.



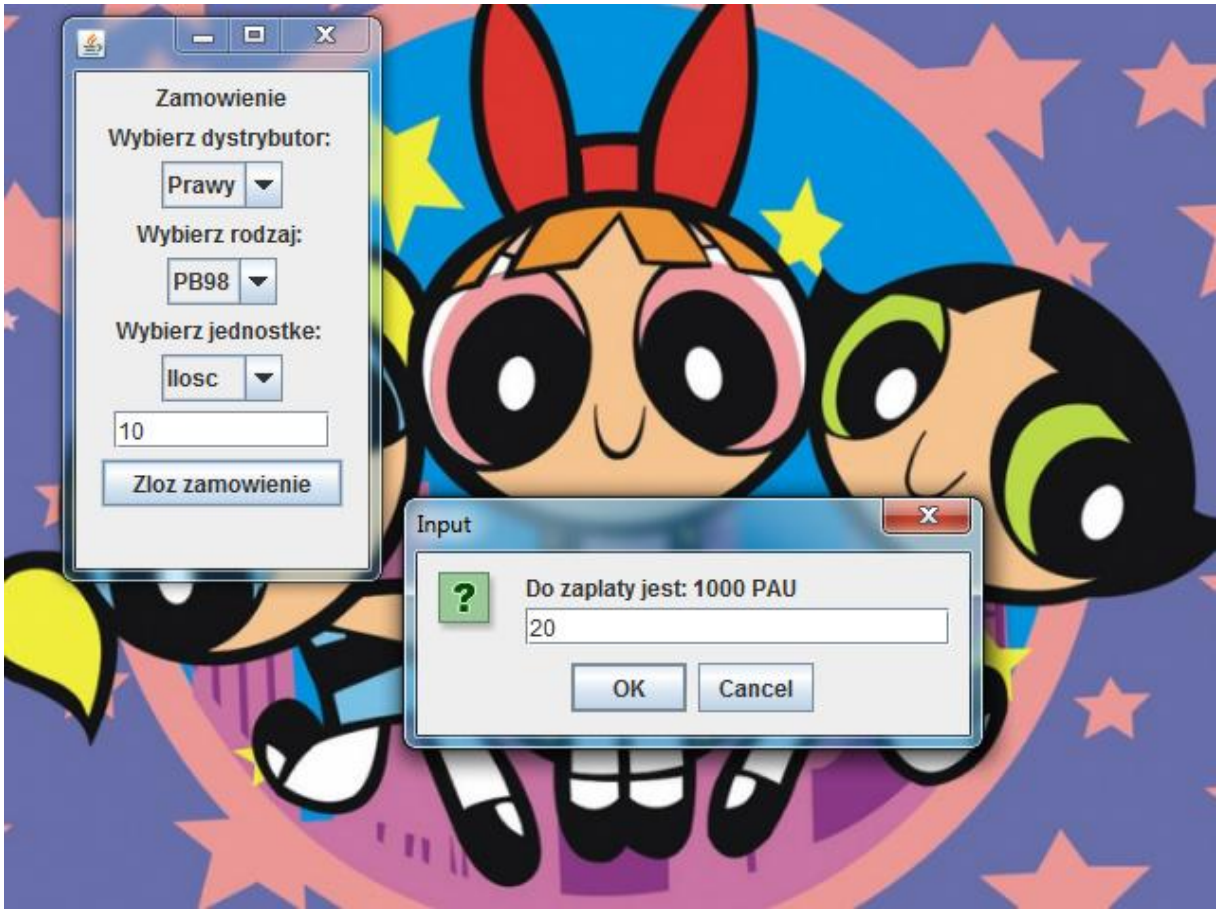
3. Sprawdzenie cen paliw.



