Modelowanie i analiza warstwy biznesowej aplikacji

1. **Warstwa biznesowa aplikacji, refaktoryzacja warstwy biznesowej, refaktoryzacja systemu informatycznego**
2. **Przykład tworzenia warstwy biznesowej systemu informatycznego**

Modelowanie i analiza warstwy biznesowej aplikacji

1. **Warstwa biznesowa aplikacji, refaktoryzacja warstwy biznesowej, refaktoryzacja systemu informatycznego**

Refaktoryzacja

Refaktoryzacja polega ona na modyfikacji oprogramowania w celu poprawy jego struktury zachowując podstawowe funkcje oprogramowania.

W praktyce poprawę struktury oprogramowania uzyskuje się za pomocą:

- podziału oprogramowania na warstwy
- wzorców oprogramowania, zastosowanych do budowy każdej warstwy na wybranym poziomie abstrakcji

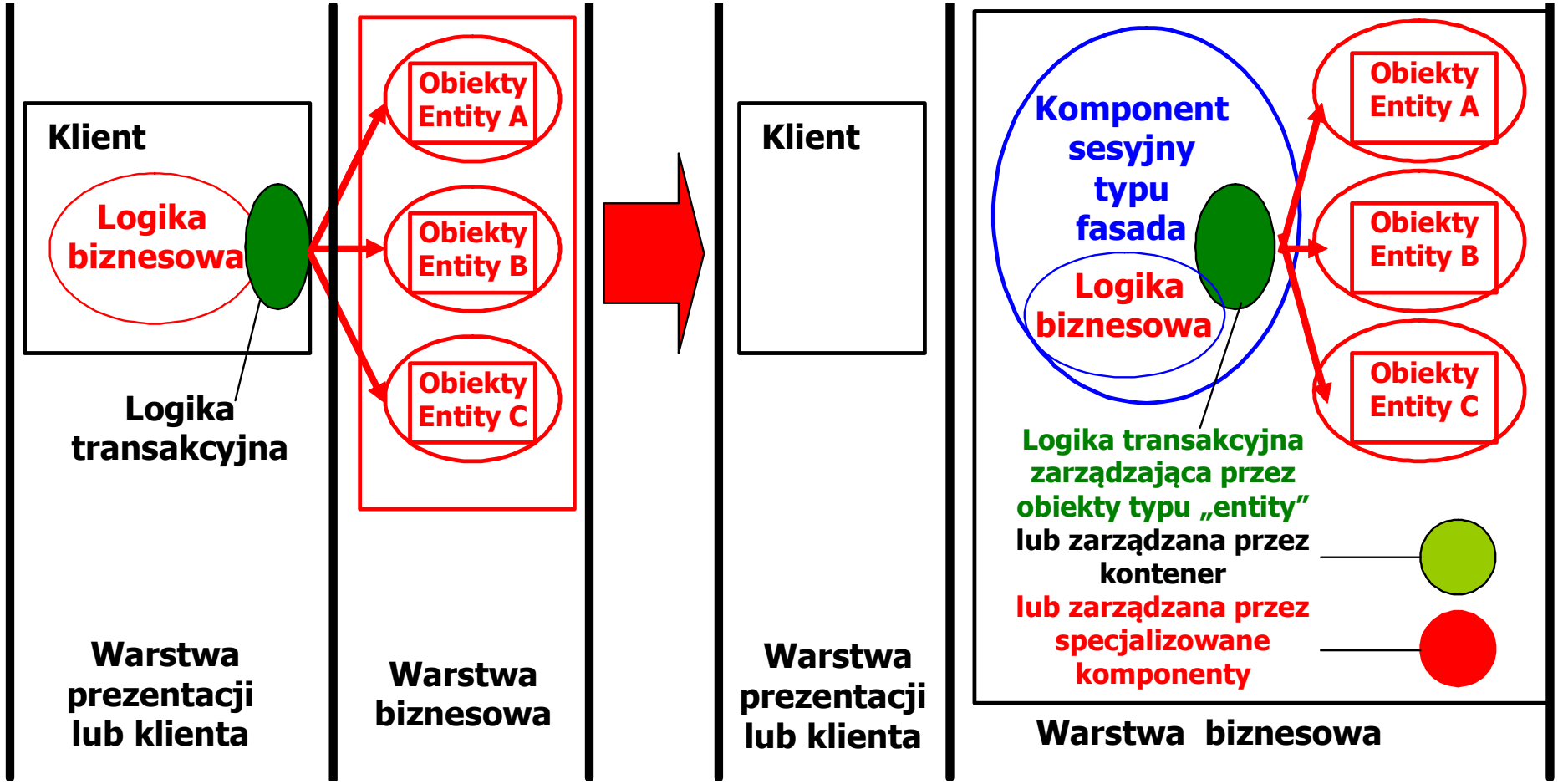
Rodzaje wzorców:

- Wzorce projektowe
- Wzorce architektury
- Wzorce analizy
- Wzorce konstrukcyjne
- Wzorce strukturalne
- Wzorce czynnościowe

Pięciowarstwowy model logicznego rozdzielania zadań (wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

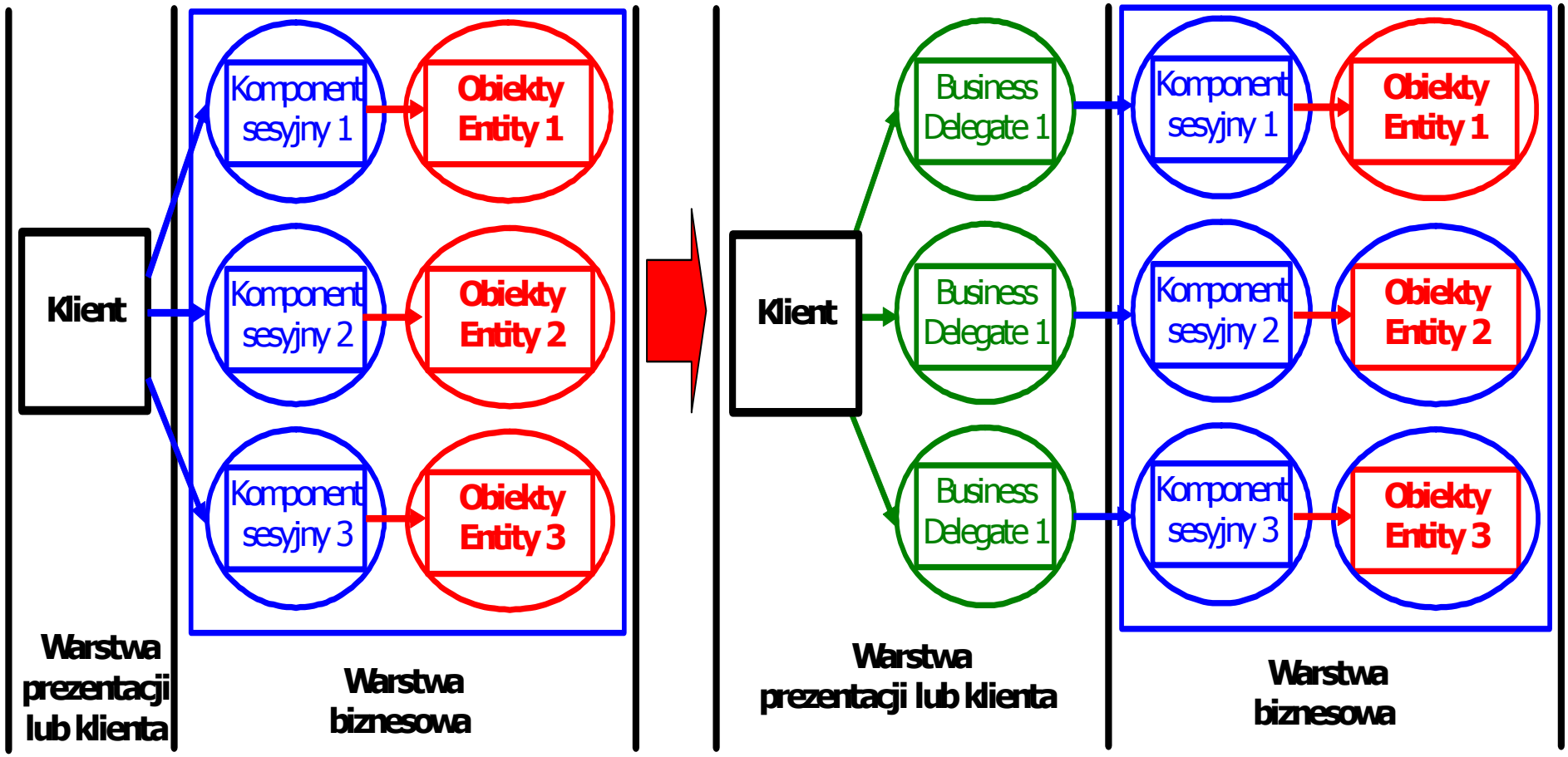
Refaktoryzacja warstwy biznesowej 1

Obiekty danych typu „Entity” (obiekty biznesowe) z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **fasadowych komponentów sesyjnych typu „Control” (komponent typu fasada - hermetyzujący dostęp do usług biznesowych)**



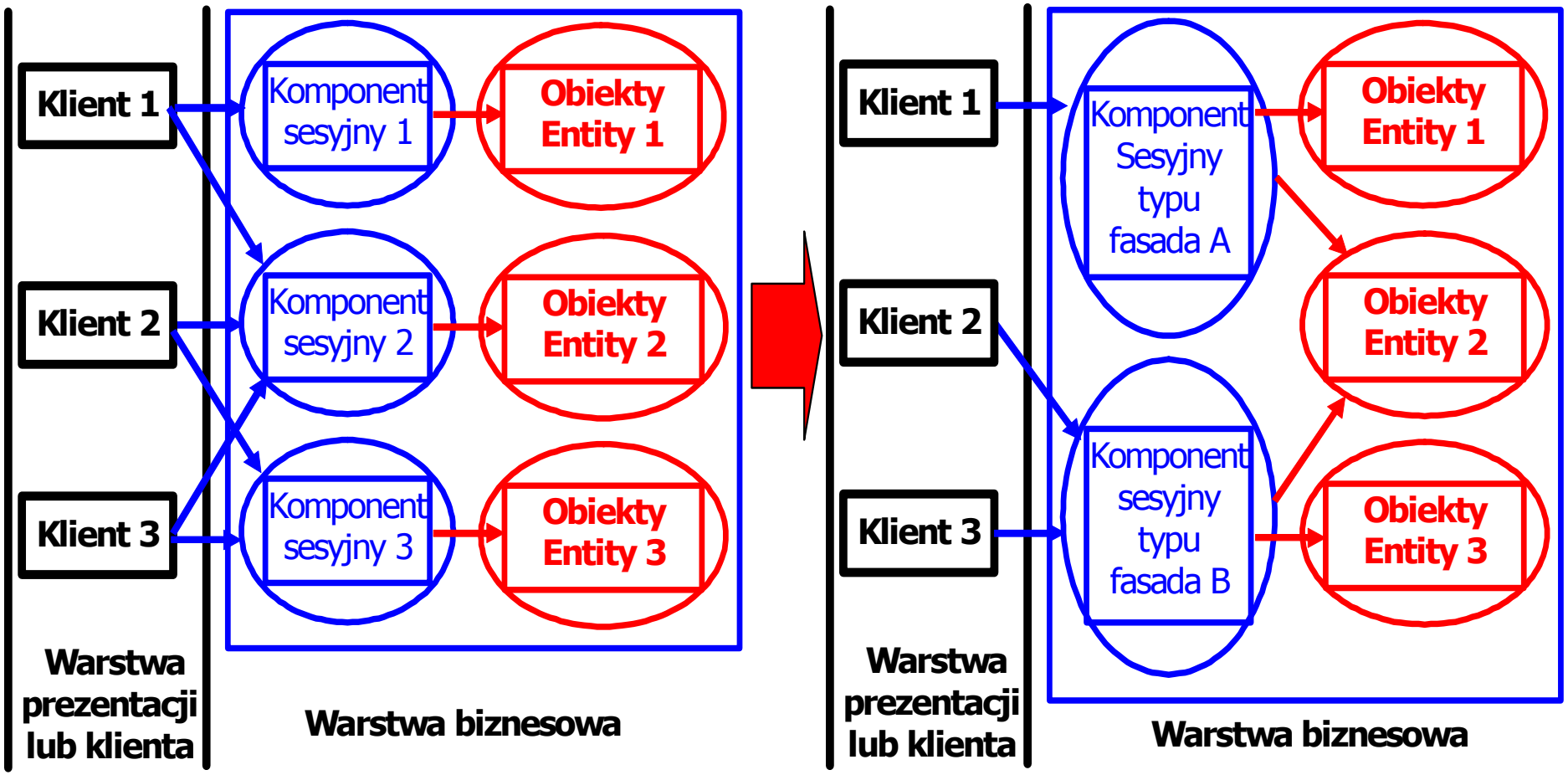
Refaktoryzacja warstwy biznesowej 2

Komponenty sesyjne typu „Control” (pośredniczące w dostępie do **obiektów danych typu „Entity”**) z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **obiektów fasadowych typu „Control” (hermetyzujących dostęp do warstwy biznesowej- komponentów Business Delegate)**



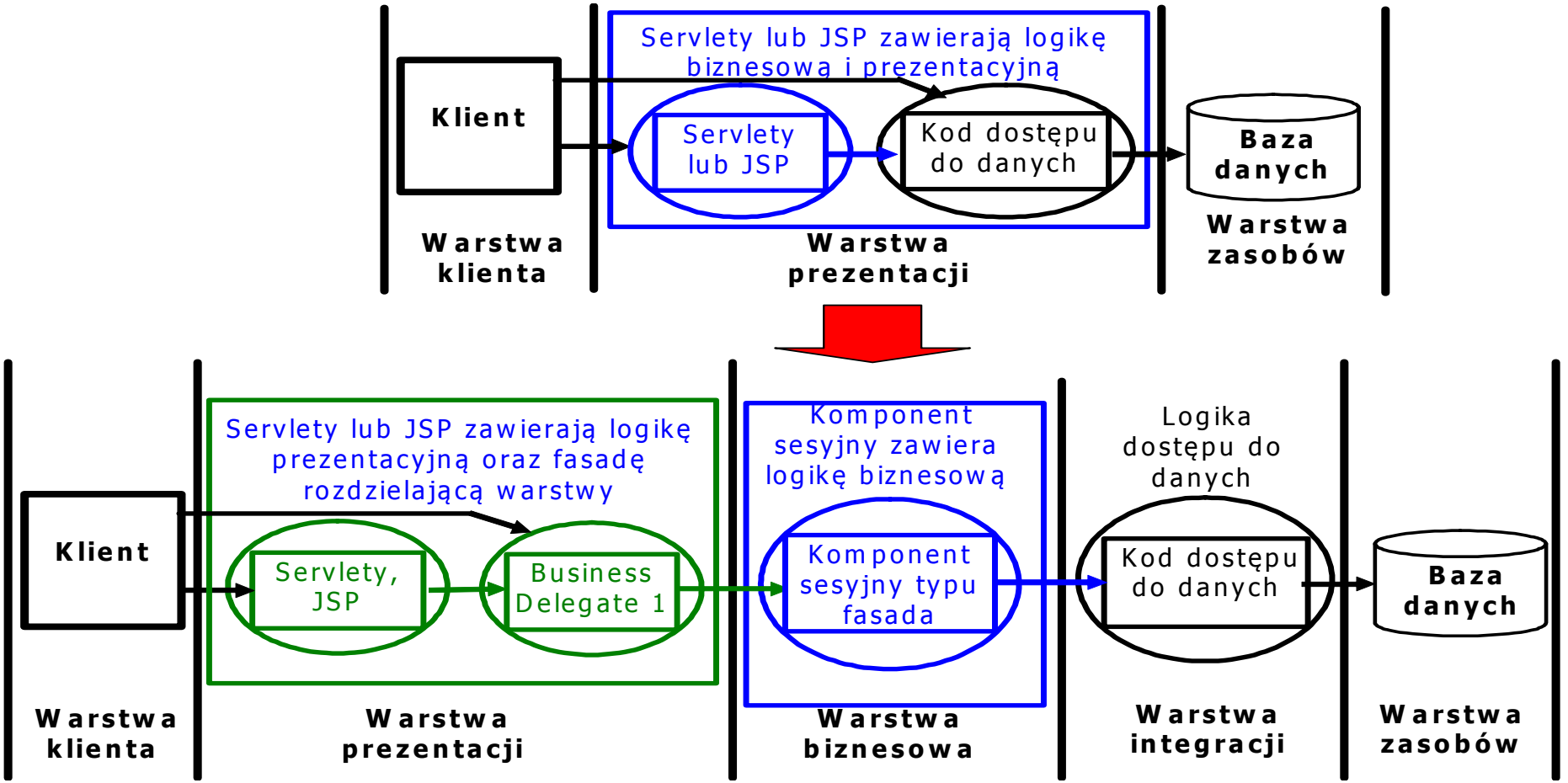
Refaktoryzacja warstwy biznesowej 3

Sesyjne komponenty fasadowe typu „Control” (każdy komponent jako odrębna usługa biznesowa), hermetyzujące **obiekty danych typu „Entity”** z warstwy biznesowej są udostępniane klientom w innych warstwach. Zwykłe obiekty sesyjne są jedynie pośrednikami obiektów „Entity”, natomiast nie hermetyzują całych usług, które wymagają odwołania do wielu zwykłych komponentów sesyjnych.



