

Instrukcja 7

Laboratoria 9, 10

Opracowanie diagramów sekwencji dla wybranych przypadków użycia reprezentujących usługi oprogramowania wynikających również z wykonanych diagramów czynności; definicja operacji klas na podstawie diagramów sekwencji w języku Java. Zastosowanie projektowych wzorców zachowania.

Cel laboratorium:

Definiowanie w sposób iteracyjno - rozwojowy modelu projektowego programowania ([wykład 1](#)) opartego na:

- **Modelowaniu logiki biznesowej reprezentowanej przez wybrany przypadek użycia za pomocą diagramów sekwencji po wykonaniu pierwszego przypadku użycia podczas laboratorium 8, stanowiącego bazową logikę biznesową, z której korzystają kolejne przypadki użycia. Należy definiować operacje i atrybuty kolejnej klasy (dziedziczenie, powiązania i agregacje) na diagramie klas zidentyfikowanej w wyniku modelowania kolejnego przypadku użycia i wykonanie scenariusza tego przypadku użycia za pomocą diagramu sekwencji.**
 - **Implementacja modelu projektowego wybranego przypadku użycia za pomocą języka Java SE – rozszerzanie kodu źródłowego programu wykonanego podczas laboratoriów 7 i 8.**
1. Zdefiniować kolejne diagramy sekwencji operacji reprezentujących scenariusze poszczególnych przypadków użycia umieszczając je w projekcie UML założonym podczas realizacji instrukcji 2 i uzupełnianym podczas realizacji instrukcji 3-6.
 2. Należy automatycznie uzupełniać definicję klas podczas modelowania kolejnych operacji za pomocą diagramów sekwencji. Należy rozwijać diagram klas utworzony podczas realizacji instrukcji 5 i 6.
 3. Podzielić ten proces modelowania na kilka iteracji. Należy wykonać kolejne przypadki użycia, których wyniki wspierają działanie kolejnego modelowanego przypadku użycia w kolejnej iteracji ([wykład4](#), **Dodatek 1 instrukcji**). Pierwszy wykryty przypadek użycia należy modelować w 1-ej iteracji procesu projektowania (podczas realizacji instrukcji 6). Podobnie należy wybierać kolejne przypadki użycia do kolejnych iteracji.
 4. Należy systematycznie uzupełniać kod programu typu **Java Class Library** w projekcie założonym podczas realizacji instrukcji 5 i 6.
 5. Informacje niezbędne do modelowania oprogramowania za pomocą klas i sekwencji (tworzenia modelu projektowego) z wykorzystaniem wzorców projektowych podane zostały w wykładach: [wykład 3](#), [wykład4](#), [wykład 5-część 1](#), [wykład5-część2](#).

Uwaga:

Notacje stosowane na diagramie klas i sekwencji w Dodatku 1, odpowiadające składni składowych klas w języku Java, różnią się od notacji przedstawionych w instrukcji 1, prezentujących diagramy wykonane w środowisku VP CE. Oczywiście, w realizowanym projekcie w ramach laboratoriów należy zastosować odpowiadające notacje proponowane w środowisku VP CE.

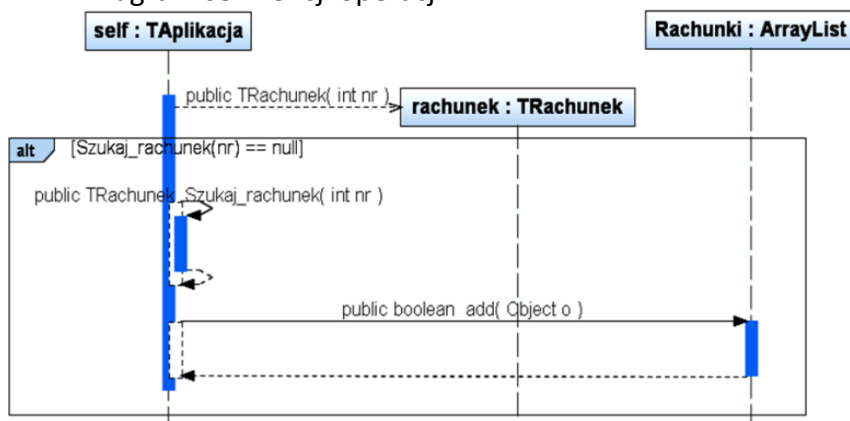
Dodatek 1

Przykład modelowania i implementacji przypadków użycia za pomocą diagramów sekwencji oraz diagramów klas i pakietów po wykonaniu bazowego przypadku użycia. Zastosowanie projektowych wzorców strukturalnych, wytwórczych i czynnościowych (cd. z instrukcji 2 - 5).

2-a iteracja: modelowanie przypadku użycia **PU Wstawianie nowego rachunku**

1. Modelowanie i implementacja operacji **void Wstaw_rachunek(int nr)** w klasie **TAplikacja**.

1.1. Diagram sekwencji operacji:



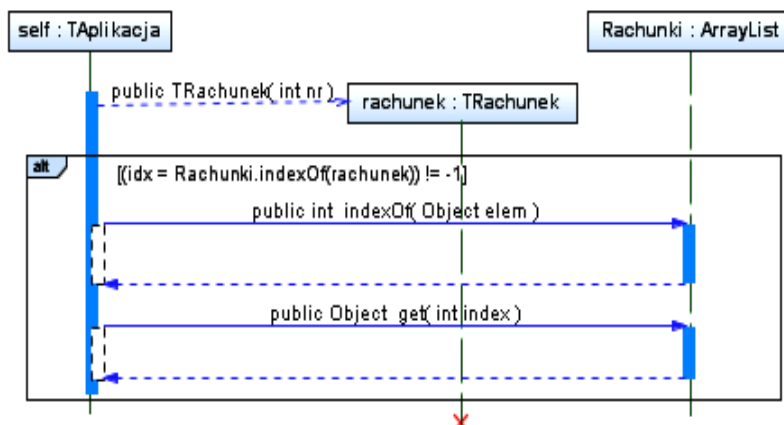
1.2. Kod operacji:

```

public void Wstaw_rachunek(int nr) {
    TRachunek rachunek = new TRachunek(nr);
    if (Szukaj_rachunek(nr) == null) {
        Rachunki.add(rachunek);
    }
}
  
```

2. Modelowanie i implementacja operacji **TRachunek Szukaj_rachunek (int nr)** z klasy **TAplikacja** (modelownie **PU Szukanie Rachunku**).