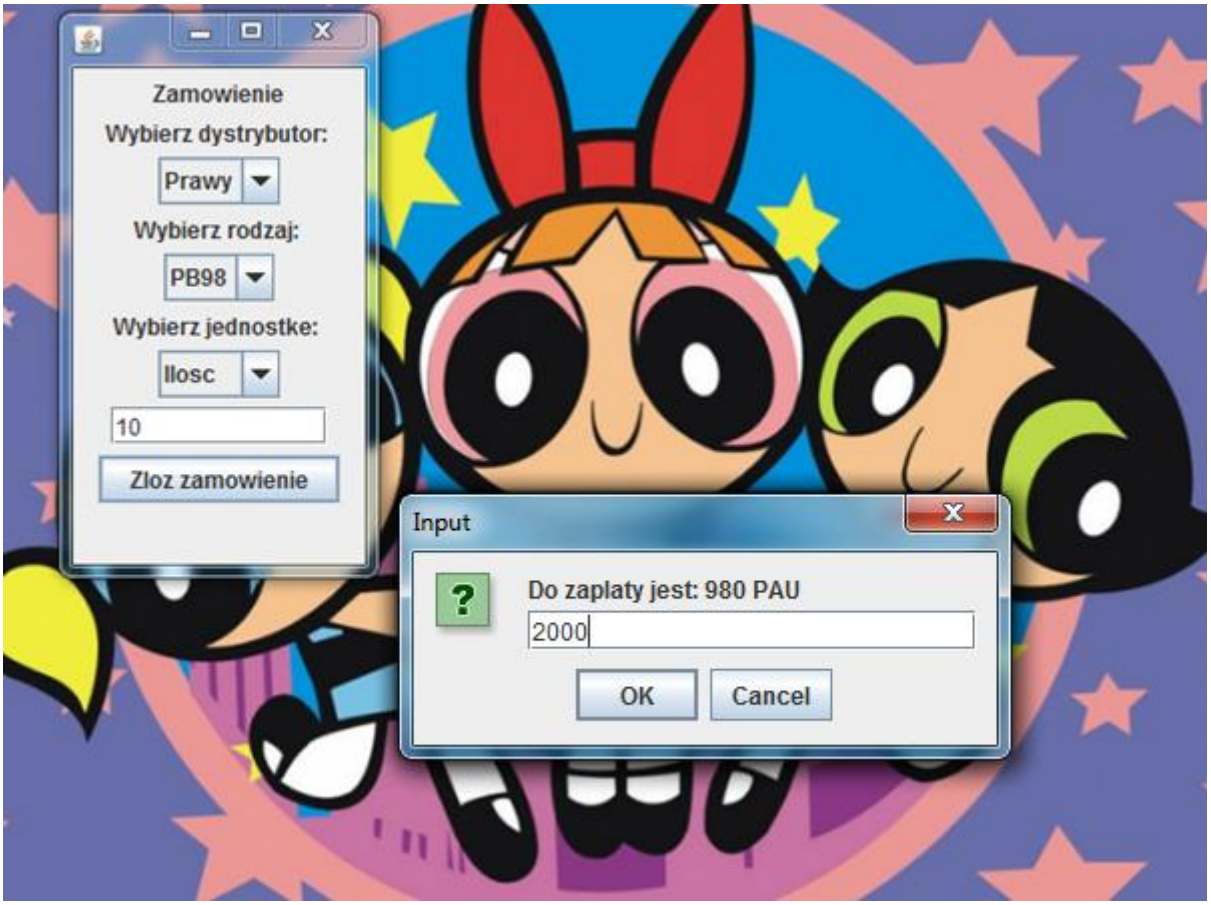
4. Generowanie zamówienia.