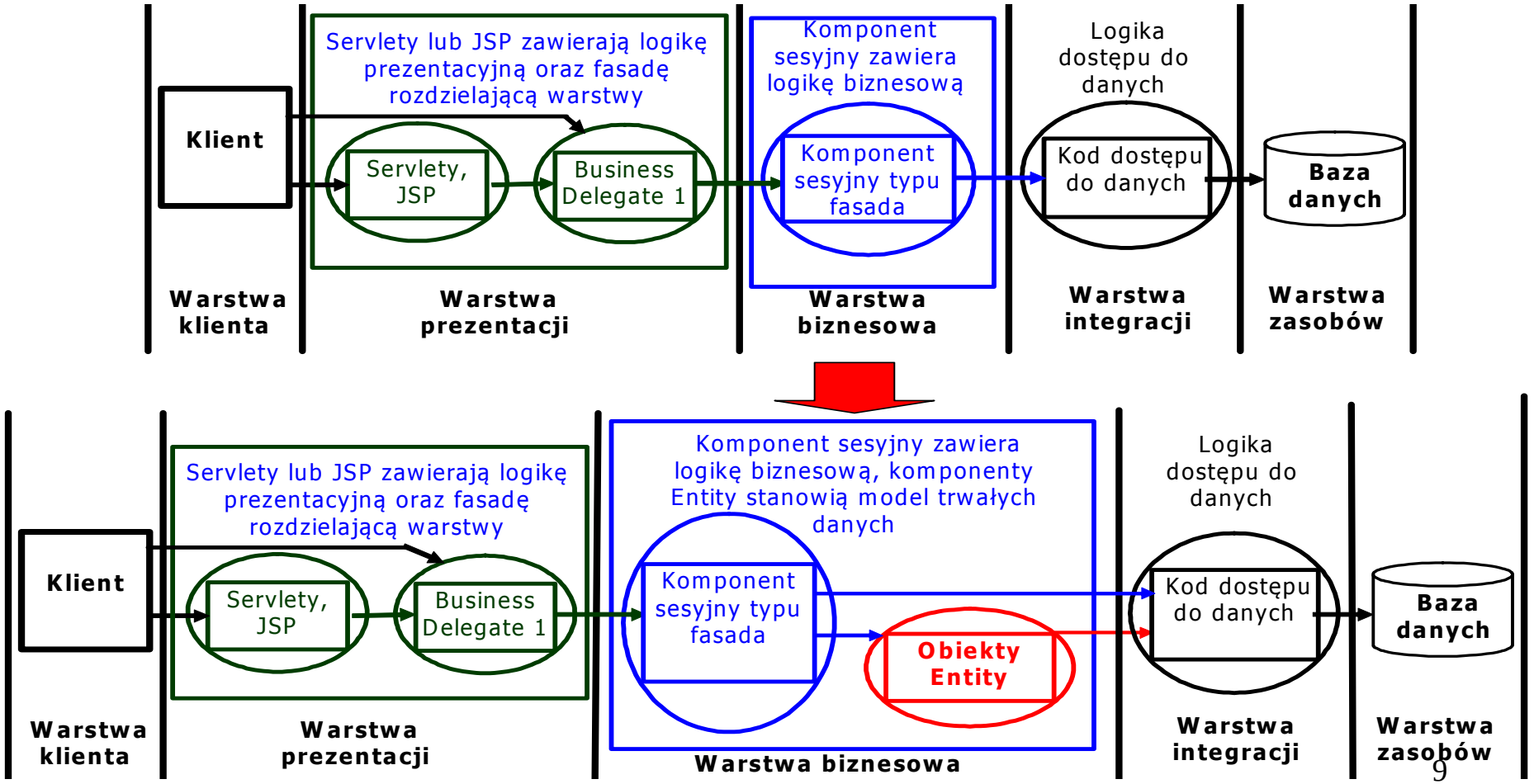
Refaktoryzacja architektury wielowarstwowej 1

Należy przenieść kod dostępu do danych logicznie lub fizycznie bliżej rzeczywistego źródła danych, a logikę przetwarzania z klienta i warstwy prezentacji do warstwy biznesowej zawierającej **fasadowe komponenty sesyjne typu „Control”.** Komponenty Business Delegate typu „Control” hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji – stanowią przedłużenie warstwy biznesowej.

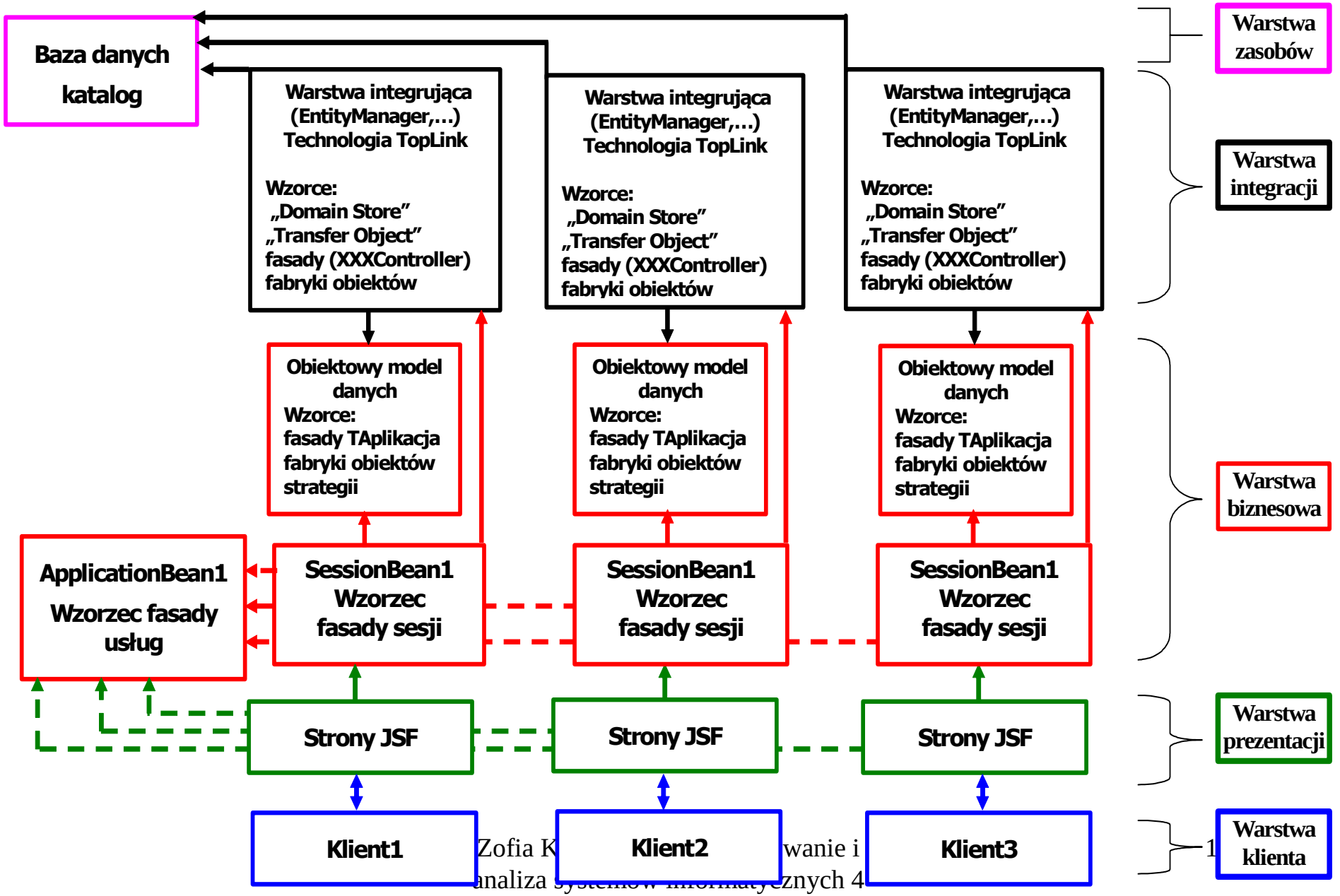


Refaktoryzacja architektury wielowarstwowej 2

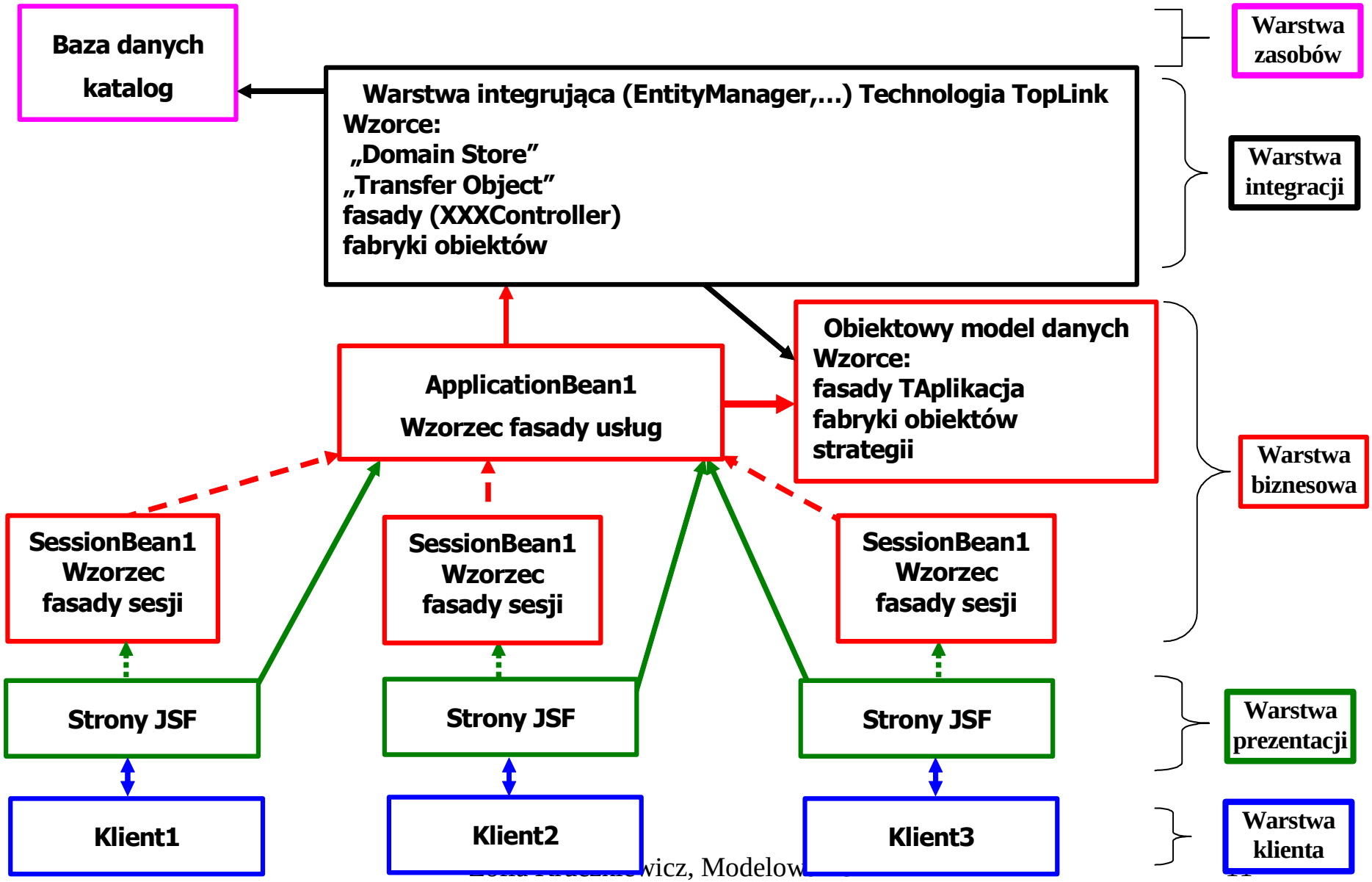
Należy przenieść kod dostępu do danych logicznie lub fizycznie bliżej rzeczywistego źródła danych, a złożoną logikę przetwarzania z klienta i warstwy prezentacji typu do warstwy biznesowej zawierającą **obiekty danych typu „Entity" i hermetyzujace dostep do tych komponentów fasadowe komponenty sesyjne typu „Control".** Komponenty Business Delegate typu „Control" hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji.



Architektura aplikacji pięciowarstwowej – Java EE 5.0 Visual Web Java Server Faces (linie przerywane oznaczają powiązania nie wykorzystane w aplikacji



Architektura aplikacji pięciowarstwowej Java EE 5.0 Visual Web Java Server Faces - linie przerywane oznaczają powiązania nie wykorzystane w aplikacji



Modelowanie i analiza warstwy biznesowej aplikacji

1. **Warstwa biznesowa aplikacji, refaktoryzacja warstwy biznesowej, refaktoryzacja systemu informatycznego**
2. **Przykład tworzenia warstwy biznesowej systemu informatycznego**

System sporządzania rachunków

4.1. Sformułowanie wymagań funkcjonalnych i niefunkcjonalnych systemu

4.2. **Model analizy całego systemu** oparty na diagramie przypadków użycia

4.3. **Model projektowy warstwy biznesowej** oparty na diagramie klas i diagramie sekwencji tworzony metodą iteracyjno-rozwojową sterowany realizacją przypadków użycia

4.4. **Implementacja warstwy biznesowej** tworzona w cyklu iteracyjno-rozwojowym sterowana rozwojem modelu projektowego