2.1. Diagram sekwencji operacji:



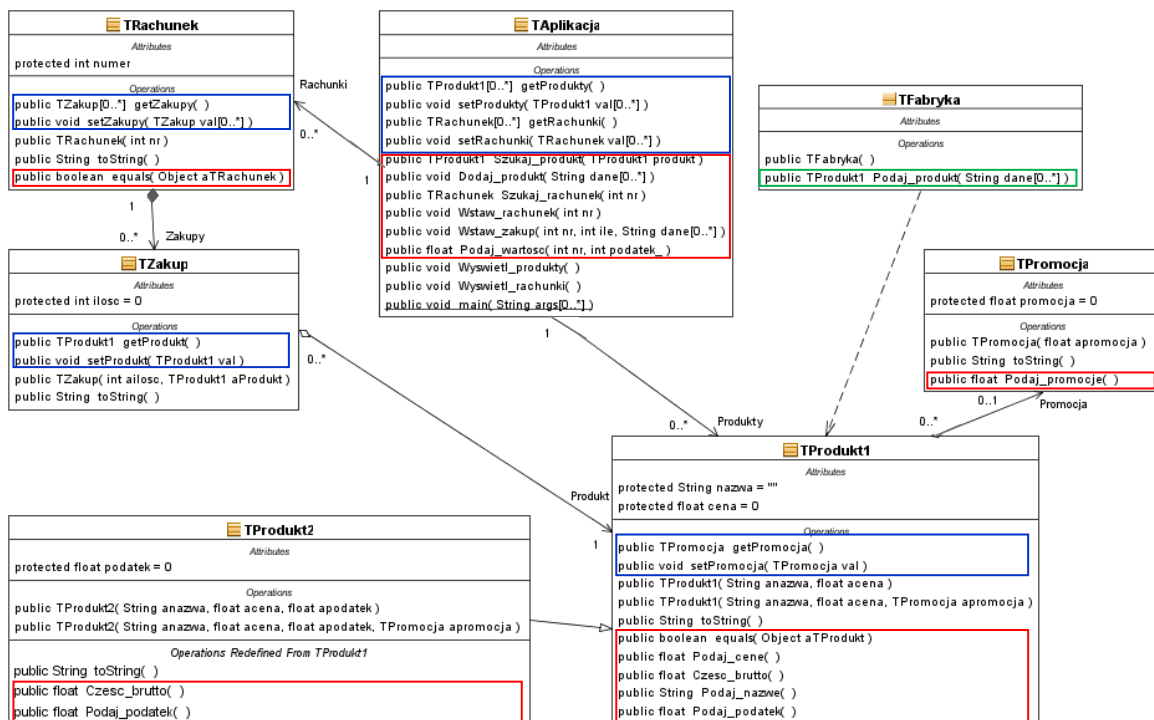
2.2. Kod operacji:

```
public TRachunek Szukaj_rachunek (int nr) {
    TRachunek rachunek = new TRachunek(nr);
    int idx;
    if ((idx=Rachunki.indexOf(rachunek)) != -1) {
        rachunek=Rachunki.get(idx);
        return rachunek;
    }
    return null;
}
```

2.3. Kod operacji boolean `equals(Object aTRachunek)` w klasie `TRachunek`, wywoływanej w metodzie `indexOf` obiektu `Rachunki` typu `ArrayList`:

```
@Override
public boolean equals (Object aTRachunek) {
    TRachunek rachunek= (TRachunek)aTRachunek;
    boolean bStatus = true;
    if ( numer!= rachunek.numer )
        bStatus = false;
    return bStatus;
}
```

3. Diagram klas zawierający elementy wynikające z wykonanych diagramów sekwencji w 2-iteracji

4. Rozszerzenie kodu źródłowego klas, dodanego do kodu wykonanego na podstawie wykonanego diagramu klas i diagramów sekwencji („inżynieria wprost”) – czyli dodanie pomocniczych metod do prezentacji wyników metod logiki biznesowej modelowanych za pomocą diagramów sekwencji. Prezentacja wyników działania kodu z 1-ej iteracji oraz kodu z 2-iteracji, gdzie wyświetla się zawartość pustych rachunków (obiektów typu `TRachunek`).

Klasa TRachunek

```

public TRachunek(int nr) {
    numer = nr; }
@Override
public String toString() {
    TZakup z;
    StringBuilder sb = new StringBuilder();
    sb.append(" Rachunek : ");
    sb.append(numer).append("\n");
    for (TZakup zakup:Zakupy)
        sb.append(zakup.toString()).append("\n");
    return sb.toString();
}

```

Klasa TAplikcja

```

public void Wyswietl_rachunki() {
    for (TRachunek rachunek: Rachunki)
        System.out.println(rachunek.toString());
}