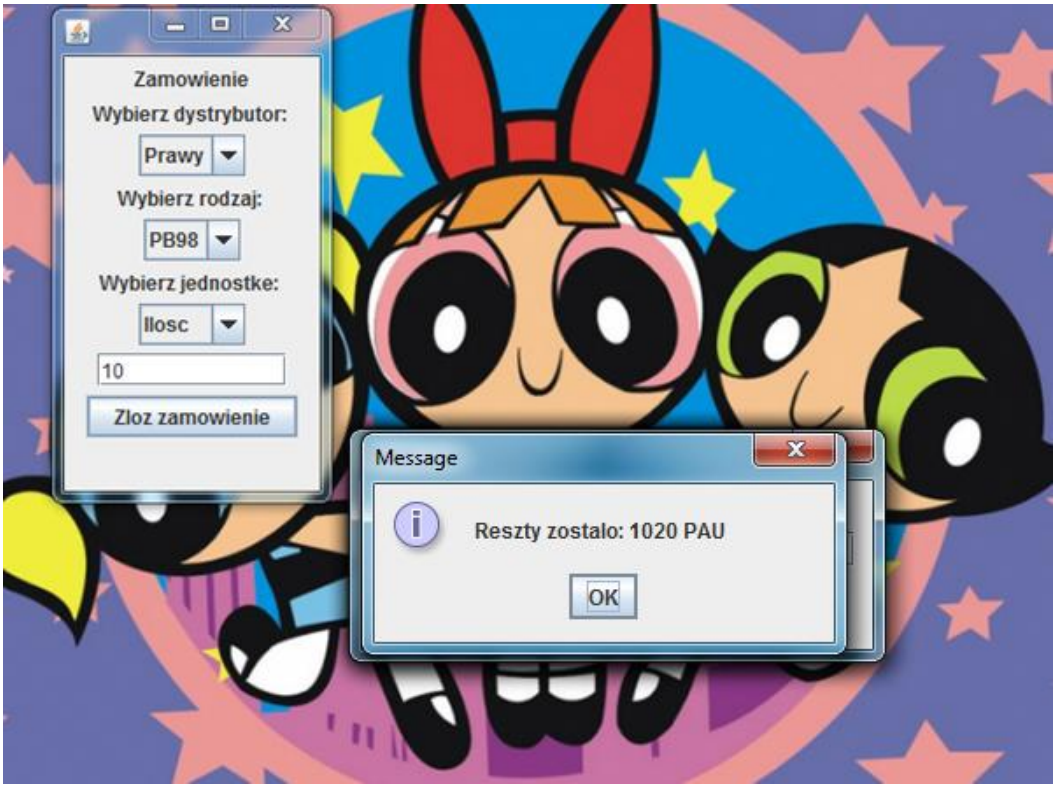
5. Wyświetlenie ceny i oczekiwanie na zapłatę.