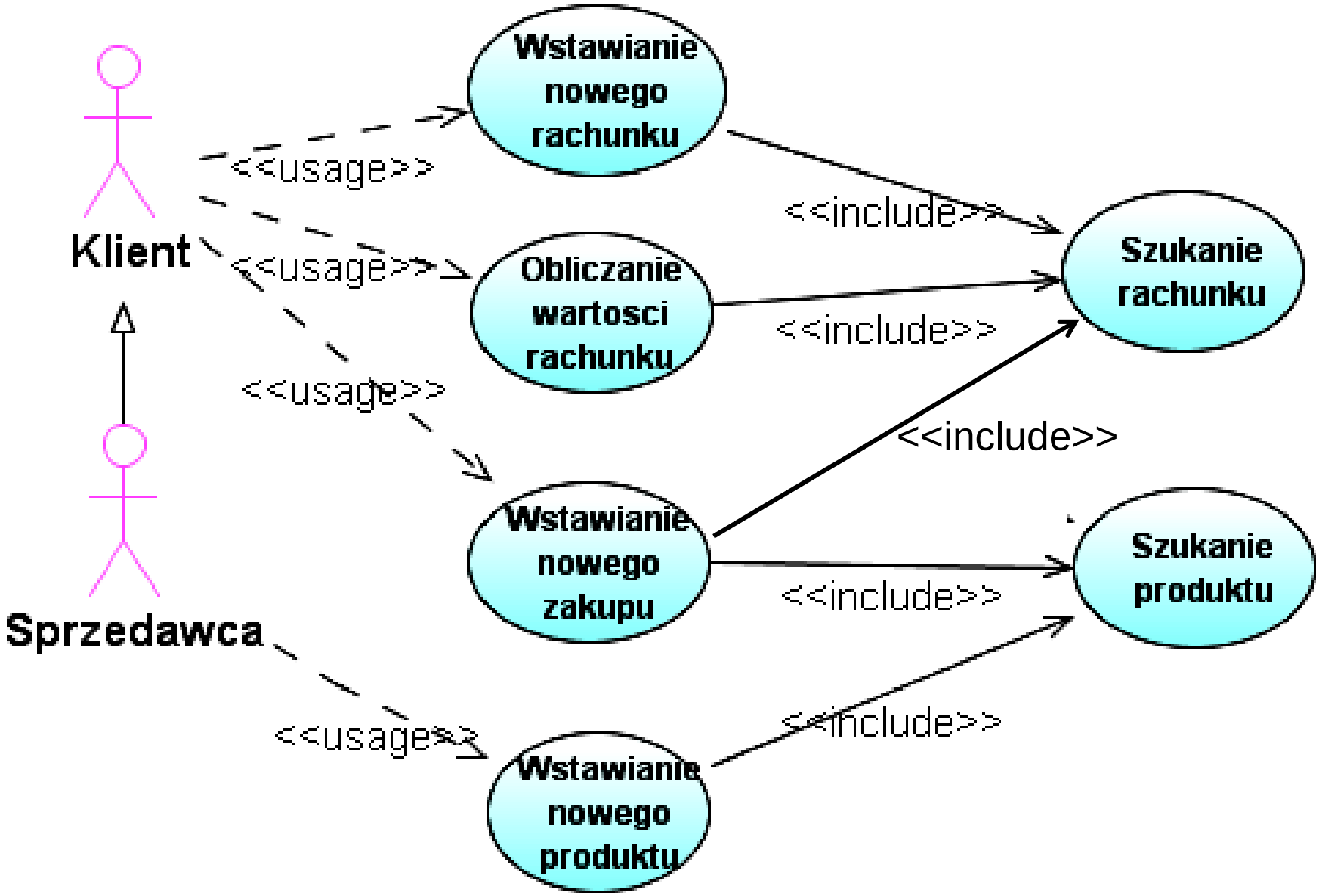
Przykład: System sporządzania rachunków

Lista wymagań funkcjonalnych

1. System zawiera katalog produktów
2. Można zakupić trzy typy produktów różniące się sposobem obliczania ceny detalicznej: netto, z podatkiem, z promocją,
3. Można wprowadzić wiele rachunków
4. Pozycje rachunku muszą zawierać produkty różne w sensie nazwy, ceny, podatku i promocji
5. Każda pozycja rachunku powinna podać swoją wartość brutto oraz dane produktu oraz ilość zakupionego produktu.
6. Na rachunku powinna znajdować się wartość łączna wszystkich zakupów oraz wartości zakupów należących do wybranych kategorii

Lista wymagań niefunkcjonalnych

1. Wstawianie produktów może odbywać się tylko przez uprawnione osoby
2. Wstawianie nowych rachunków oraz wstawianie nowych zakupów jest dokonywane przez klientów
3. Zakupy mogą być dokonane przez Internet przez aplikację uruchamianą przez przeglądarkę lub bez jej pośrednictwa


Wstawianie nowego rachunku
Klient
<<usage>>
<<usage>>
<<include>>
Szukanie rachunku
Obliczanie wartosci rachunku
<<include>>
<<usage>>
<<include>>
Wstawianie nowego zakupu
Szukanie produktu
<<include>>
Sprzedawca
<<usage>>
<<include>>
Wstawianie nowego produktu

AKTOR	OPIS	PRZYPADKI UŻYCIA
Klient	*Klient może dokonywać zakupów wybranych produktów przez Internet korzystając z przeglądarki lub z aplikacji*	• Wstawianie nowego rachunku powiązane przez <<include>> z PU Szukanie rachunku • Obliczanie wartosci rachunku powiązane przez <<include>> z PU Szukanie rachunku • Wstawianie nowego zakupu powiązane przez <<include>> z PU Szukanie rachunku oraz powiązane przez <<include>> z PU Szukanie produktu
Sprzedawca	*Sprzedawca może dodatkowo dodawać nowe produkty*	• Wstawianie nowego rachunku powiązane przez <<include>> z PU Szukanie rachunku • Obliczanie wartosci rachunku powiązane przez <<include>> z PU Szukanie rachunku • Wstawianie nowego zakupu powiązane przez <<include>> z PU Szukanie rachunku oraz powiązane przez <<include>> z PU Szukanie produktu • Wstawianie nowego produktu powiązane przez <<include>> z PU Szukanie produktu

PU Szukanie produktu
OPIS
CEL: Poszukiwanie produktu
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): podanie produktu o podanych atrybutach obowiązkowych: nazwa i cena oraz jeśli jest to wymagane: z podatkiem i promocją lub komunikat o braku produktu
PRZEBIEG:
1. Szukanie produktu przebiega wedlug atrybutow: nazwy i ceny (obowiazkowo) oraz podatku i promocji (jeśli jest to wymagane) zgodnie z danymi podanymi do przypadku uzycia
2. Jesli istnieje produkt o podanych atrybutach, zwracany jest produkt, w przeciwnym wypadku zwracana jest informacja o braku produktu.

PU Wstawianie nowego produktu
OPIS
CEL: Wstawienie nowego produktu
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): dodanie produktu o podanych atrybutach obowiązkowych: nazwa i cena oraz jeśli jest to wymagane: z podatkiem i promocją, jeśli nie było takiego produktu
PRZEBIEG:
1. Nalezy podac atrybuty produktu: nazwe, cene jako obowiazkowe dane oraz podatek i cene detaliczna, jeśli jest to wymagane
2. Należy wywolac **PU Szukanie produktu**. Nalezy sprawdzic, czy produkt o podanych atrybutach juz istnieje. Jesli tak, nalezy zakonczyc PU, w przeciwnym wypadku nalezy wstawic nowy produkt.

PU Szukanie rachunku
OPIS
CEL: Poszukiwanie rachunku
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): podanie rachunku o podanym numerze lub komunikat o braku rachunku
PRZEBIEG:
1. Szukanie rachunku przebiega wedlug numeru podanego do przypadku uzycia
2. Jesli istnieje rachunek o podanym numerze, zwracany jest rachunek, w przeciwnym wypadku zwracana jest informacja o braku rachunku.

PU Wstawianie nowego rachunku
OPIS
CEL: Wstawienie nowego rachunku
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): dodanie rachunku o podanym numerze, jeśli jest to unikatowy numer
PRZEBIEG:
1. Nalezy podac numer rachunku, ktory powinien byc niepowtarzalny, poniewaz sluzy do identyfikacji rachunku
2. Nalezy wywolac **PU Szukanie rachunku** w celu sprawdzenia, czy numer rachunku sie powtarza.
3. Jesli zwrocony wynik oznacza brak rachunku o podanym numerze, mozna wstawic nowy rachunek i zakonczyc PU, w przeciwnym wypadku nalezy zakonczyc PU bez wstawiania nowego rachunku.

PU Obliczanie wartosci rachunku

OPIS

CEL: Obliczanie wartosci rachunku wg podanego podatku

WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

WK (warunki końcowe): podanie wartości całego rachunku o podanym numerze i parametrze wejściowym równym -2 lub wartości zakupionych towarów wg podanej kategorii podatku lub komunikat o braku rachunku

PRZEBIEG:

1. Nalezy podac numer rachunku, ktory powinien byc niepowtarzalny, poniewaz sluzy do identyfikacji rachunku oraz wartość podatku lub wartosc -2
2. Nalezy wywolac **PU Szukanie rachunku** w celu sprawdzenia, czy rachunek o podanym numerze istnieje.
3. Jesli zwrocony wynik oznacza brak rachunku o podanym numerze, nie mozna obliczyc wartosci wybranego rachunku i nalezy zakonczyc PU, w przeciwnym wypadku nalezy obliczyc wartosc rachunku
4. Nalezy uruchomic petle, w ktorej sumowane sa wartosci zakupu obliczane jako iloczyn ceny jednostkowej zakupionego produktu i ilosci zakupu. Jesli zachodzi potrzeba sumowania wartosci zakupu zalezna od wysokosci podatku, nalezy podac wartosc podatku i sumowac jedynie zakupy o podanym podatku, w przeciwnym wypadku sumowane sa wszystkie zakupy (gdy zamiast podatku zostanie przekazana wartosc -2).

PU Wstawianie nowego zakupu

OPIS

CEL: Wstawianie nowego zakupu

WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

WK (warunki końcowe): podanie nowego zakupu o podanych atrybutach lub zwiekszenie ilosci zakupionego produktu, jeśli już taki produkt zakupiono lub komunikat o braku rachunku

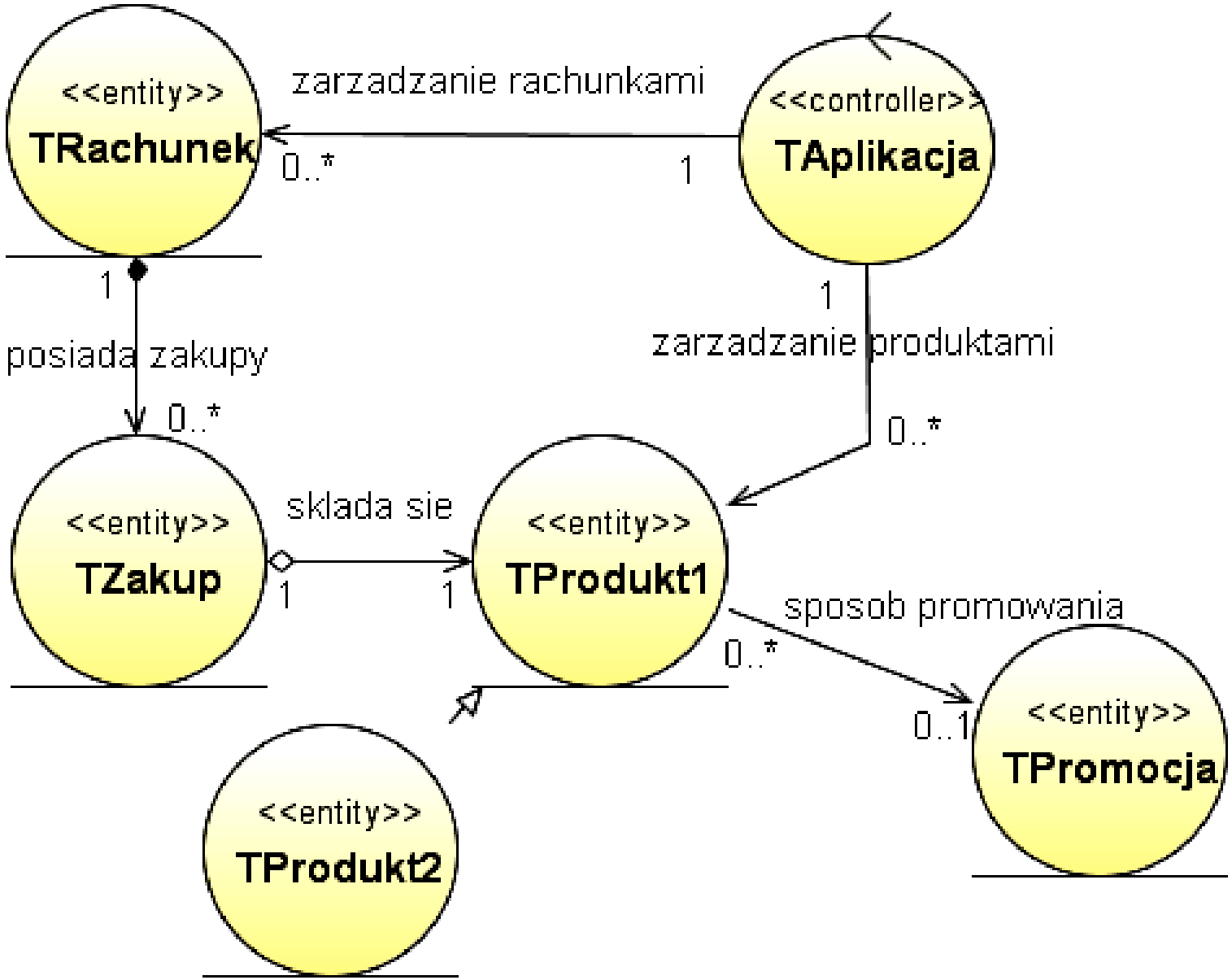
PRZEBIEG:

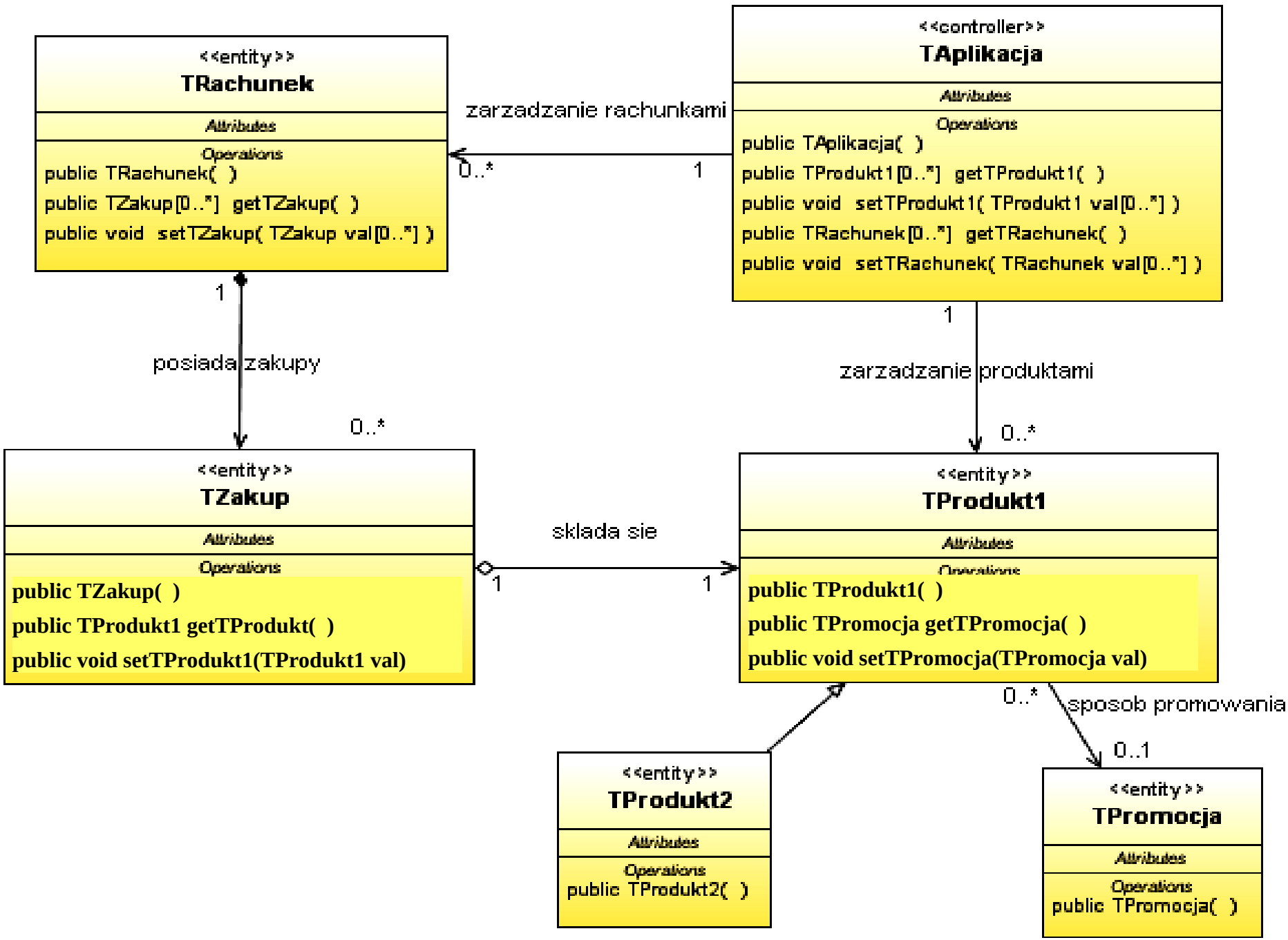
1. Nalezy podac numer rachunku, ktory powinien byc niepowtarzalny, poniewaz sluzy do identyfikacji rachunku
2. Nalezy wywolac **PU Szukanie rachunku** w celu sprawdzenia, czy istnieje rachunek o podanym numerze.
3. Jesli zwrocony wynik oznacza brak rachunku o podanym numerze, nie mozna wstawic nowego zakupu do rachunku i nalezy zakonczyc PU, w przeciwnym wypadku nalezy wstawic nowy zakup
4. Nalezy wybrac produkt oraz ilosc zakupionego produktu.
5. Należy wywolac **PU Szukanie produktu**. Jeśli wybrany produkt nie istnieje, należy zakonczyc PU. W przeciwnym przypadku nalezy wstawic nowy zakup do rachunku, przegladajac, czy istnieje juz zakup z takim samym produktem. Jesli istnieje, nie tworzy sie nowego zakupu, tylko powieksza się ilosc zakupu istniejacego o ilosc nowego zakupu, w przeciwnym przypadku wstawia sie nowy zakup.