```

```

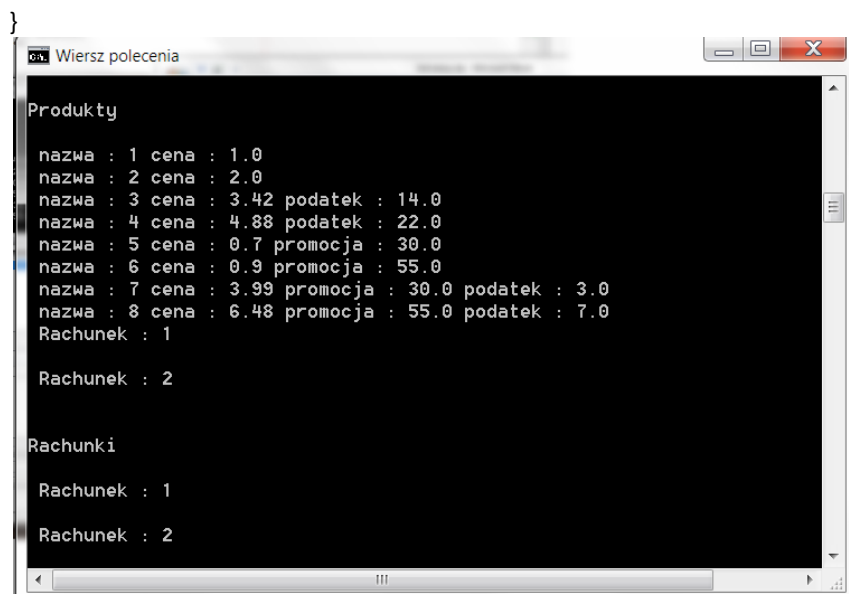
public static void main(String args[]) {
    TAplikcja app = new TAplikcja();
    String dane1[] = {"0", "1", "1"};
    String dane2[] = {"0", "2", "2"};
    app.Dodaj_produkt(dane1);
    app.Dodaj_produkt(dane2);
    app.Dodaj_produkt(dane1);
    String dane3[] = {"2", "3", "3", "14"};
    String dane4[] = {"2", "4", "4", "22"};
    app.Dodaj_produkt(dane3);
    app.Dodaj_produkt(dane4);
    app.Dodaj_produkt(dane3);
    String dane5[] = {"1", "5", "1", "30"};
    String dane6[] = {"1", "6", "2", "50"};
    String dane7[] = {"3", "7", "3", "3", "30"};
    String dane8[] = {"3", "8", "4", "7", "50"};
    app.Dodaj_produkt(dane5);
    app.Dodaj_produkt(dane6);

```

```

    app.Dodaj_produkt(dane5);
    app.Dodaj_produkt(dane7);
    app.Dodaj_produkt(dane8);
    app.Dodaj_produkt(dane7);
    System.out.println("\nProdukty\n");
    app.Wyswietl_produkty();
    app.Wstaw_rachunek(1);
    app.Wstaw_rachunek(1);
    app.Wstaw_rachunek(2);
    app.Wyswietl_rachunki();
    System.out.println("\nRachunki\n");
    TRachunek rachunek;
    if ((rachunek = app.Szukaj_rachunek(1)) != null)
        System.out.println(rachunek.toString());
    if ((rachunek = app.Szukaj_rachunek(2)) != null)
        System.out.println(rachunek.toString())

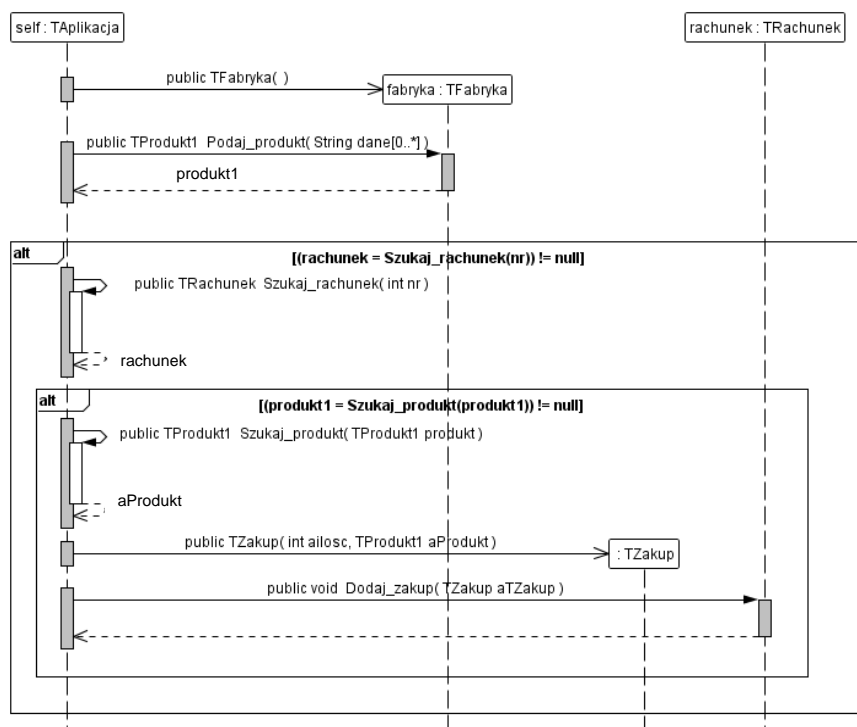
```



3-a iteracja: modelowanie przypadku użycia **PU Wstawianie nowego zakupu**

1. Modelowanie i implementacja operacji **void Wstaw_zakup(int nr, int ile, String dane[])** w klasie **TAplikacja**.

1.1. Diagram sekwencji operacji:



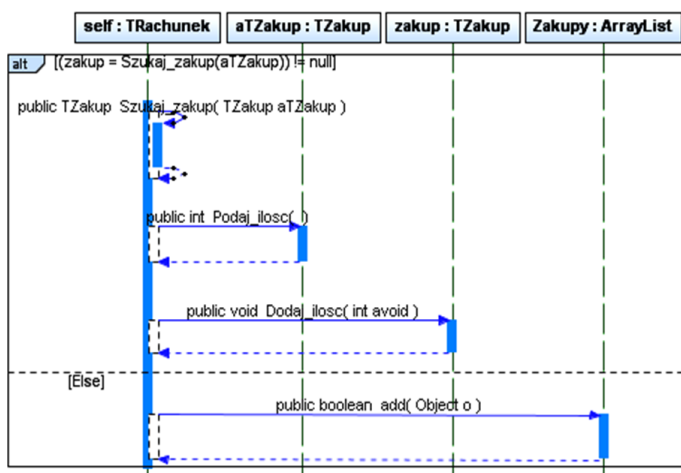
1.2. Kod operacji:

```

public void Wstaw_zakup(int nr, int ile, String dane[]) {
    TRachunek rachunek;
    TFabryka fabryka = new TFabryka();
    TProdukt1 produkt1 = fabryka.Podaj_produkt(dane);
    if ((rachunek = Szukaj_rachunek(nr)) != null)
        if ((produkt1 = Szukaj_produkt(produkt1)) != null)
            rachunek.Dodaj_zakup(new TZakup(ile, produkt1);
        }
}
  