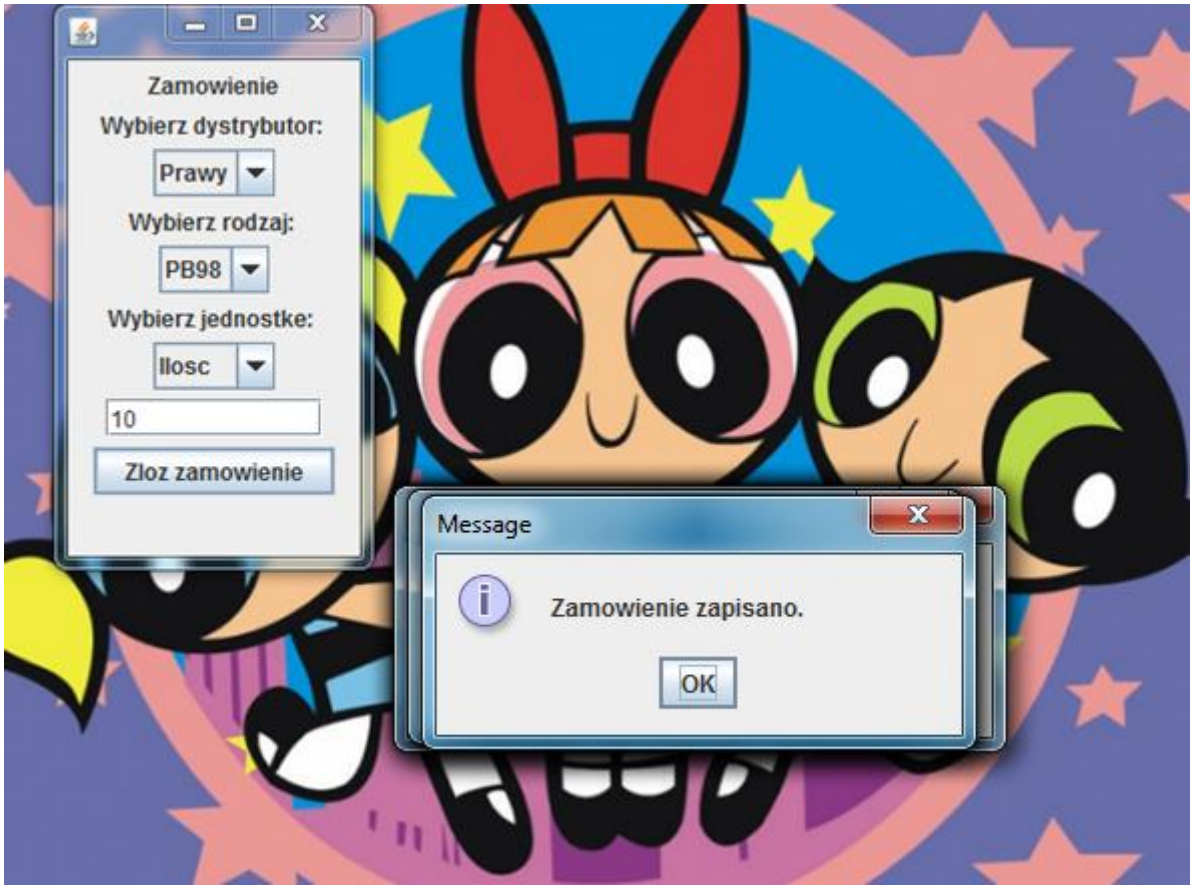
6. Jeśli zapłacimy za mało, program oczekuje pozostałej zapłaty.



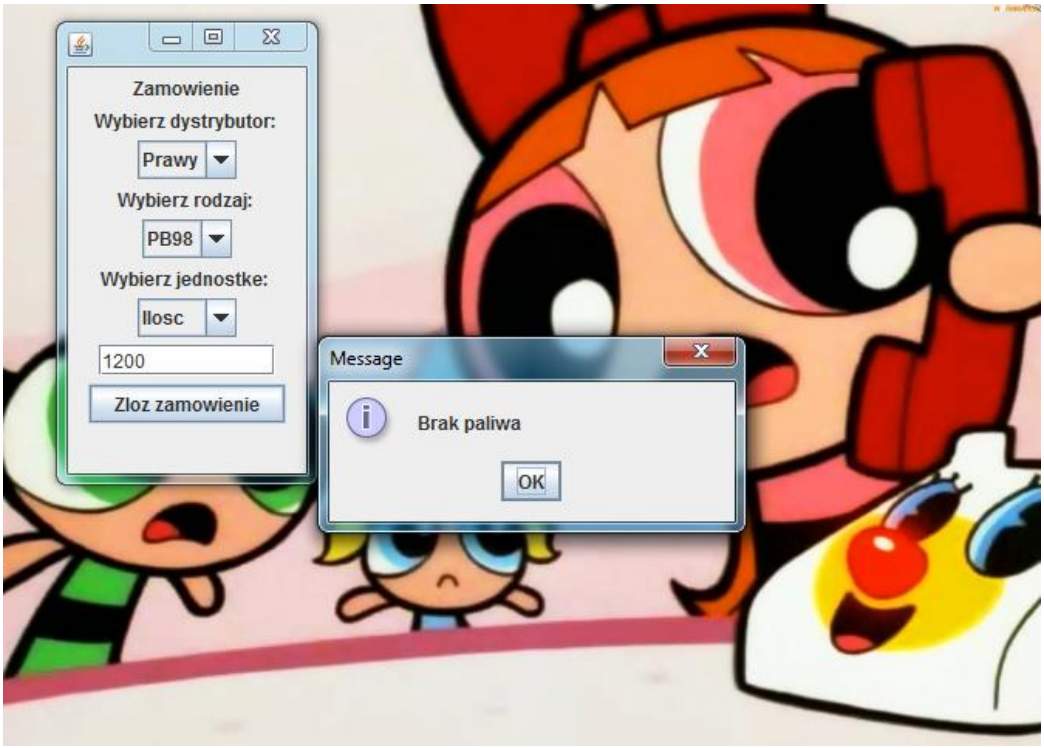
7. Jeśli zapłacimy więcej, Stacja zwróci resztę.