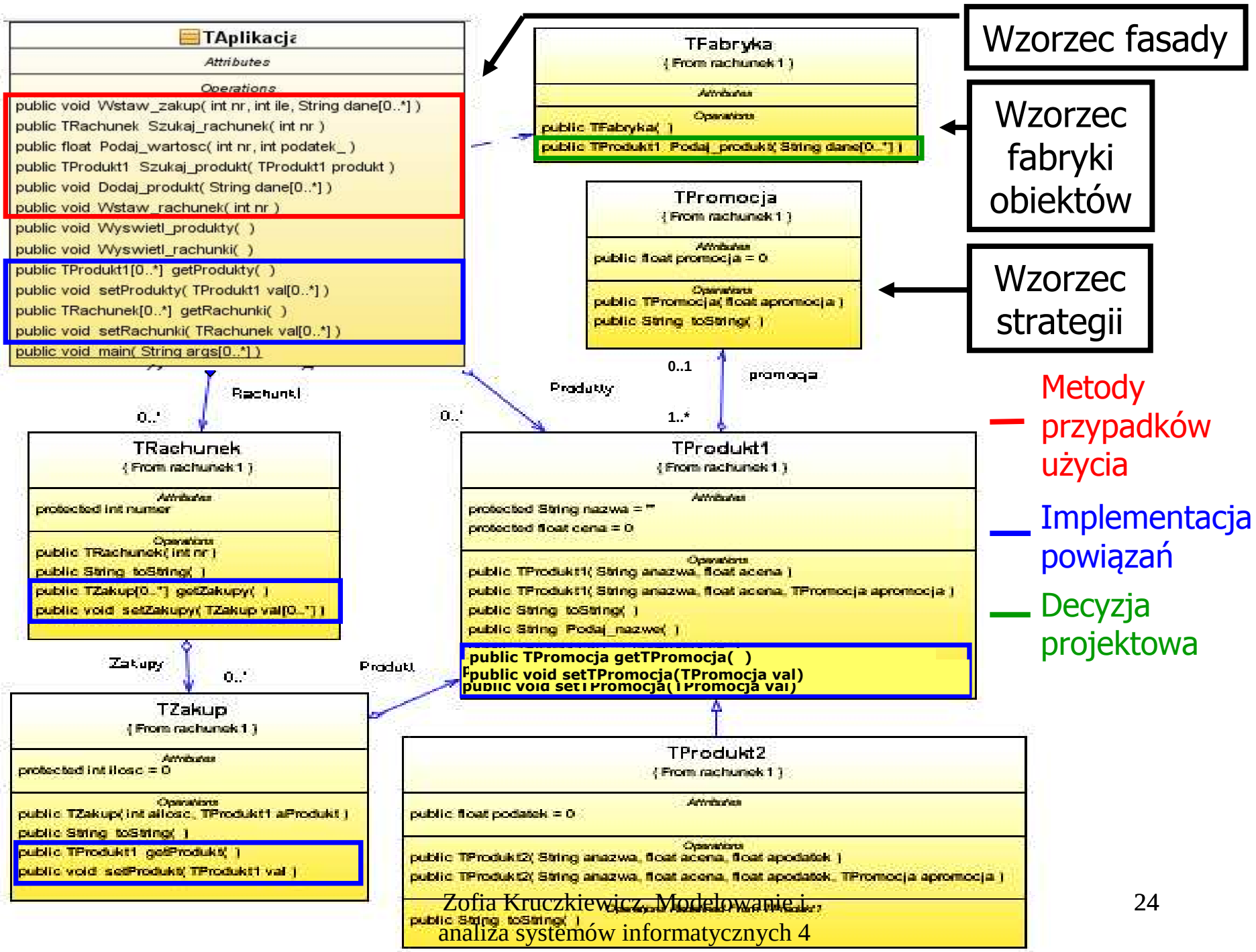
Analiza wspólności i zmienności (wykład 2)

- Wykryto trzy główne klasy typu „Entity” ze względu na odpowiedzialność: TRachunek (wstawia zakupy, oblicza wartość), TZakup (oblicza wartość zakupu) oraz TProdukt1 (posiada nazwę oraz oblicza cenę detaliczną)
- Wykryto dziedziczenie w właściwościach produktów, które podają cenę jednostkową podawaną jako cenę netto, jeśli produkt nie posiada atrybutu podatek lub cenę brutto, jeśli posiada atrybut podatek (klasa TProdukt2 typu „Entity”, która dziedziczy od klasy TProdukt1) oraz strategię zmniejszania ceny jednostkowej wynikającej z promocji powiązaną z produktem zarówno z podatkiem, jak bez podatku. Ponieważ jednak promocja nie musi dotyczyć każdego produktu, jest w związku powiązania z bazowym (głównym) produktem typu 0..* do 1. Klasa TPromocja typu „Entity” jest dziedziczona przez pozostałe typy produktu. Stad produkt powinien podawać uogólnioną cenę detaliczną: bez podatku, z podatkiem oraz w razie potrzeby z uwzględnieniem scenariusza dodawania promocji do ceny detalicznej produktu dla dwóch pierwszych przypadków (cztery typy ceny detalicznej).
- Wykryto związki silnej agregacji między rachunkiem i zakupami (rachunek posiada kolekcję zakupów) oraz słabej agregacji między zakupem a produktem (zakup składa się z produktu bazowego lub jego następców), oraz związek typu powiązanie między promocją a produktem bazowym dziedziczony przez produkty potomne.
- Zastosowano klasę fasadową TAplikacja typu „Control” do oddzielenia obiektów typu „Entity” od pozostałej części systemu oraz klasę typu „Control” jako fabrykę obiektów (TFabryka) do tworzenia różnych typów produktów

Diagram klas – koncepcja klas typu „Entity” oraz „Controller”




«entity»
TRachunek
Attributes
Operations
public TRachunek( )
public TZakup[0..*] getTZakup( )
public void setTZakup(TZakup val[0..*])
zarzadzanie rachunkami
0..*
1
«controller»
TAplikacja
Attributes
Operations
public TAplikacja( )
public TProdukt1[0..*] getTProdukt1( )
public void setTProdukt1(TProdukt1 val[0..*])
public TRachunek[0..*] getTRachunek( )
public void setTRachunek(TRachunek val[0..*])
1
posiada zakupy
0..*
1
zarzadzanie produktami
0..*
«entity»
TZakup
Attributes
Operations
public TZakup( )
public TProdukt1 getTProdukt( )
public void setTProdukt1(TProdukt1 val)
sklada sie
1
1
«entity»
TProdukt1
Attributes
public TProdukt1( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
0..*
sposob promowania
0..1
«entity»
TProdukt2
Attributes
Operations
public TProdukt2( )
«entity»
TPromocja
Attributes
Operations
public TPromocja( )


TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
Wzorzec fasady
Wzorzec fabryki obiektów
Wzorzec strategii
Rachunki
0..*
Produkty
0..*
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public String Podaj_nazwe( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
public String toString( )
Metody przypadków użycia
Implementacja powiązań
Decyzja projektowa

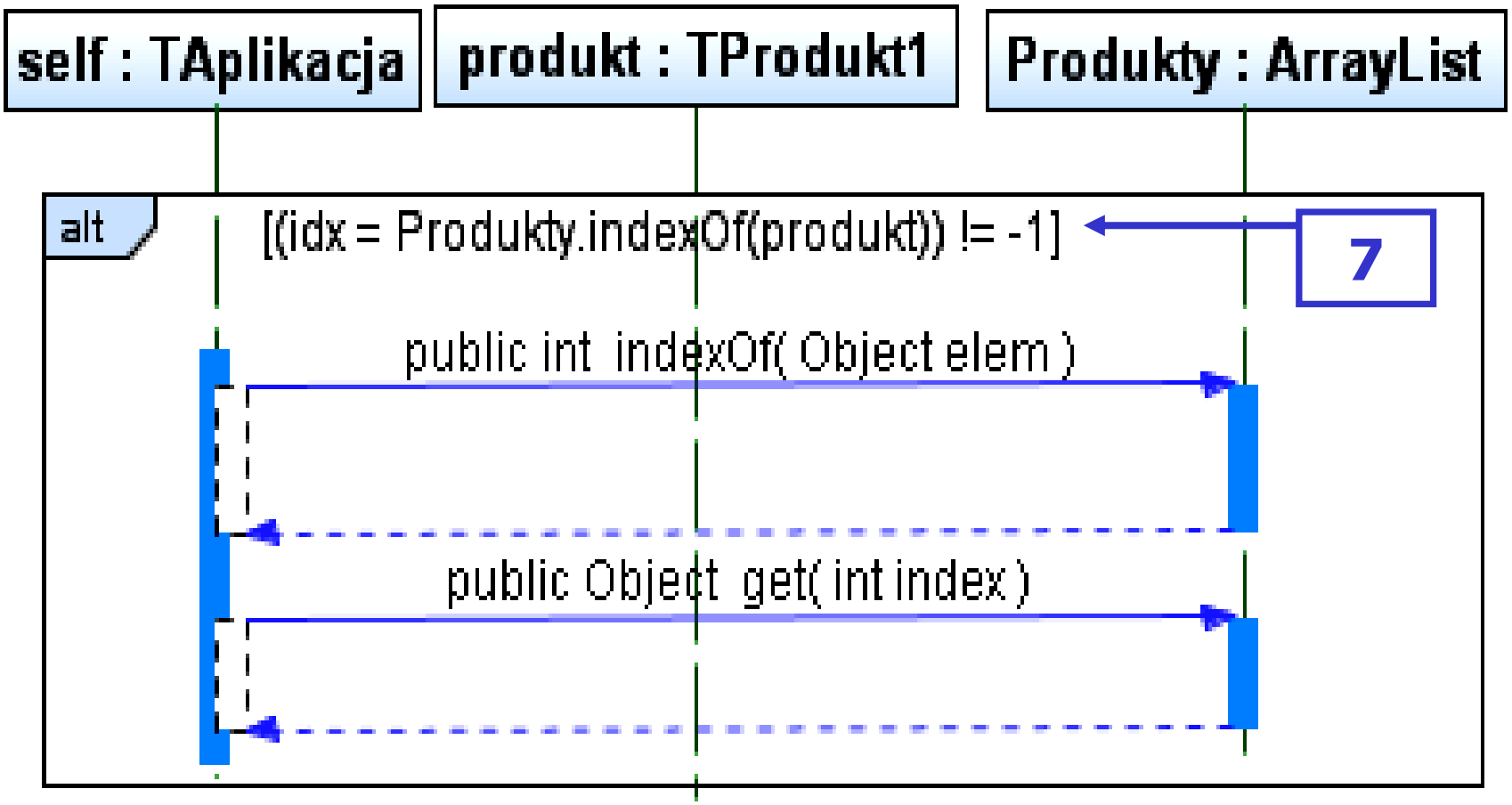
Projekt przypadku użycia

„ **Szukanie produktu**”

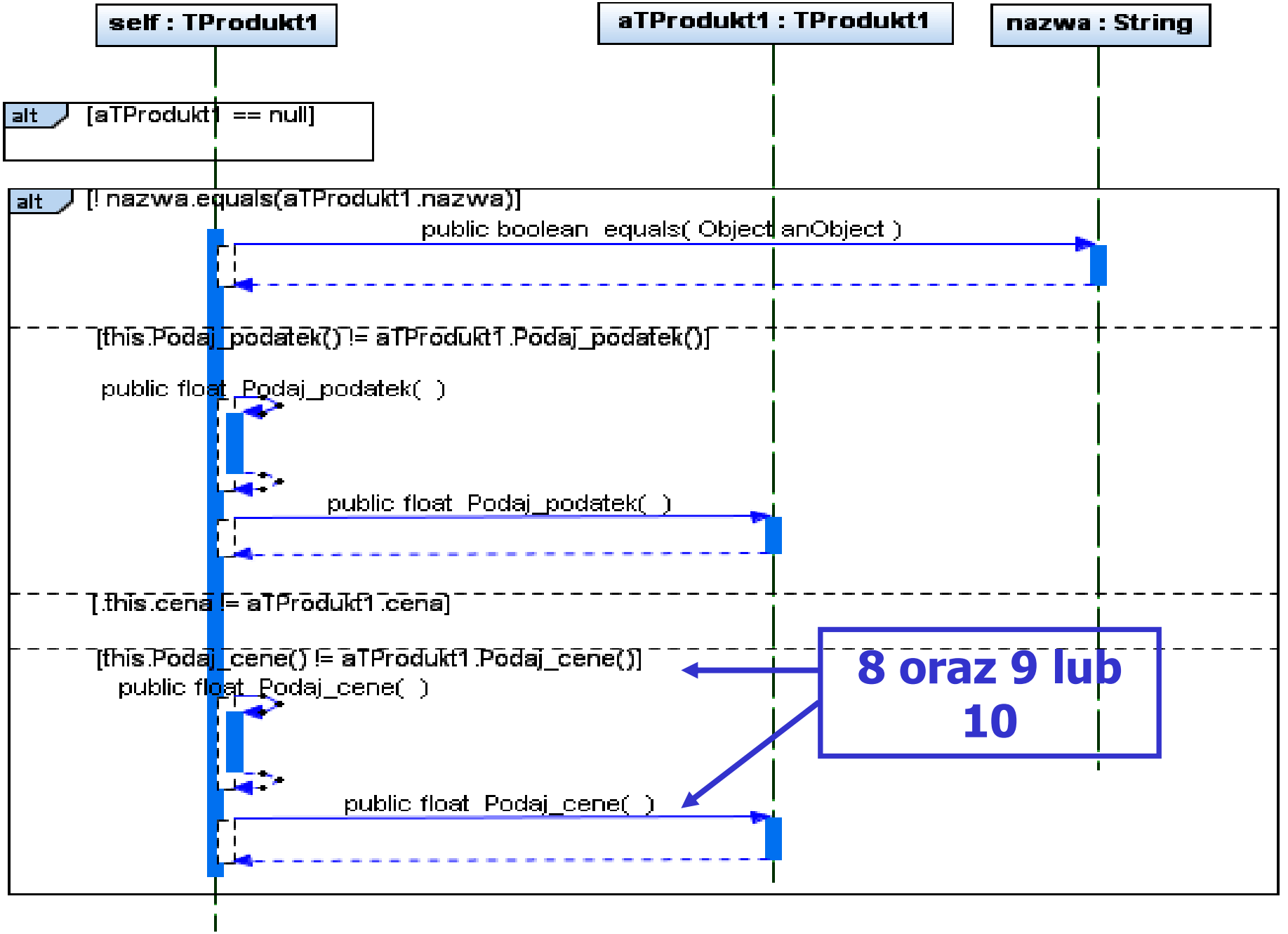
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