```

2. Modelowanie i implementacja operacji **void Dodaj_zakup(TZakup aTZakup)** z klasy **TRachunek**.

2.1. Diagram sekwencji operacji **void Dodaj_zakup(TZakup aTZakup)** z klasy **TRachunek**:



2.2. Kod operacji **void Dodaj_zakup(TZakup aTZakup)** z klasy **TRachunek**:

```
public void Dodaj_zakup(TZakup aTZakup) {
    TZakup zakup;
    if ((zakup = Szukaj_zakup(aTZakup)) != null)
        zakup.Dodaj_ilosc(aTZakup.Podaj_ilosc());
    else Zakupy.add(aTZakup);
}
```

2.3. Kod operacji **public void Dodaj_ilosc(int avoid)** w klasie **TZakup**:

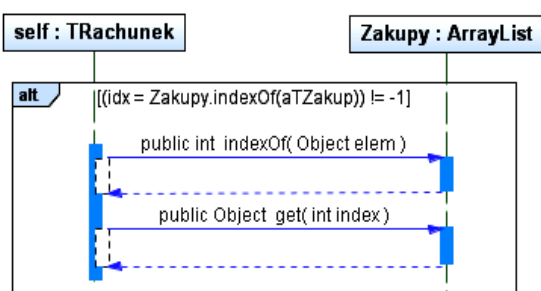
```
public void Dodaj_ilosc(int avoid) {
    ilosc += avoid;
}
```

2.4. Kod operacji **int Podaj_ilosc()** z w klasie **TZakup**:

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

3. Modelowanie i implementacja operacji **TZakup Szukaj_zakup(TZakup aTZakup)** z klasy **TRachunek**.

3.1. Diagram sekwencji operacji:

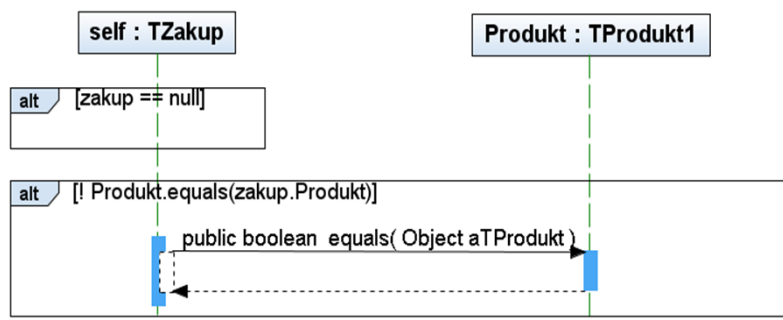


3.2. Kod operacji:

```
public TZakup Szukaj_zakup(TZakup aTZakup) {
    int idx;
    if ((idx = Zakupy.indexOf(aTZakup)) != -1) {
        aTZakup = Zakupy.get(idx);
        return aTZakup;
    }
    return null;
}
```

4. Modelowanie i implementacja operacji **boolean equals(Object aTZakup)** z klasy **TZakup**, wywoływanej w metodzie **indexOf** obiektu **Zakupy** typu **ArrayList**.

4.1. Diagram sekwencji operacji:



4.2. Kod operacji:

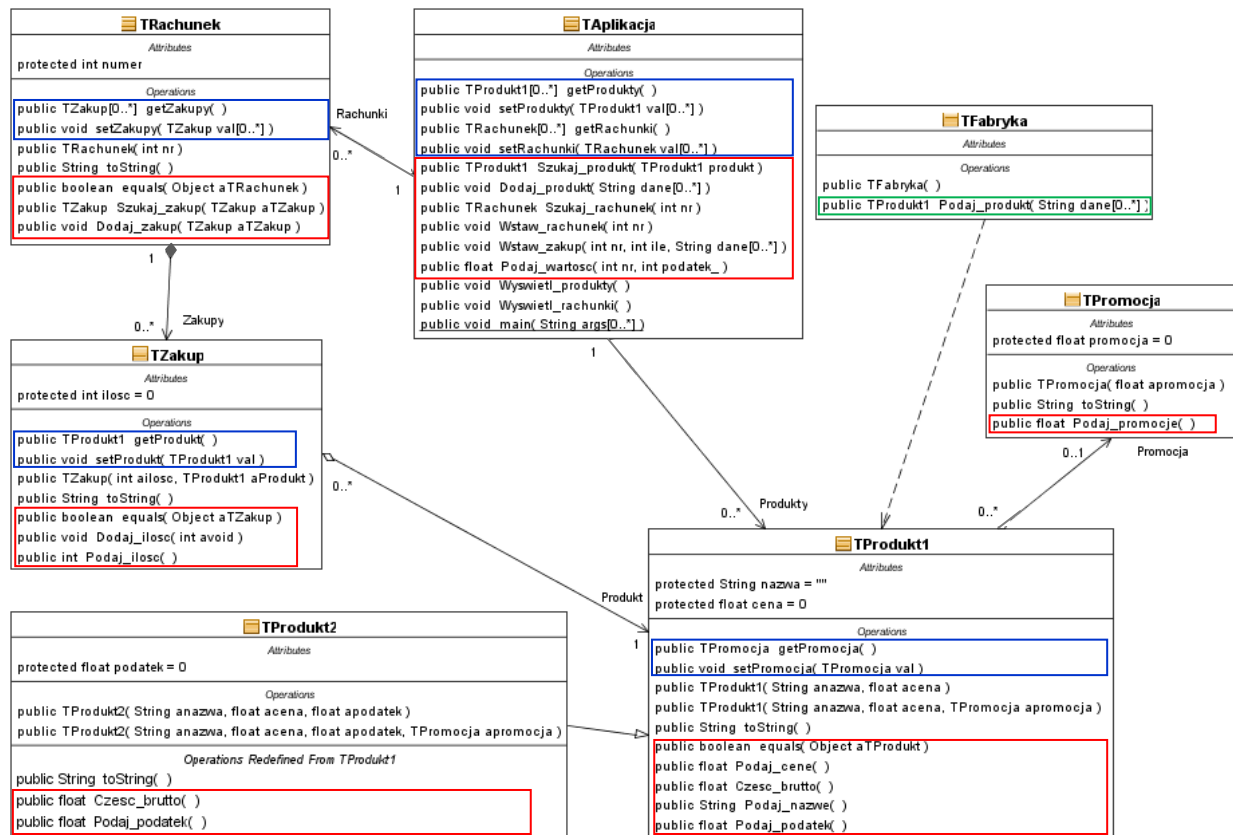
@Override

```

public boolean equals(Object aTZakup) {
    TZakup zakup = (TZakup) aTZakup;
    if (zakup == null)
        return false;
    boolean bStatus = true;
    if (!Produkt.equals(zakup.Produkt))
        bStatus = false;
    return bStatus;
}