8. Zamówienie zostaje zapisane i zrealizowane.

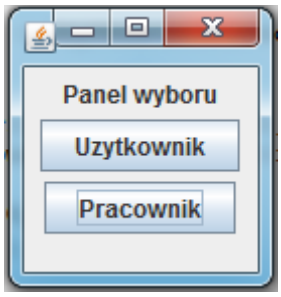


9. Czasem zdarzy się tak, że nie ma wystarczającej ilości paliwa, bądź dystrybutor jest uszkodzony. Wtedy zostaje wyświetlony odpowiedni komunikat.

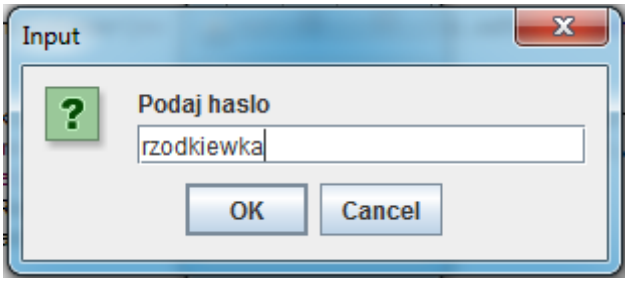


5.2. PRACOWNIK

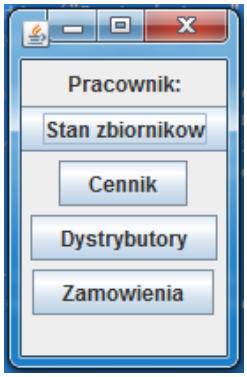
1. Wskazanie pracownika.



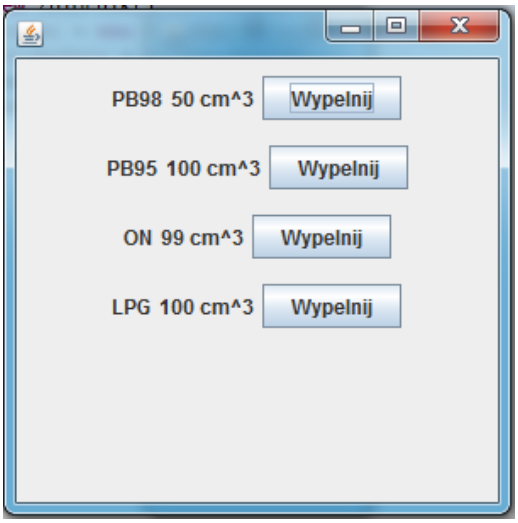
2. Logowanie.