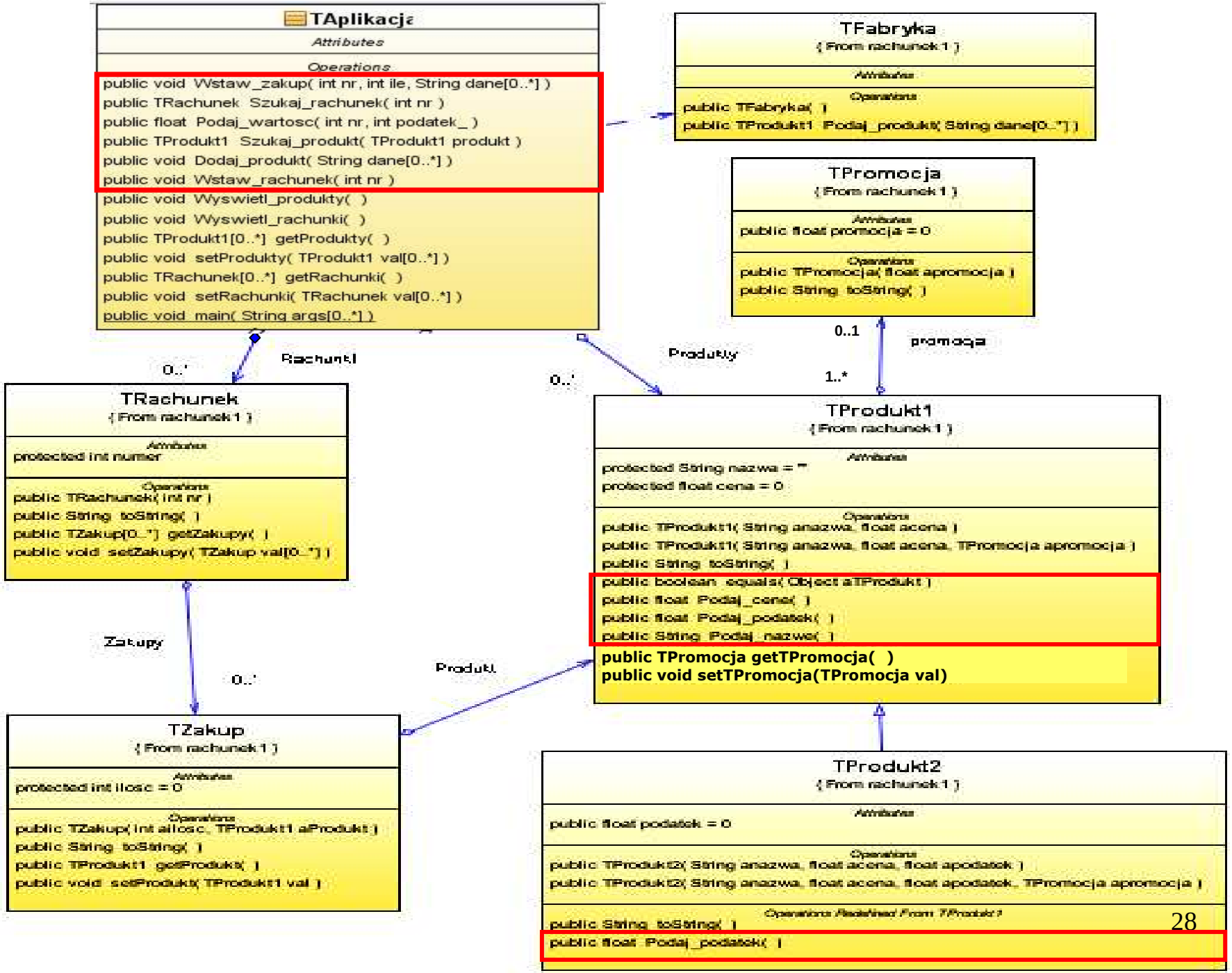
(1) Szukanie produktu

(TProdukt1 TAplikacja::Szukaj_produkt(TProdukt1 produkt))



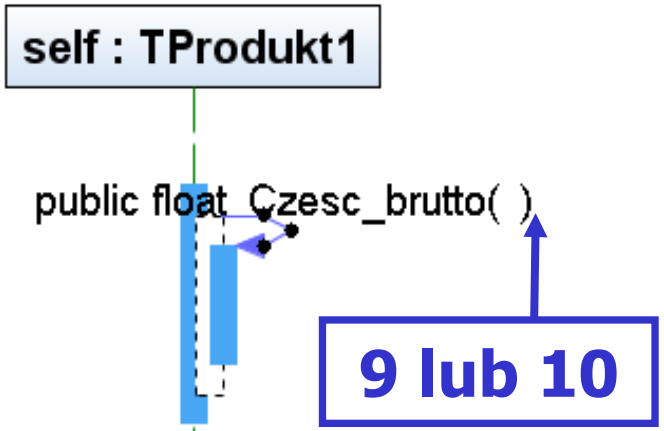
(7) boolean TProdukt1::equals(Object aTProdukt)




TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
0..1
promocja
1..*
Produkty
0..*
Rachunki
0..*
TRachunek
{ From rachunek1 }
Attributes
protected int number
Operations
public TRachunek(int nr)
public String toString( )
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Podaj_podatek( )

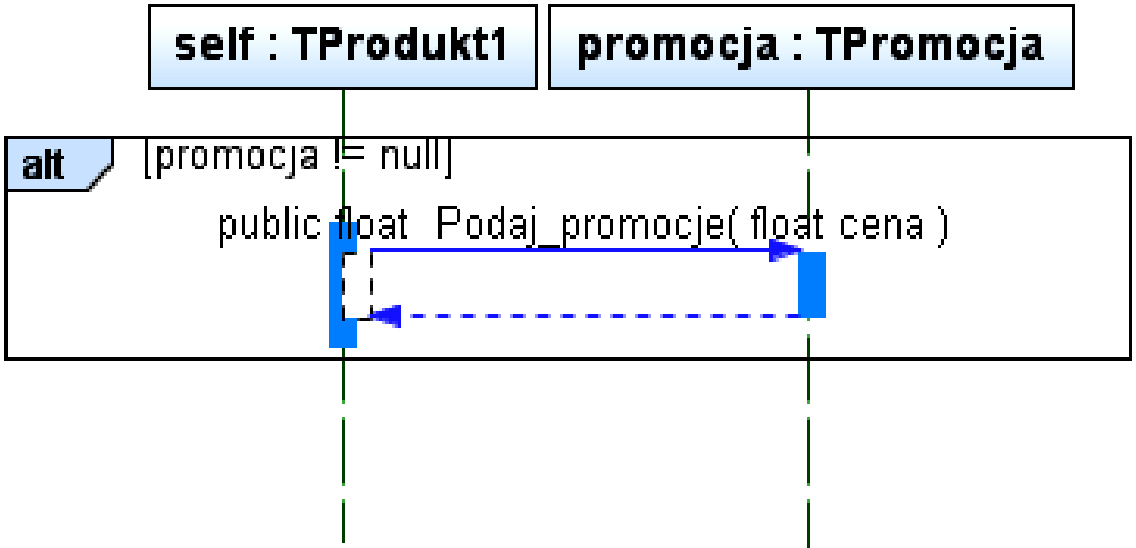
(8)

float TProdukt1::Podaj_cene()



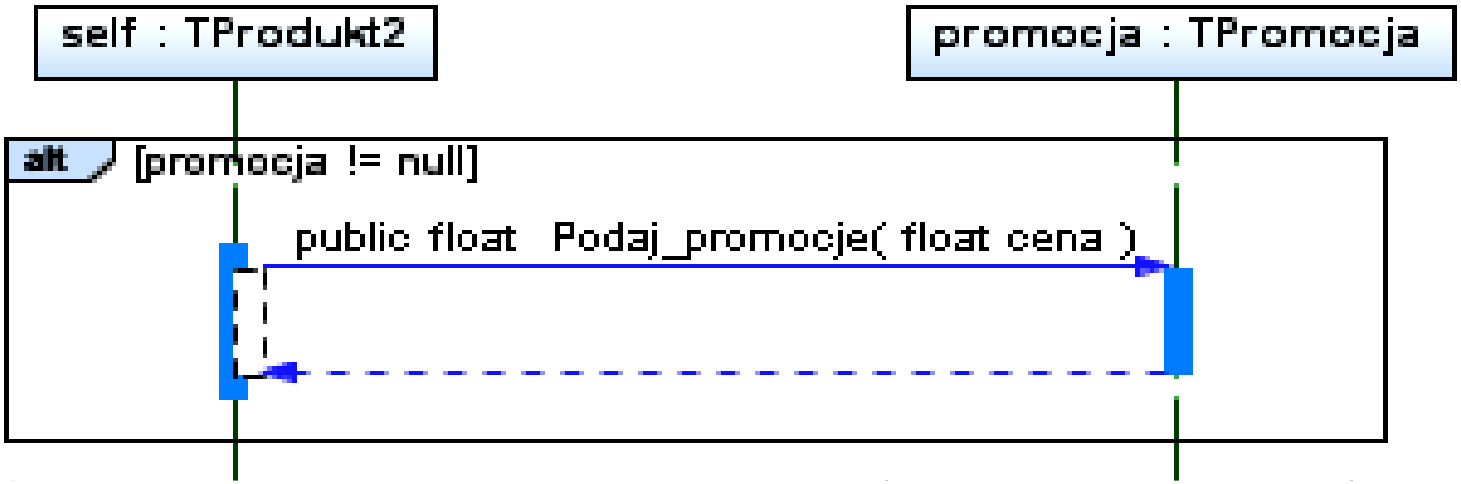
(9)

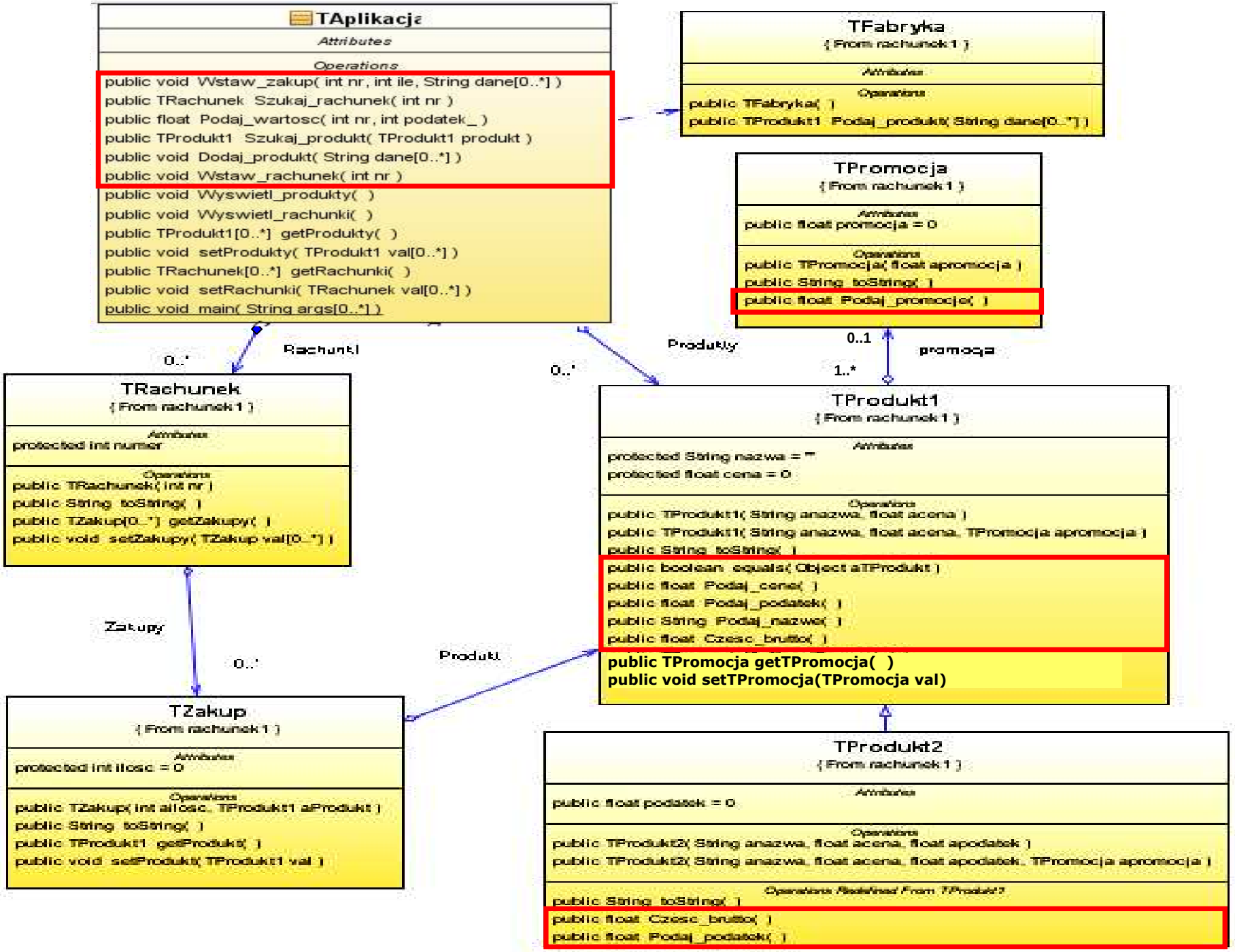
float TProdukt1::Czesc_brutto()



(10)

float TProdukt2::Czesc_brutto()




TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
public float Podaj_promocje( )
0..1
promocja
1..*
Produkty
0..*
Rachunki
0..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )

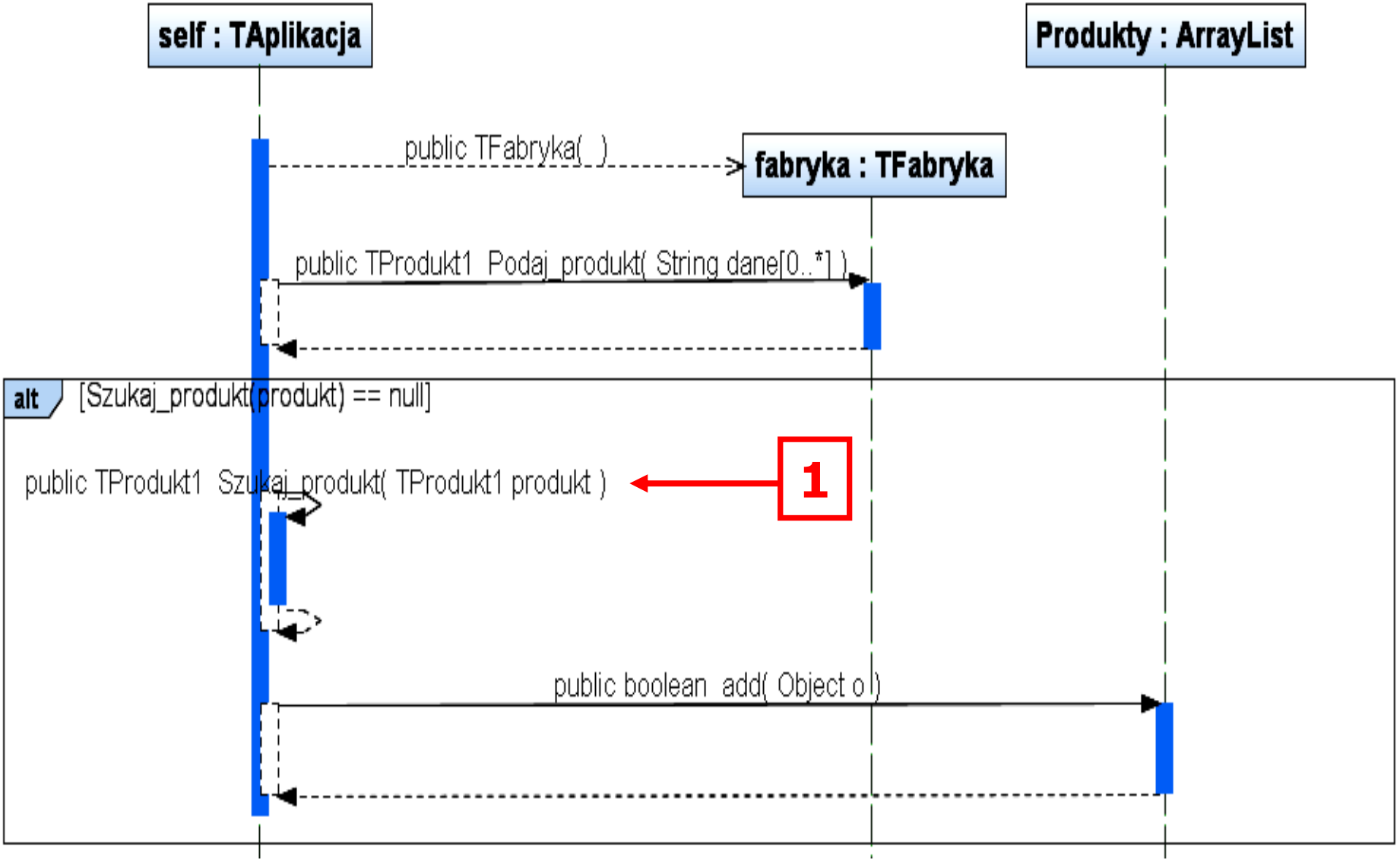
Projekt przypadku użycia

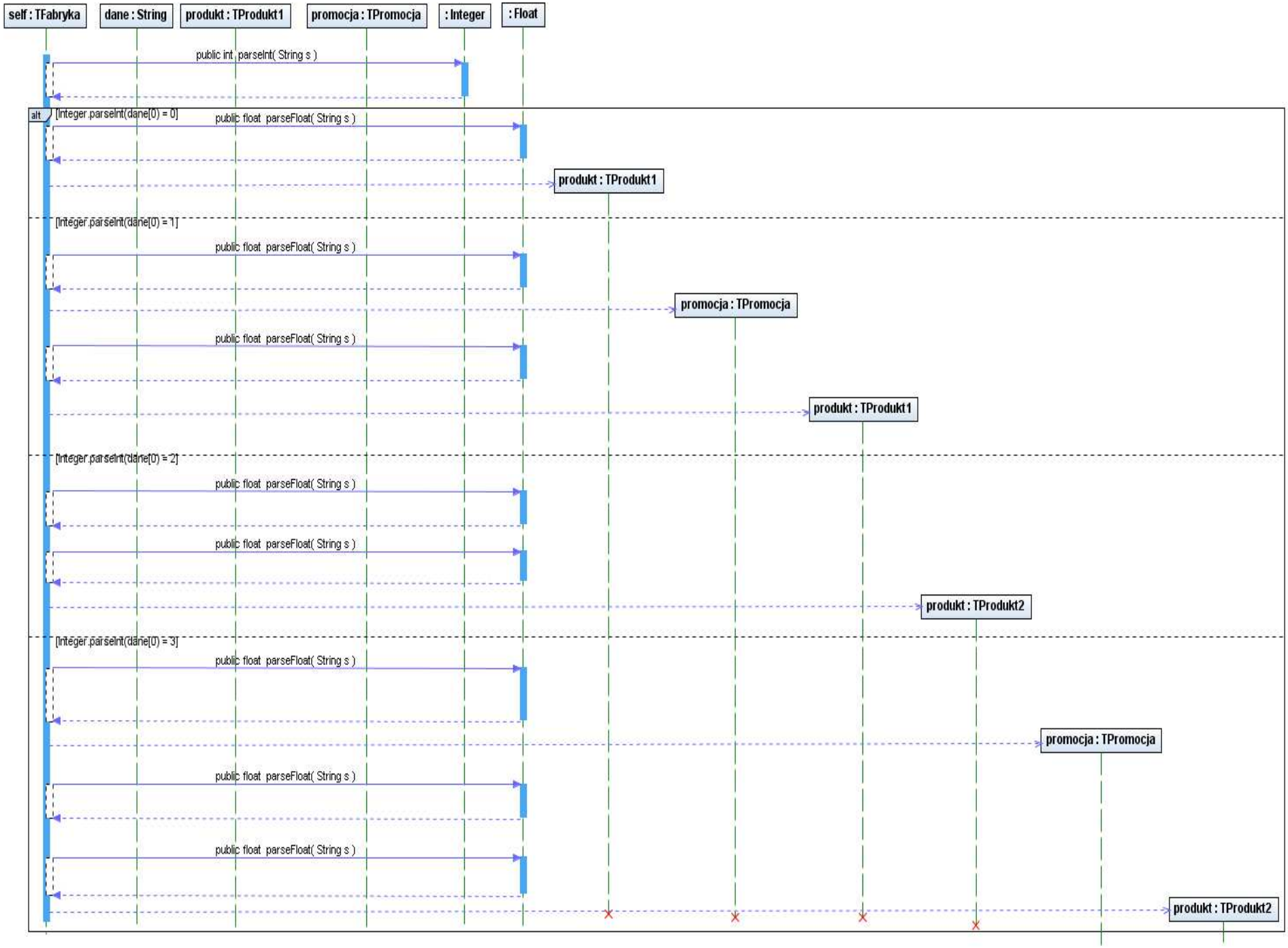
„ **Wstawianie nowego produktu**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

(2) Wstawianie nowego produktu

(void TAplikacja::Dodaj_produkt(String [] dane))




self : TFabryka
dane : String
produkt : TProdukt1
promocja : TPromocja
: Integer
: Float
public int parseInt(String s)
alt
[Integer.parseInt(dane[0]) = 0]
public float parseFloat(String s)
produkt : TProdukt1
[Integer.parseInt(dane[0]) = 1]
public float parseFloat(String s)
promocja : TPromocja
public float parseFloat(String s)
produkt : TProdukt1
[Integer.parseInt(dane[0]) = 2]
public float parseFloat(String s)
public float parseFloat(String s)
produkt : TProdukt2
[Integer.parseInt(dane[0]) = 3]
public float parseFloat(String s)
promocja : TPromocja
public float parseFloat(String s)
public float parseFloat(String s)
produkt : TProdukt2

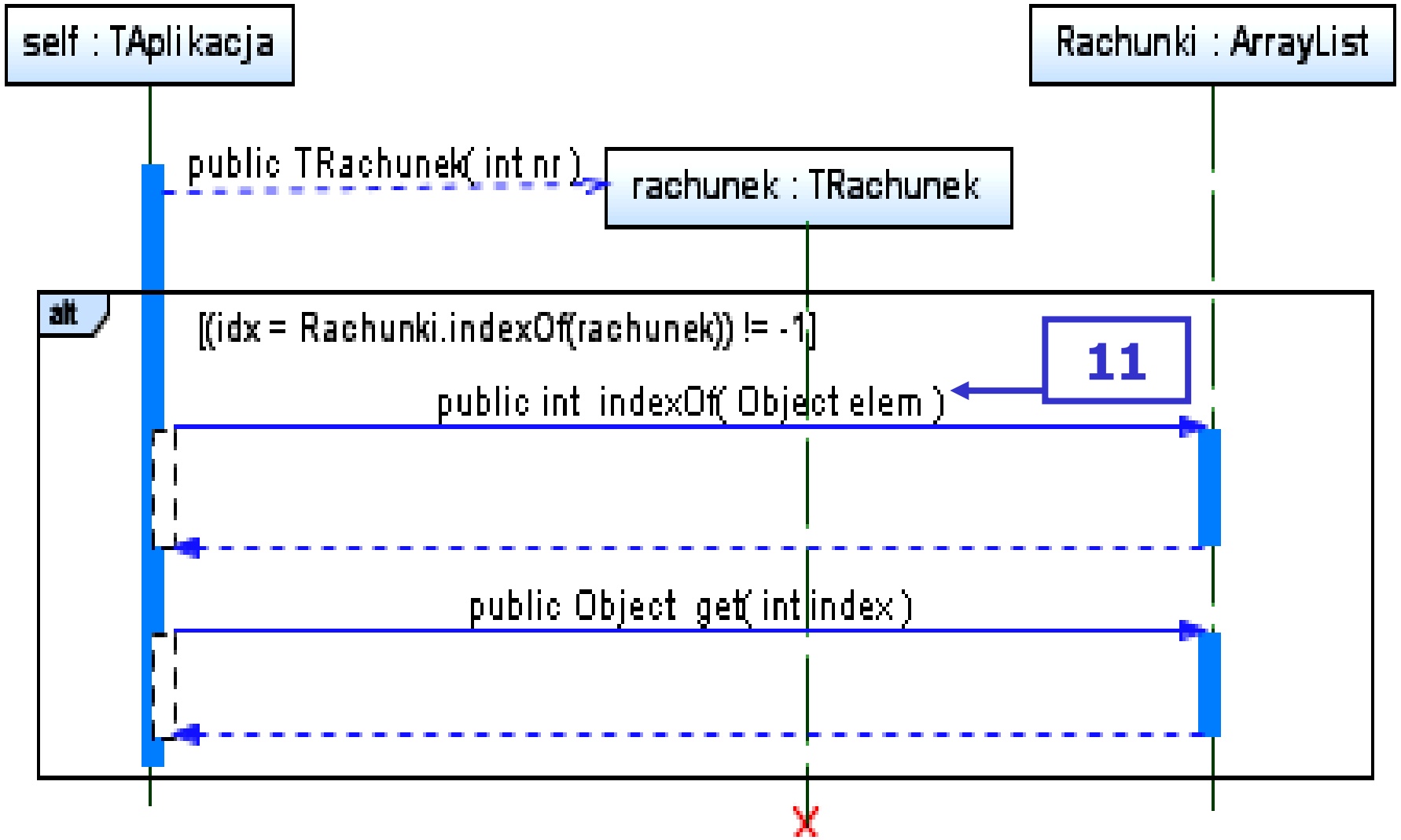
Projekt przypadku użycia

„ Szukanie rachunku ”

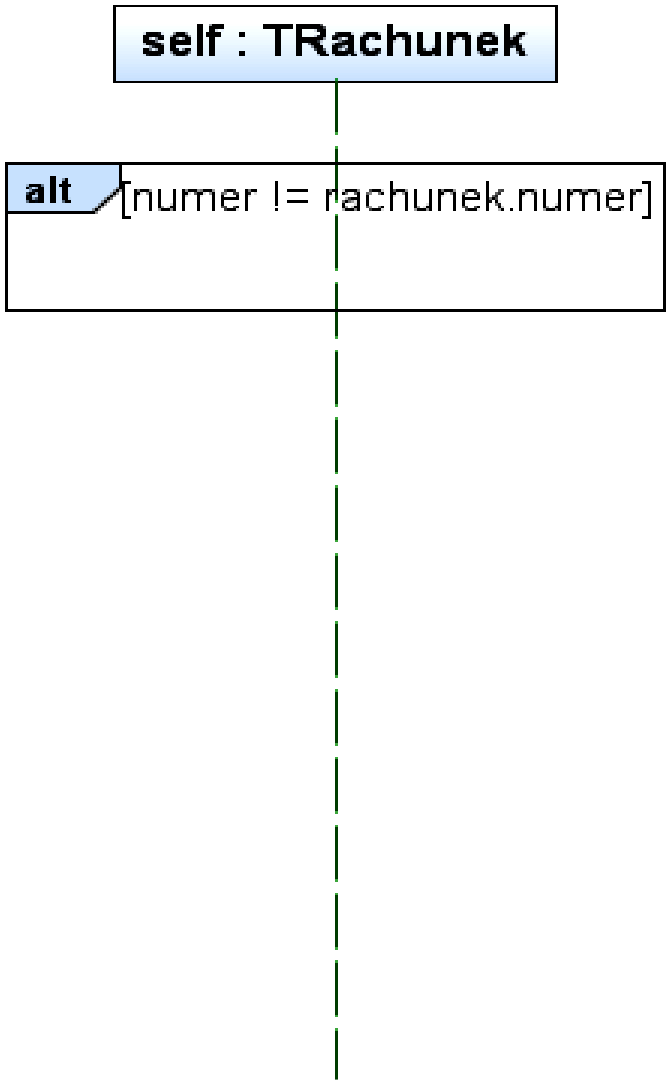
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

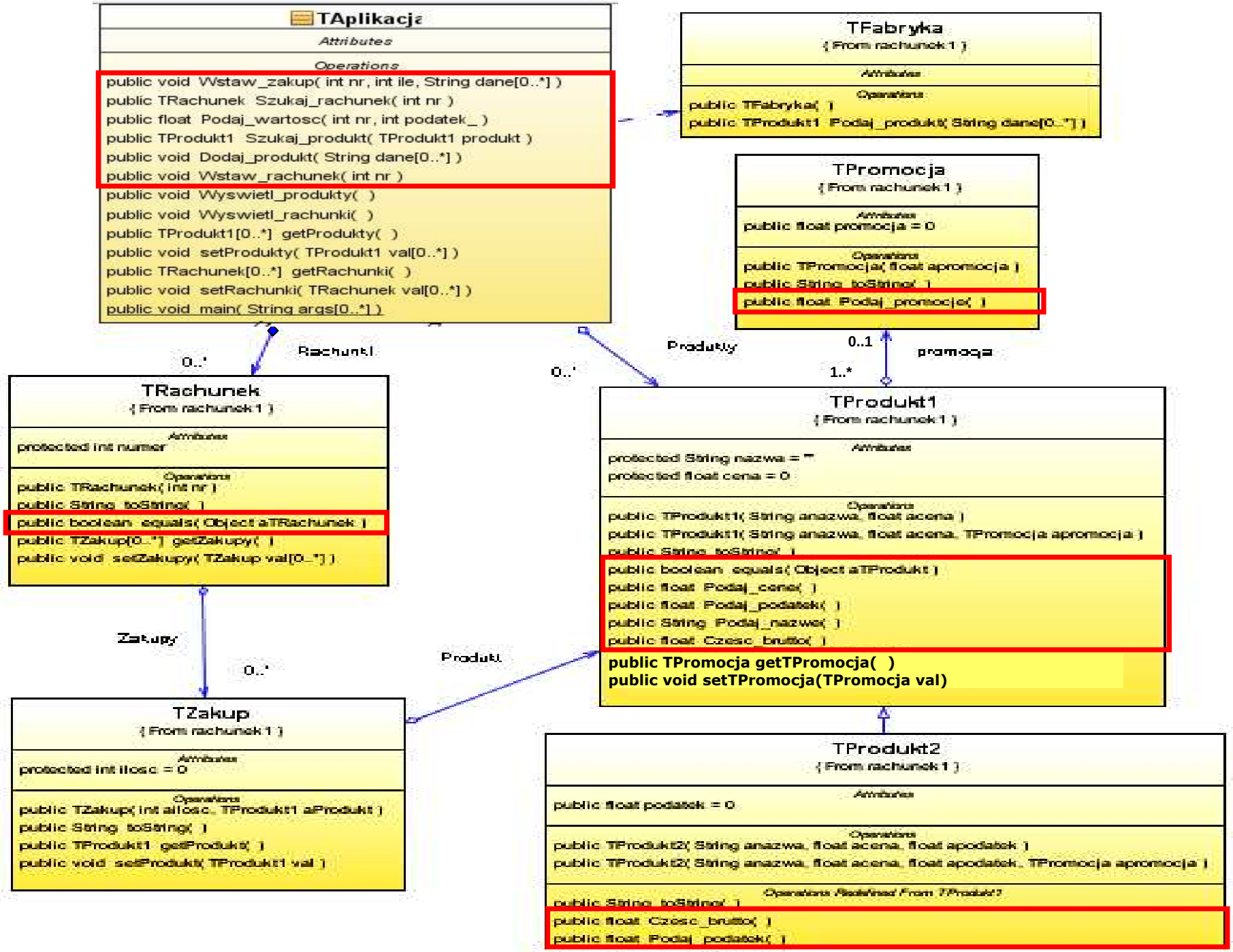
(3) Szukanie rachunku

(TRachunek TAplikacja::Szukaj_rachunek(int nr))



(11) boolean TRachunek::equals(Object rachunek)




TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
public float Podaj_promocje( )
Rachunki
0..*
0..*
Produkty
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public boolean equals(Object aTRachunek)
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )

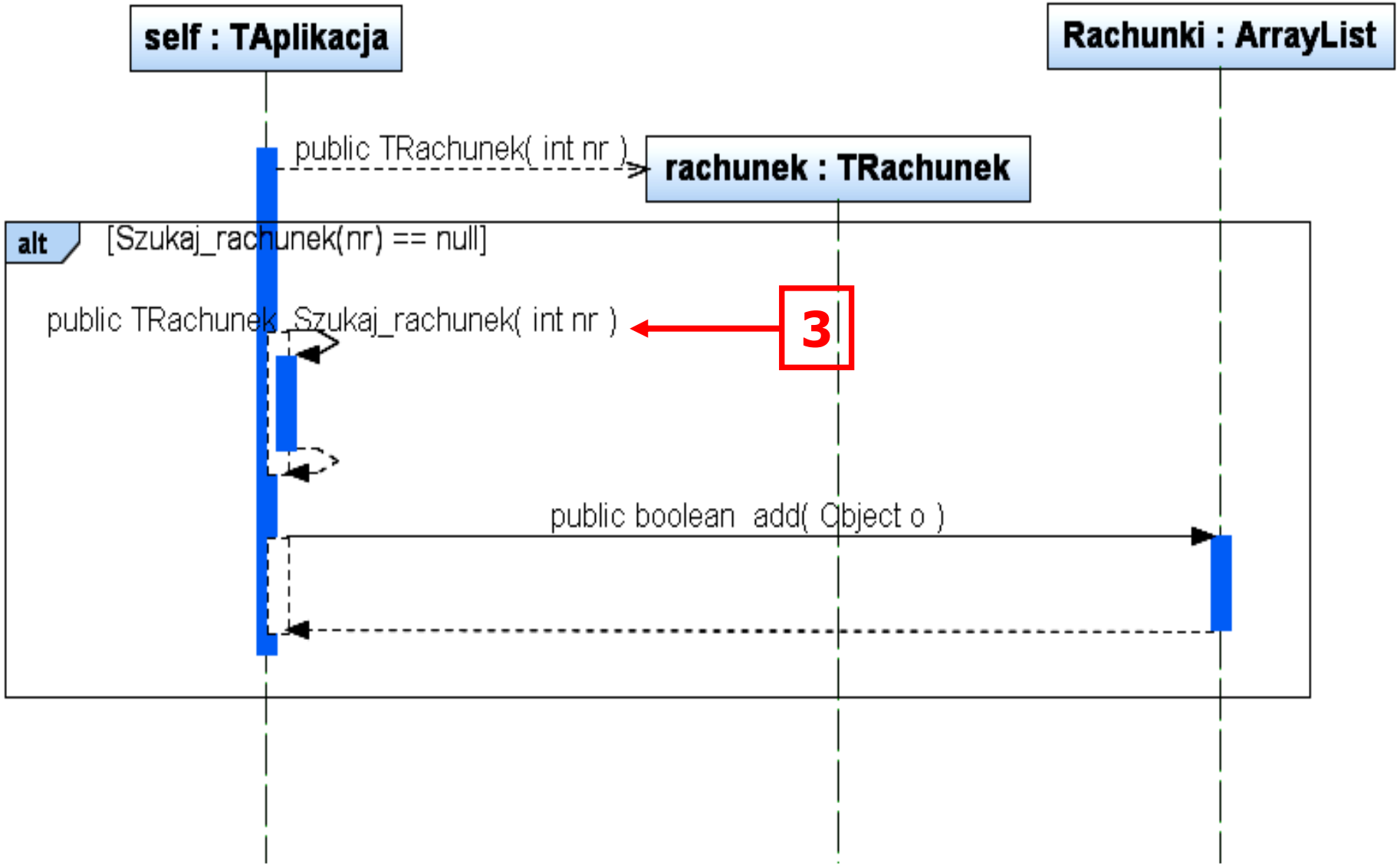
Projekt przypadku użycia

„ **Wstawianie nowego rachunku**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