```

5. Diagram klas zawierający elementy wynikające z wykonanych diagramów sekwencji w 3-iteracji.



6. Wykonanie kodu źródłowego programu na podstawie wykonanego diagramu klas i diagramów sekwencji oraz pomocniczych metod do prezentacji wyników metod logiki biznesowej modelowanych za pomocą diagramów sekwencji – prezentacja wyników pierwszych trzech iteracji, gdzie dodatkowo prezentuje się zawartość obiektów typu TRachunek, zawierających kolekcję obiektów typu TZakup.

Klasa Zakup

```

public TZakup(int ailosc, TProdukt1 aProdukt) {
    super();
    ilosc = ailosc;
    Produkt = aProdukt;
}

```

@Override

```

public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append(" ilosc : ");
    sb.append(ilosc);
    sb.append(" Produkt : ");
    sb.append(Produkt.toString());
    return sb.toString();
}

```


Klasa TAplikacja

```

public static void main(String args[]) {
    TAplikacja app = new TAplikacja();
    String dane1[] = {"0", "1", "1"};
    String dane2[] = {"0", "2", "2"};
    app.Dodaj_produkt(dane1);
    app.Dodaj_produkt(dane2);
    app.Dodaj_produkt(dane1);
    String dane3[] = {"2", "3", "3", "14"};
    String dane4[] = {"2", "4", "4", "22"};
    app.Dodaj_produkt(dane3);
    app.Dodaj_produkt(dane4);
    app.Dodaj_produkt(dane3);
    String dane5[] = {"1", "5", "1", "30"};
    String dane6[] = {"1", "6", "2", "50"};
    app.Dodaj_produkt(dane5);
    app.Dodaj_produkt(dane6);
    app.Dodaj_produkt(dane5);
    String dane7[] = {"3", "7", "3", "3", "30"};
    String dane8[] = {"3", "8", "4", "7", "50"};
    app.Dodaj_produkt(dane7);
    app.Dodaj_produkt(dane8);
    app.Dodaj_produkt(dane7);
    System.out.println("\nProdukty\n");
    app.Wyswietl_produkty();

    app.Wstaw_rachunek(1);
    app.Wstaw_rachunek(1);
    app.Wstaw_rachunek(2);

    app.Wstaw_zakup(1, 1, dane1);
    app.Wstaw_zakup(1, 2, dane2);
    app.Wstaw_zakup(1, 1, dane3);
    app.Wstaw_zakup(1, 4, dane4);
    app.Wstaw_zakup(1, 1, dane5);
    app.Wstaw_zakup(2, 1, dane6);
    app.Wstaw_zakup(2, 3, dane7);
    app.Wstaw_zakup(2, 1, dane8);
    app.Wstaw_zakup(2, 4, dane2);
    app.Wstaw_zakup(2, 1, dane4);
    app.Wstaw_zakup(2, 1, dane6);
    app.Wstaw_zakup(2, 1, dane8);

    app.Wyswietl_rachunki();
    System.out.println("\nRachunki\n");
    TRachunek rachunek;
    if ((rachunek = app.Szukaj_rachunek(1)) != null){
        System.out.println(rachunek.toString()); }
    if ((rachunek = app.Szukaj_rachunek(2)) != null){
        System.out.println(rachunek.toString()); }

}

```

```

Wiersz polecenia

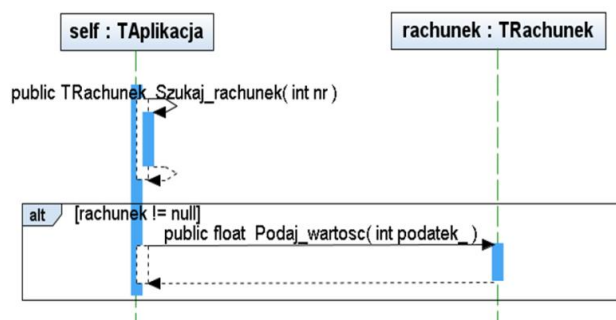
Produkty
nazwa : 1 cena : 1.0
nazwa : 2 cena : 2.0
nazwa : 3 cena : 3.42 podatek : 14.0
nazwa : 4 cena : 4.88 podatek : 22.0
nazwa : 5 cena : 0.7 promocja : 30.0
nazwa : 6 cena : 0.9 promocja : 55.0
nazwa : 7 cena : 3.99 promocja : 30.0 podatek : 3.0
nazwa : 8 cena : 6.48 promocja : 55.0 podatek : 7.0
Rachunek : 1
ilosc : 1 Produkt : nazwa : 1 cena : 1.0
ilosc : 2 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 3 cena : 3.42 podatek : 14.0
ilosc : 4 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0
ilosc : 1 Produkt : nazwa : 5 cena : 0.7 promocja : 30.0
Rachunek : 2
ilosc : 2 Produkt : nazwa : 6 cena : 0.9 promocja : 55.0
ilosc : 3 Produkt : nazwa : 7 cena : 3.99 promocja : 30.0 podatek : 3.0
ilosc : 2 Produkt : nazwa : 8 cena : 6.48 promocja : 55.0 podatek : 7.0
ilosc : 4 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0
Rachunki
Rachunek : 1
ilosc : 1 Produkt : nazwa : 1 cena : 1.0
ilosc : 2 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 3 cena : 3.42 podatek : 14.0
ilosc : 4 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0
ilosc : 1 Produkt : nazwa : 5 cena : 0.7 promocja : 30.0
Rachunek : 2
ilosc : 2 Produkt : nazwa : 6 cena : 0.9 promocja : 55.0
ilosc : 3 Produkt : nazwa : 7 cena : 3.99 promocja : 30.0 podatek : 3.0
ilosc : 2 Produkt : nazwa : 8 cena : 6.48 promocja : 55.0 podatek : 7.0
ilosc : 4 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0