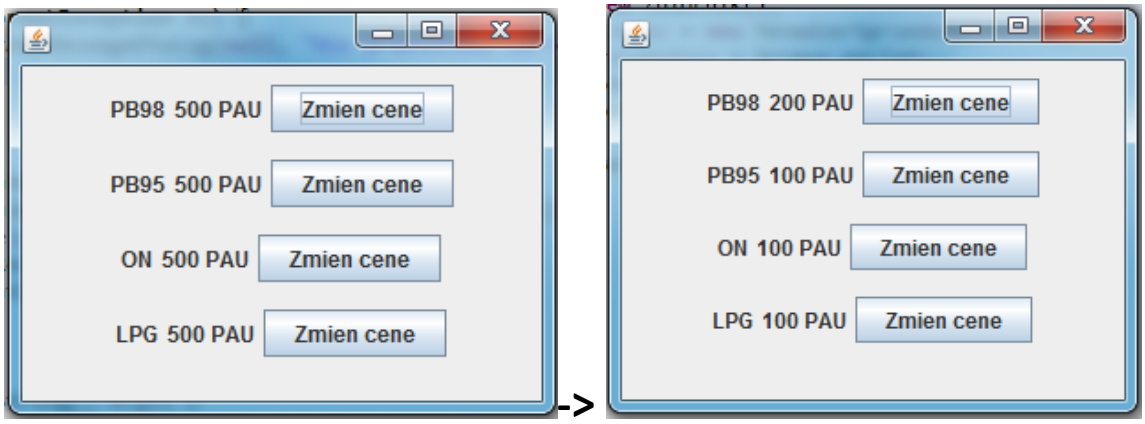
2. Uruchamia się menu pracownika.



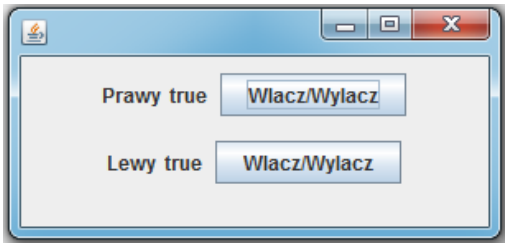
3. Sprawdzenie stanu zbiorników oraz możliwość ich uzupełnienia.



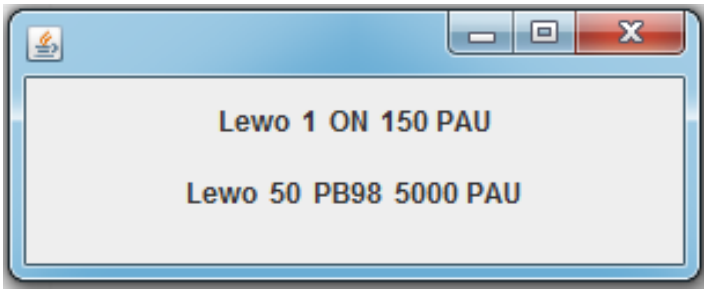
4. Sprawdzenie cennika oraz możliwość zmiana cen.



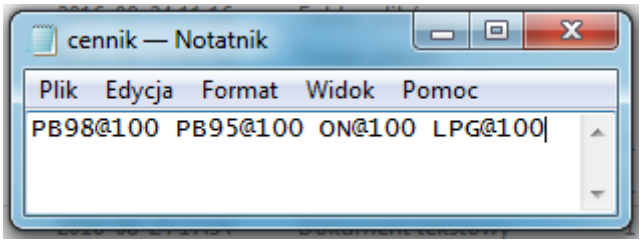
5. Możliwość włączenia/wyłączenia dystrybutorów oraz sprawdzenia ich stanu.



6. Sprawdzenie listy ostatnich zamówień.



7. Każda zmiana ceny jest zapisywana do zewnętrznego pliku tekstowego.



6. NARZĘDZIA

W rozwiązaniu wykorzystałam możliwości środowiska programistycznego *Eclipse* na systemie operacyjnym Windows.

7. DIAGRAM KLAS UML

8. MATERIAŁY WYKORZYSTANE PODCZAS REALIZACJI PROJEKTU

1. http://anna.ii.uph.edu.pl/PrO
2. http://www.jareks.ii.uph.edu.pl/po/
3. http://javastart.pl/

Autor:

Paulina Czubaj
Informatyka I rok
Uniwersytet Przyrodniczo – Humanistyczny w Siedlcach
2016