(4) Wstawianie nowego rachunku

(void TAplikacja::Wstaw_rachunek(int nr))



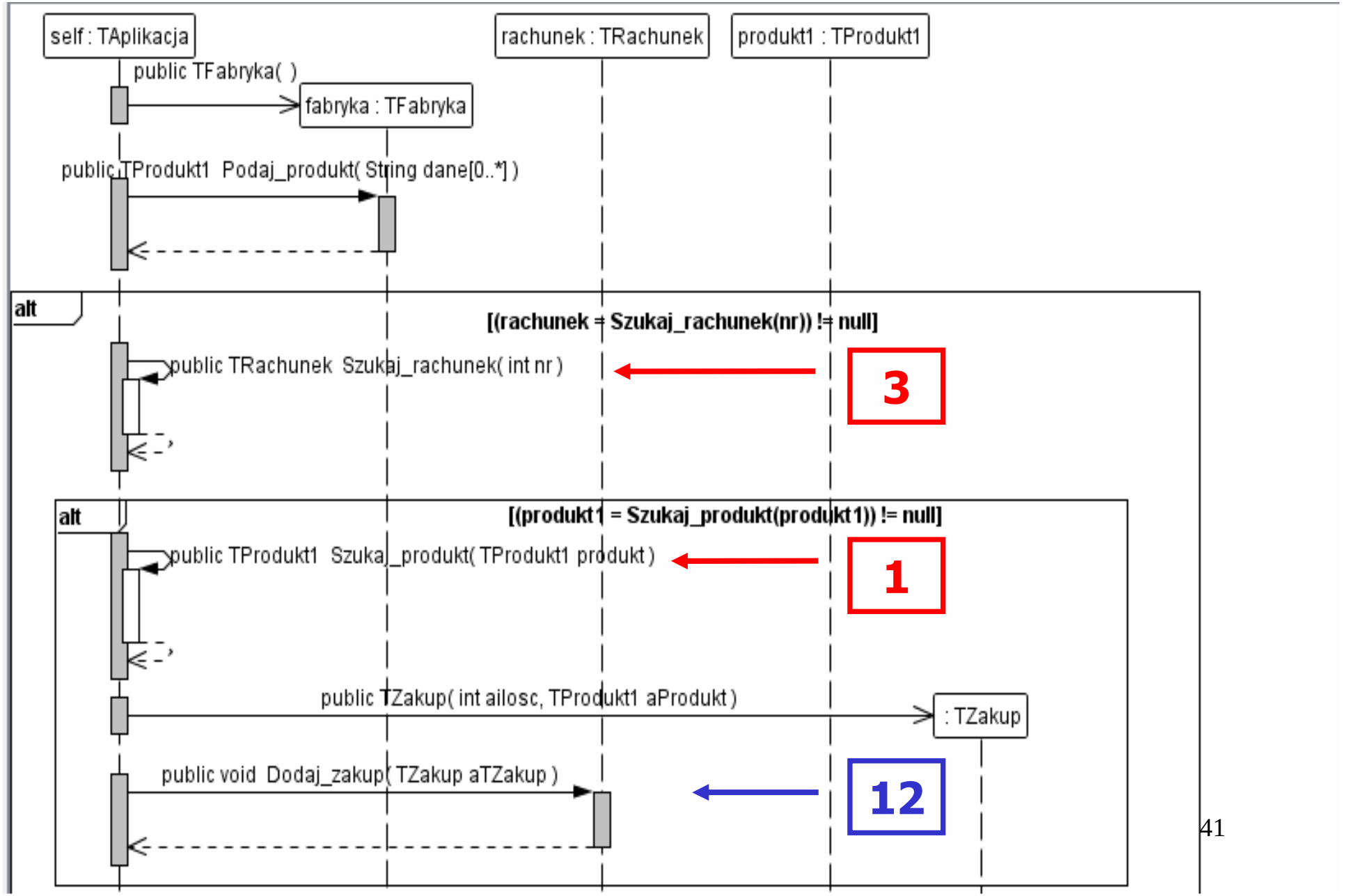
Projekt przypadku użycia

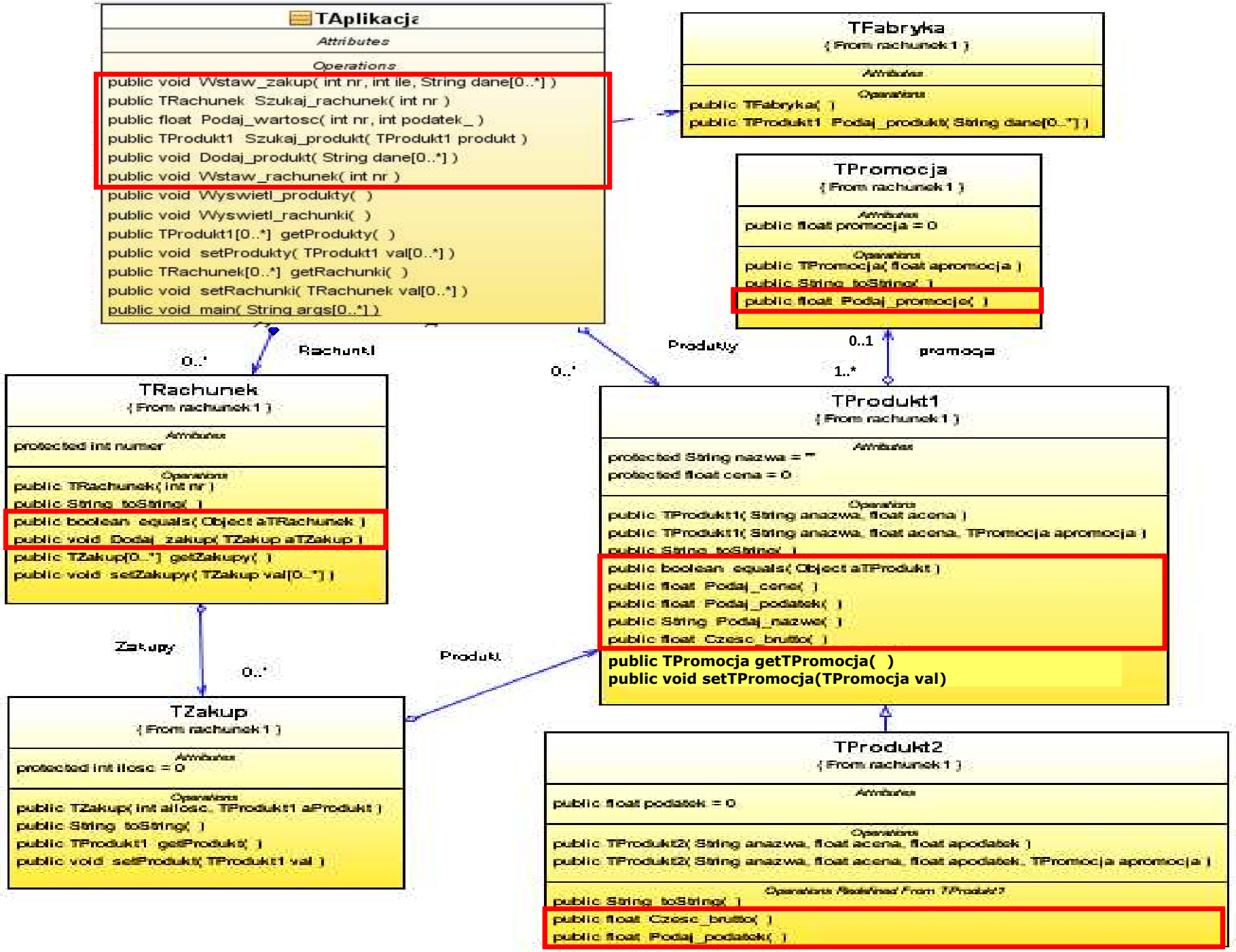
„**Wstawianie nowego zakupu**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

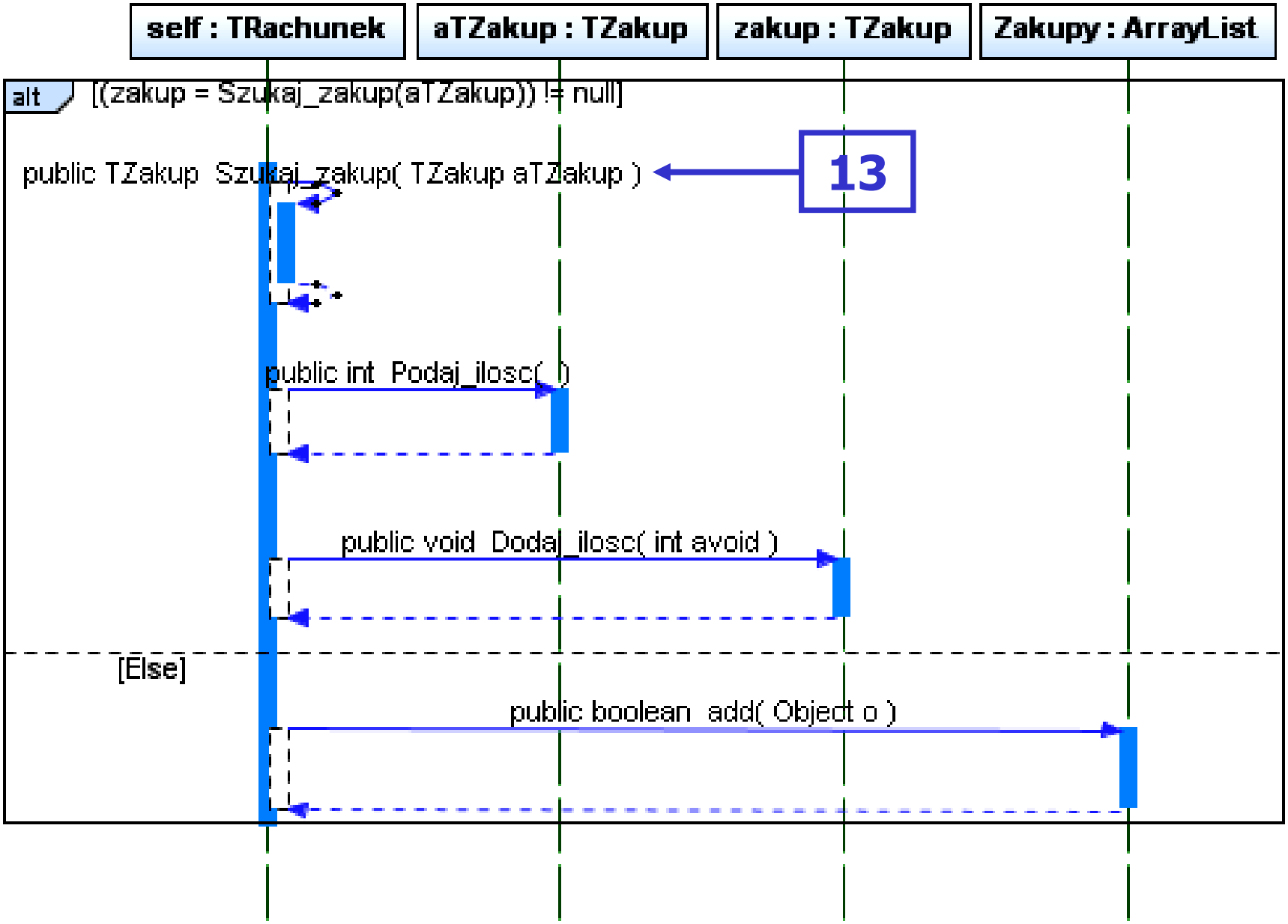
(5) Wstawianie nowego zakupu

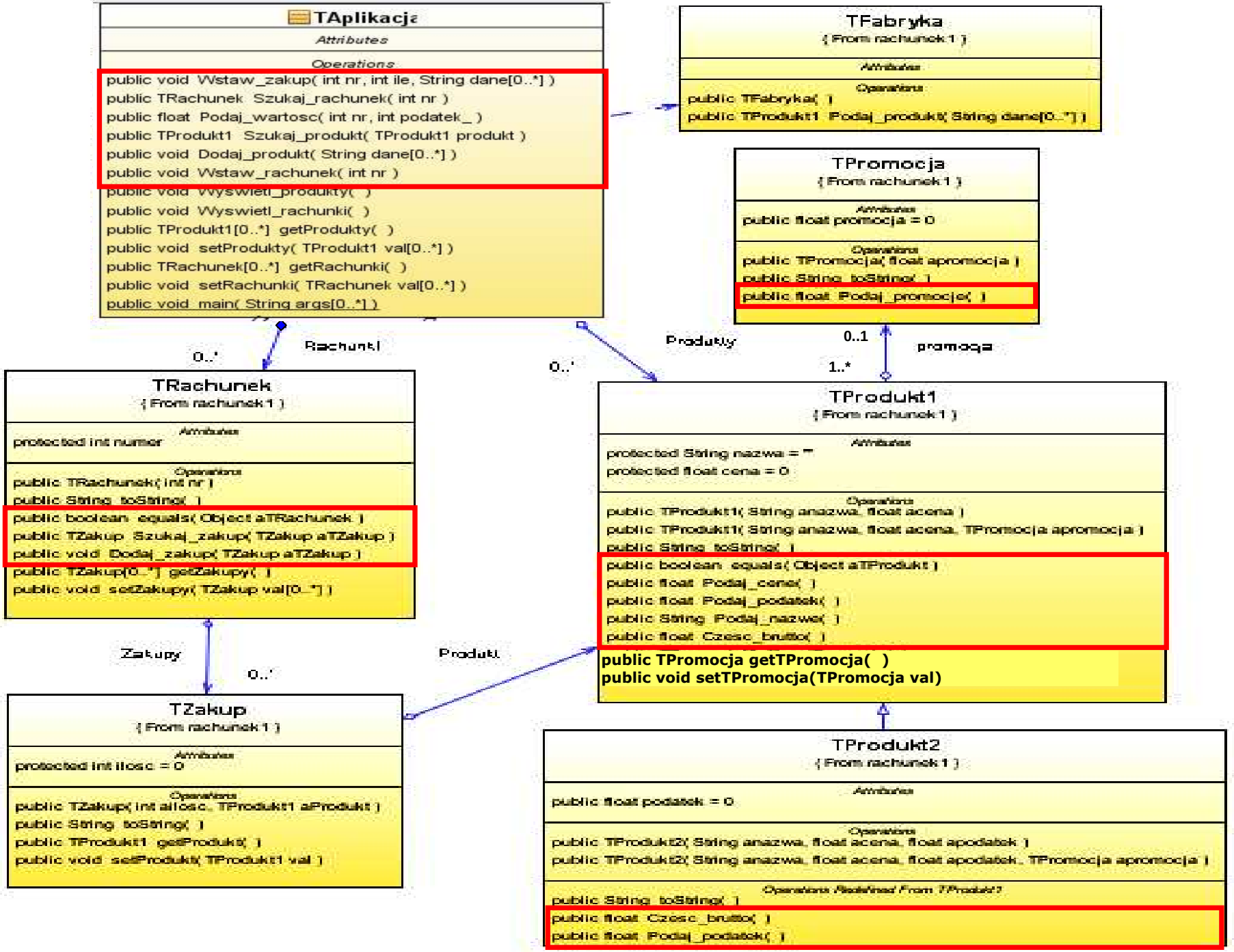
(void TAplikacja::Wstaw_zakup (int nr, int ailosc, String dane[]))