```

4-a iteracja: modelowanie przypadku użycia **PU Obliczanie wartosci rachunku**

1. Modelowanie i implementacja operacji **float Podaj_wartosc(int nr, int podatek_)** z klasy **TAplikacja**

1.1. Diagram sekwencji operacji:



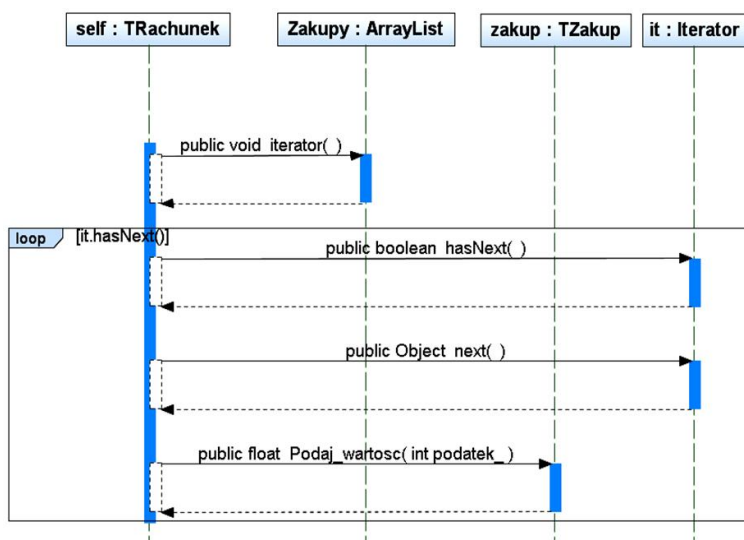
1.2. Kod operacji:

```

public float Podaj_wartosc(int nr, int podatek_) {
    TRachunek rachunek;
    rachunek = Szukaj_rachunek(nr);
    if (rachunek != null) {
        return rachunek.Podaj_wartosc(podatek_);
    }
    return 0F;
}
  
```

2. Modelowanie i implementacja operacji **float Podaj_wartosc(int podatek_)** z klasy **TRachunek**

2.1. Diagram sekwencji operacji:



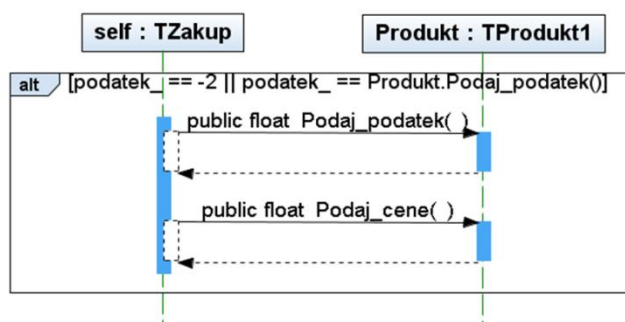
2.2. Kod operacji:

```

public float Podaj_wartosc(int podatek_) {
    float suma = 0;
    TZakup zakup;
    Iterator<TZakup> it = Zakupy.iterator();
    while (it.hasNext()) {
        zakup = it.next();
        suma += zakup.Podaj_wartosc(podatek_);
    }
    return suma;
}
  
```

3. Modelowanie i implementacja operacji **float Podaj_wartosc(int podatek_)** z klasy **TZakup**.

3.1. Diagram sekwencji operacji:

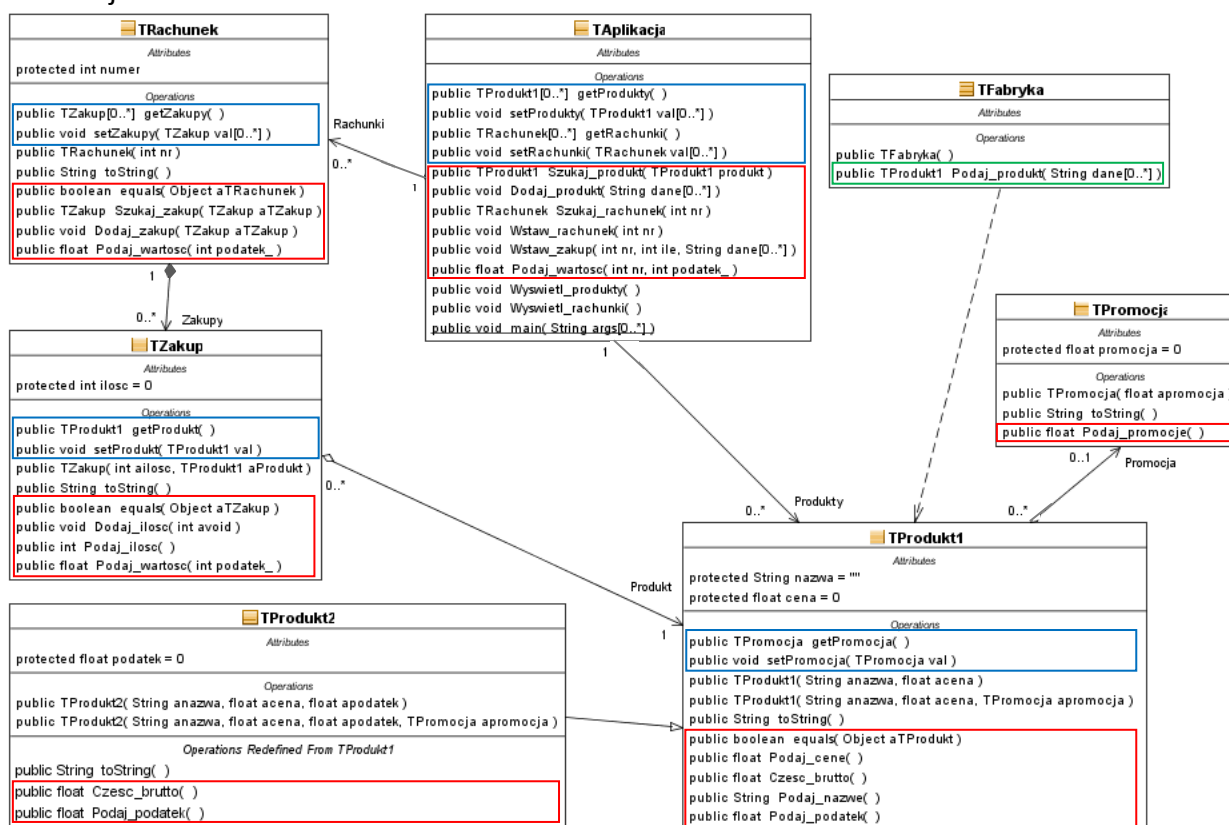


3.2. Kod operacji:

```

public float Podaj_wartosc(int podatek_) {
    if (podatek_ == -2 || podatek_ == Produkt.Podaj_podatek()) {
        return ilosc * Produkt.Podaj_cene();
    }
    return 0F;
}
  
```

4. Diagram klas zawierający elementy wynikające z wykonanych diagramów sekwencji w 4-iteracji



5. Wykonanie kodu źródłowego programu na podstawie wykonanego diagramu klas i diagramów sekwencji oraz pomocniczych metod do prezentacji wyników metod logiki biznesowej modelowanych za pomocą diagramów sekwencji – prezentacja wyników czterech iteracji, gdzie prezentuje się zawartość obiektów typu **TRachunek** zawierających kolekcję obiektów typu **TZakup** oraz wartość tych rachunków.

TRachunek

@Override

```
public String toString() {
    TZakup z;
    StringBuilder sb = new StringBuilder();
    sb.append(" Rachunek : ");
    sb.append(numer).append("\n");
    for (TZakup zakup: Zakupy)
        sb.append(zakup.toString()).append("\n");
    sb.append("Wartosc zakupow O: ").append(Podaj_wartosc(-1)).append("\n");
    sb.append("Wartosc zakupow A: ").append(Podaj_wartosc(3)).append("\n");
    sb.append("Wartosc zakupow B: ").append(Podaj_wartosc(7)).append("\n");
    sb.append("Wartosc zakupow C: ").append(Podaj_wartosc(14)).append("\n");
    sb.append("Wartosc zakupow D: ").append(Podaj_wartosc(22)).append("\n");
    sb.append("Wartosc rachunku: ").append(Podaj_wartosc(-2)).append("\n");
    return sb.toString();
}
```

TAplikacja

```
public static void main(String args[]) {
    TAplikacja app = new TAplikacja();
    String dane1[] = {"0", "1", "1"};           String dane2[] = {"0", "2", "2"};
    app.Dodaj_produkt(dane1);
    app.Dodaj_produkt(dane2);
    app.Dodaj_produkt(dane1);
    String dane3[] = {"2", "3", "3", "14"};      String dane4[] = {"2", "4", "4", "22"};
    app.Dodaj_produkt(dane3);
    app.Dodaj_produkt(dane4);
    app.Dodaj_produkt(dane3);
    String dane5[] = {"1", "5", "1", "30"};      String dane6[] = {"1", "6", "2", "50"};
    app.Dodaj_produkt(dane5);
    app.Dodaj_produkt(dane6);
    app.Dodaj_produkt(dane5);
    String dane7[] = {"3", "7", "3", "3", "30"}; String dane8[] = {"3", "8", "4", "7", "50"};
    app.Dodaj_produkt(dane7);
    app.Dodaj_produkt(dane8);
    app.Dodaj_produkt(dane7);
    System.out.println("\nProdukty\n");
    app.Wyswietl_produkty();

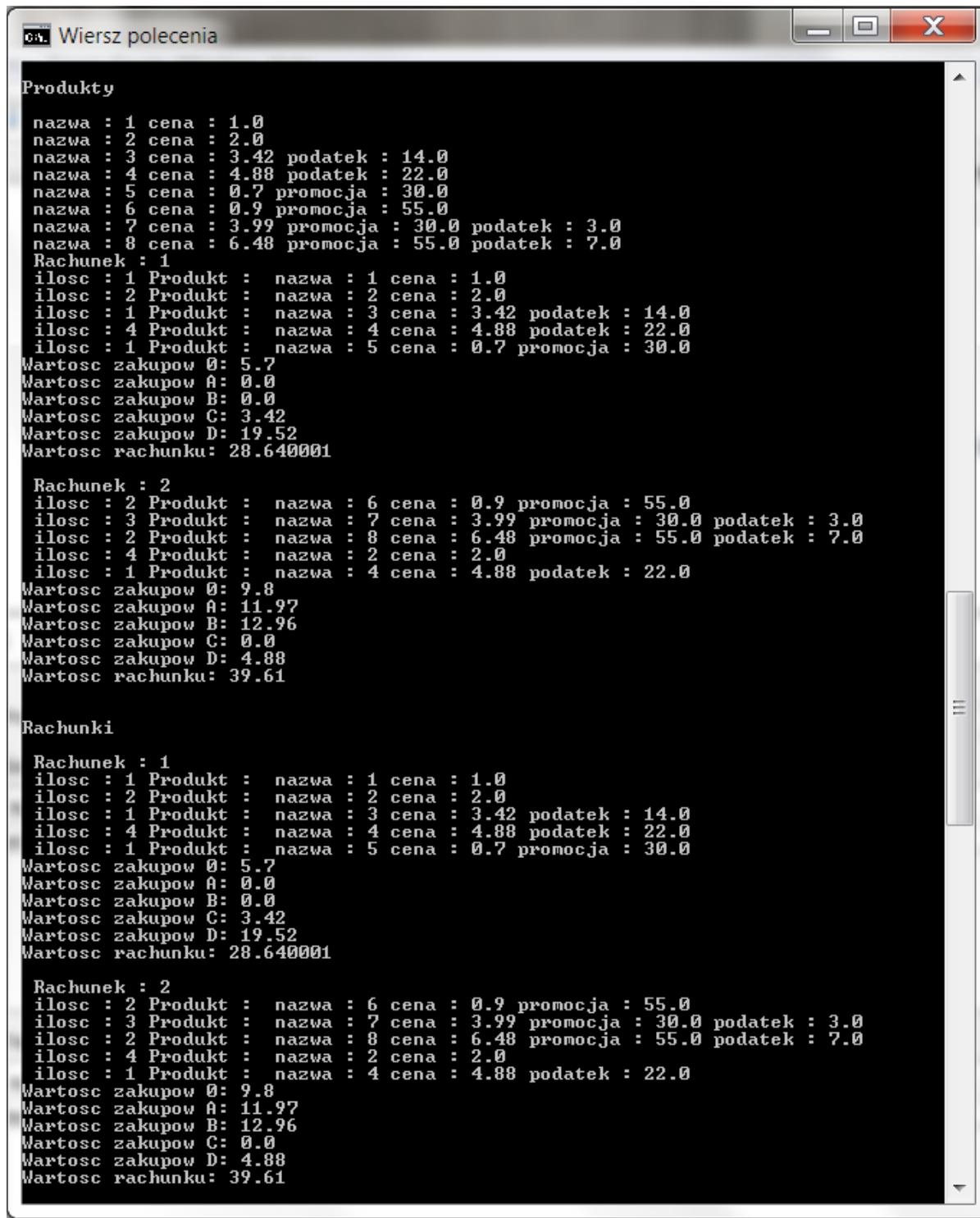
    app.Wstaw_rachunek(1);
    app.Wstaw_rachunek(1);
    app.Wstaw_rachunek(2);
    app.Wstaw_zakup(1, 1, dane1);
    app.Wstaw_zakup(1, 2, dane2);
    app.Wstaw_zakup(1, 1, dane3);
    app.Wstaw_zakup(1, 4, dane4);
    app.Wstaw_zakup(1, 1, dane5);
    app.Wstaw_zakup(2, 1, dane6);
    app.Wstaw_zakup(2, 3, dane7);
    app.Wstaw_zakup(2, 1, dane8);
    app.Wstaw_zakup(2, 4, dane2);
    app.Wstaw_zakup(2, 1, dane4);
}
```

```

app.Wstaw_zakup(2, 1, dane6);
app.Wstaw_zakup(2, 1, dane8);
app.Wyswietl_rachunki();

System.out.println("\nRachunki\n");
TRachunek rachunek;
if ((rachunek = app.Szukaj_rachunek(1)) != null) {
    System.out.println(rachunek.toString());
}
if ((rachunek = app.Szukaj_rachunek(2)) != null) {
    System.out.println(rachunek.toString());
}
}

```



```

Wiersz polecenia

Produkty
nazwa : 1 cena : 1.0
nazwa : 2 cena : 2.0
nazwa : 3 cena : 3.42 podatek : 14.0
nazwa : 4 cena : 4.88 podatek : 22.0
nazwa : 5 cena : 0.7 promocja : 30.0
nazwa : 6 cena : 0.9 promocja : 55.0
nazwa : 7 cena : 3.99 promocja : 30.0 podatek : 3.0
nazwa : 8 cena : 6.48 promocja : 55.0 podatek : 7.0
Rachunek : 1
ilosc : 1 Produkt : nazwa : 1 cena : 1.0
ilosc : 2 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 3 cena : 3.42 podatek : 14.0
ilosc : 4 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0
ilosc : 1 Produkt : nazwa : 5 cena : 0.7 promocja : 30.0
Wartosc zakupow 0: 5.7
Wartosc zakupow A: 0.0
Wartosc zakupow B: 0.0
Wartosc zakupow C: 3.42
Wartosc zakupow D: 19.52
Wartosc rachunku: 28.640001

Rachunek : 2
ilosc : 2 Produkt : nazwa : 6 cena : 0.9 promocja : 55.0
ilosc : 3 Produkt : nazwa : 7 cena : 3.99 promocja : 30.0 podatek : 3.0
ilosc : 2 Produkt : nazwa : 8 cena : 6.48 promocja : 55.0 podatek : 7.0
ilosc : 4 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0
Wartosc zakupow 0: 9.8
Wartosc zakupow A: 11.97
Wartosc zakupow B: 12.96
Wartosc zakupow C: 0.0
Wartosc zakupow D: 4.88
Wartosc rachunku: 39.61

Rachunki

Rachunek : 1
ilosc : 1 Produkt : nazwa : 1 cena : 1.0
ilosc : 2 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 3 cena : 3.42 podatek : 14.0
ilosc : 4 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0
ilosc : 1 Produkt : nazwa : 5 cena : 0.7 promocja : 30.0
Wartosc zakupow 0: 5.7
Wartosc zakupow A: 0.0
Wartosc zakupow B: 0.0
Wartosc zakupow C: 3.42
Wartosc zakupow D: 19.52
Wartosc rachunku: 28.640001

Rachunek : 2
ilosc : 2 Produkt : nazwa : 6 cena : 0.9 promocja : 55.0
ilosc : 3 Produkt : nazwa : 7 cena : 3.99 promocja : 30.0 podatek : 3.0
ilosc : 2 Produkt : nazwa : 8 cena : 6.48 promocja : 55.0 podatek : 7.0
ilosc : 4 Produkt : nazwa : 2 cena : 2.0
ilosc : 1 Produkt : nazwa : 4 cena : 4.88 podatek : 22.0
Wartosc zakupow 0: 9.8
Wartosc zakupow A: 11.97
Wartosc zakupow B: 12.96
Wartosc zakupow C: 0.0
Wartosc zakupow D: 4.88
Wartosc rachunku: 39.61

```

Dodatek 2

Tworzenie diagramów klas i sekwencji użycia w wybranym środowisku np Visual Paradigm

1. Pomoc: **Drawing class diagrams.**
(http://www.visual-paradigm.com/support/documents/vpumluserguide/94/2576/7190_drawingclass.html)
2. Pomoc: **Drawing sequence diagrams.**
(http://www.visual-paradigm.com/support/documents/vpumluserguide/94/2577/7025_drawingseque.html)