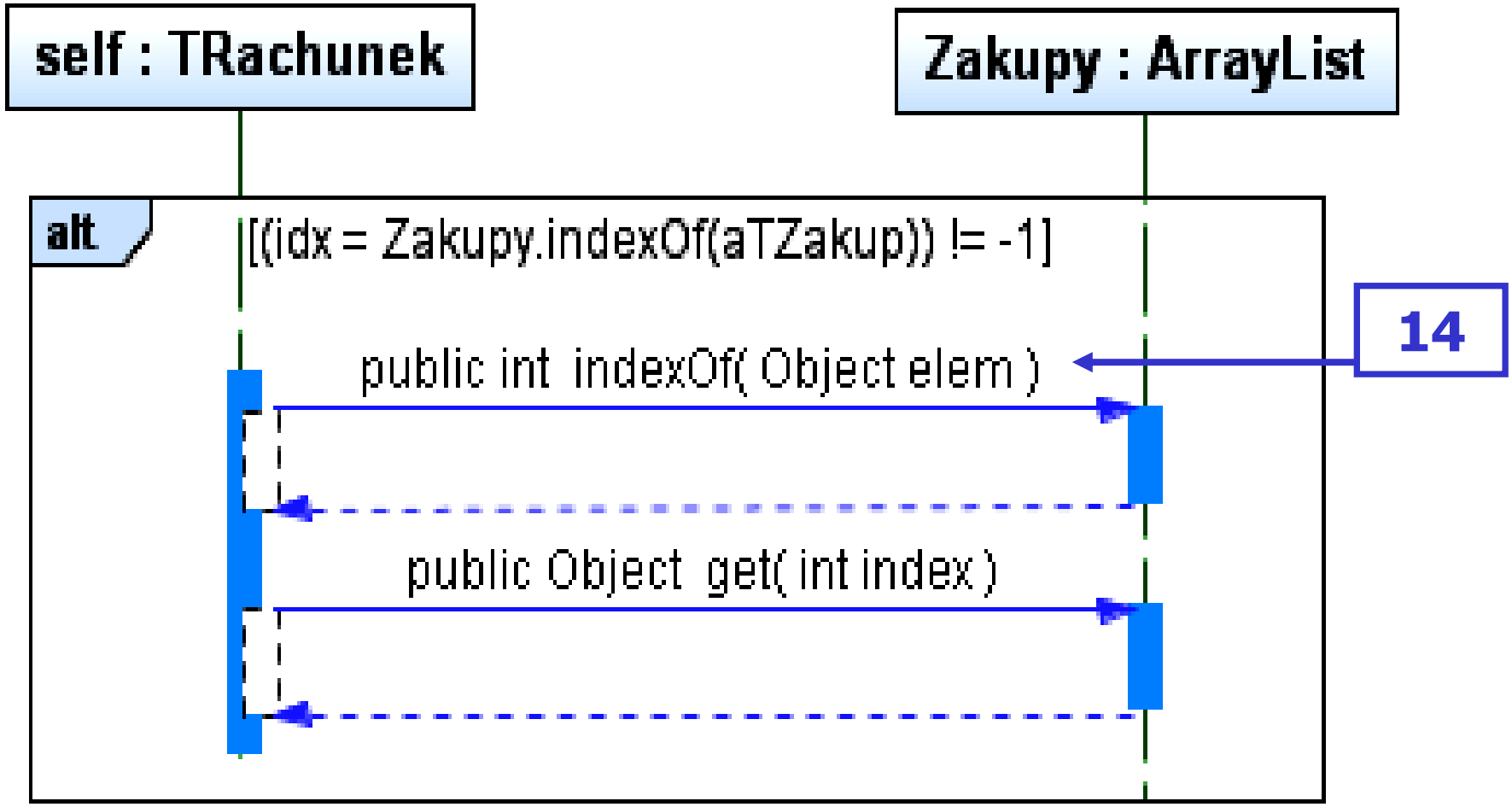

TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
public float Podaj_promocje( )
Rachunki
0..*
Produkty
0..*
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public boolean equals(Object aTRachunek)
public void Dodaj_zakup(TZakup aTZakup)
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )

(12) void TRachunek::Dodaj zakup(TZakup aTZakup)

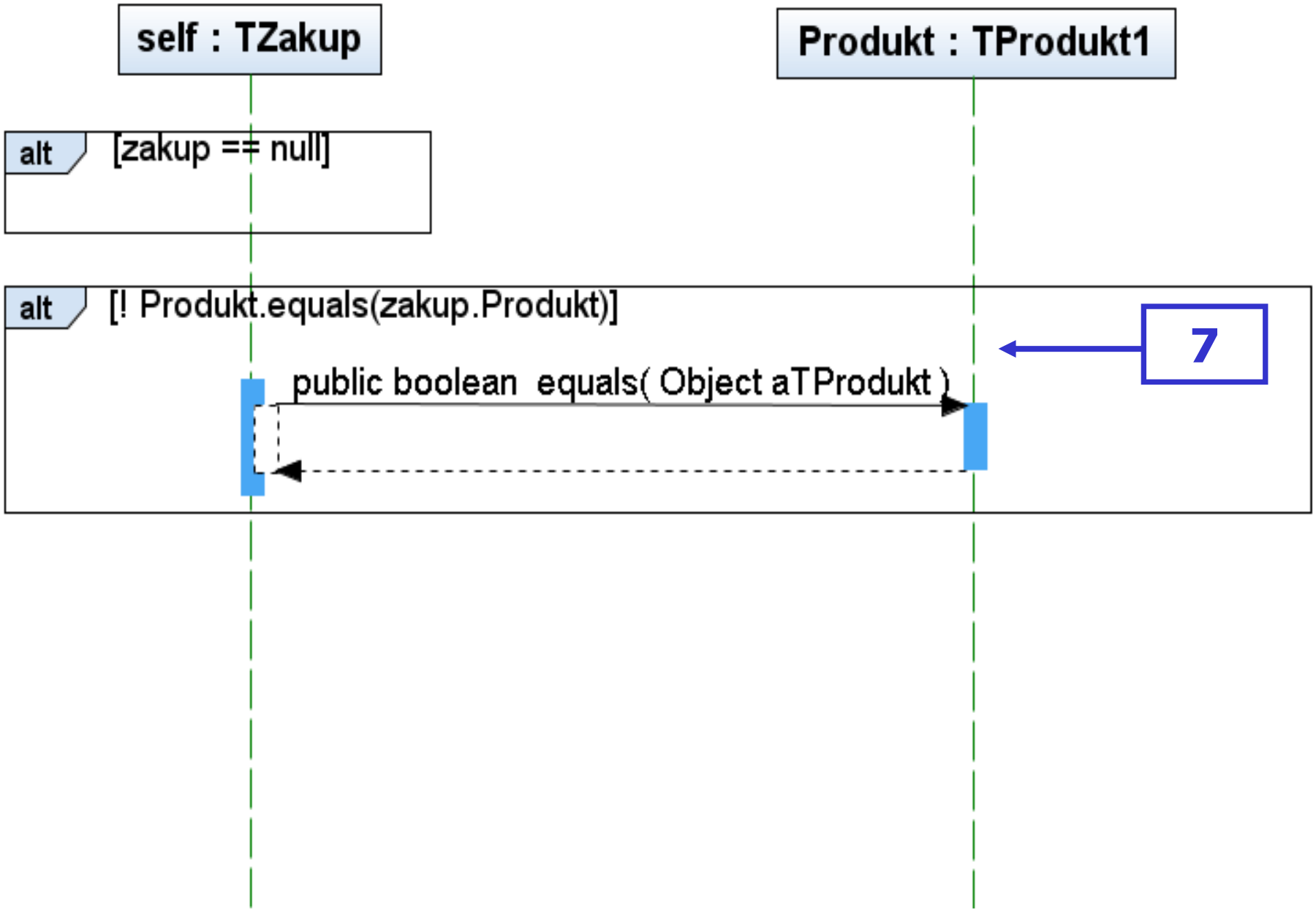


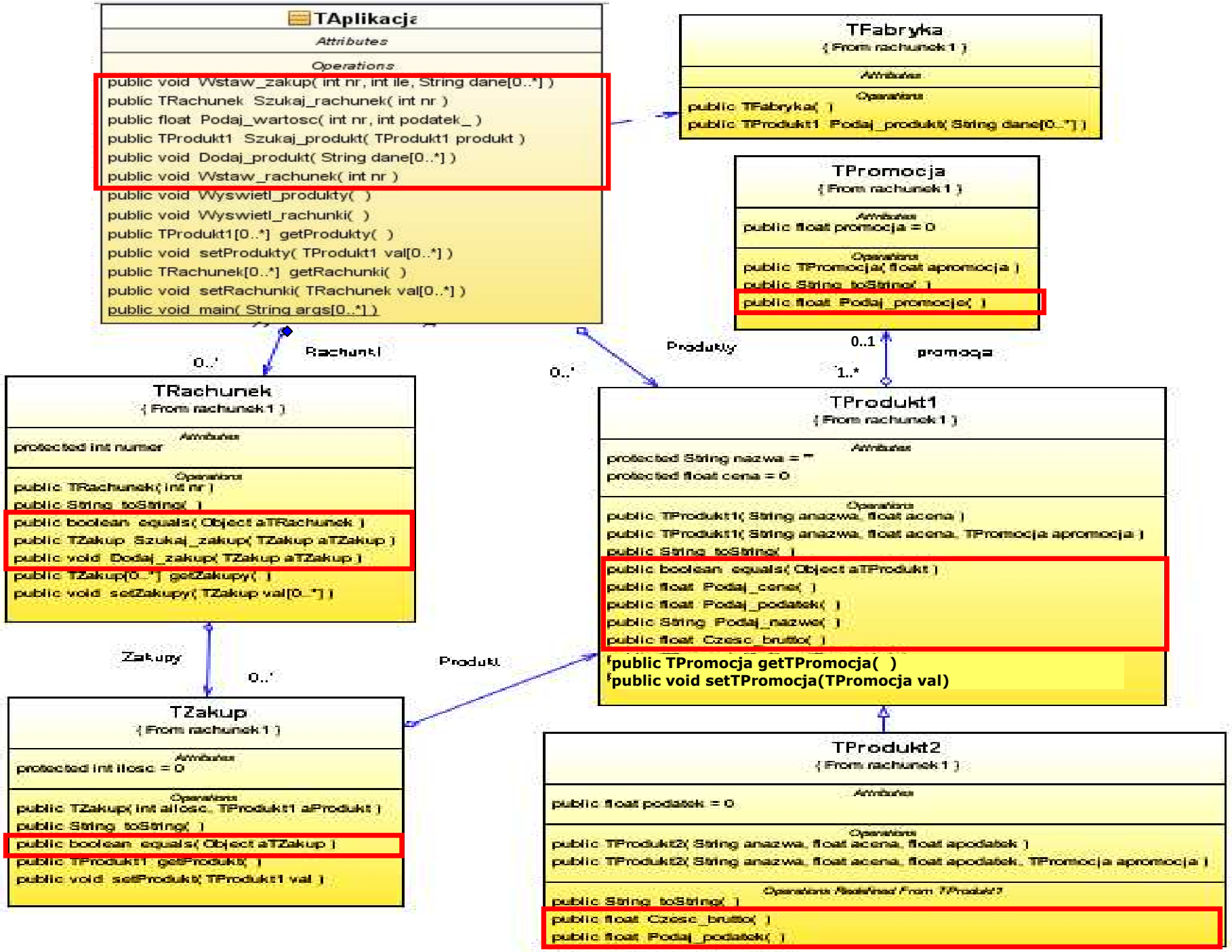

TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
public float Podaj_promocje( )
Rachunki
0..*
0..*
Produkty
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public boolean equals(Object aTRachunek)
public TZakup Szukaj_zakup(TZakup aTZakup)
public void Dodaj_zakup(TZakup aTZakup)
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )

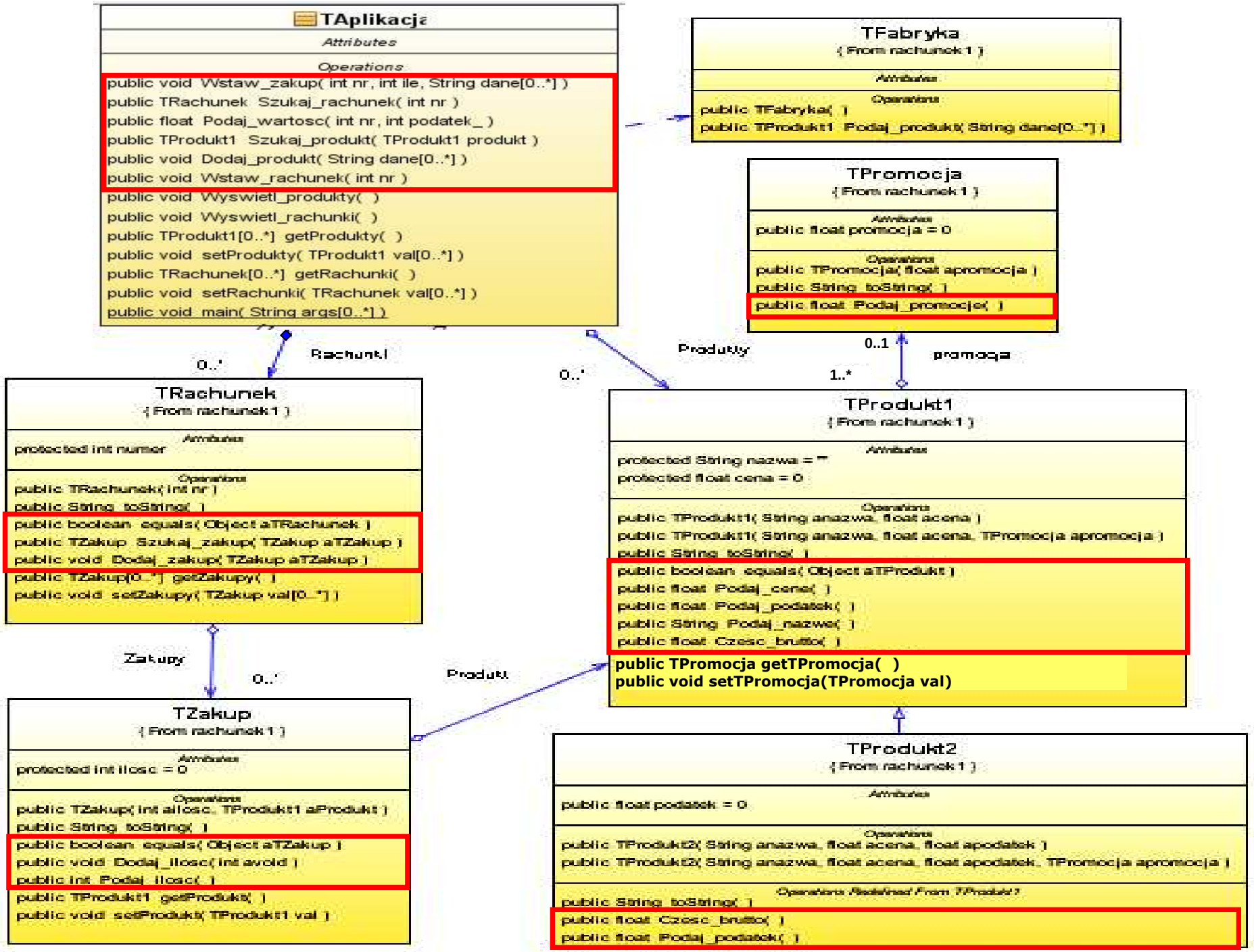
(13) TZakup TRachunek::Szukaj_zakup(TZakup aTZakup)



(14) boolean TZakup::equals(Object zakup)




TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
public float Podaj_promocje( )
Rachunki
0..*
0..*
Produkty
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public boolean equals(Object aTRachunek)
public TZakup Szukaj_zakup(TZakup aTZakup)
public void Dodaj_zakup(TZakup aTZakup)
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public boolean equals(Object aTZakup)
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )


TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
public float Podaj_promocje( )
Rachunki
0..*
0..*
Produkty
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public boolean equals(Object aTRachunek)
public TZakup Szukaj_zakup(TZakup aTZakup)
public void Dodaj_zakup(TZakup aTZakup)
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public boolean equals(Object aTZakup)
public void Dodaj_ilosc(int avoid)
public int Podaj_ilosc( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )

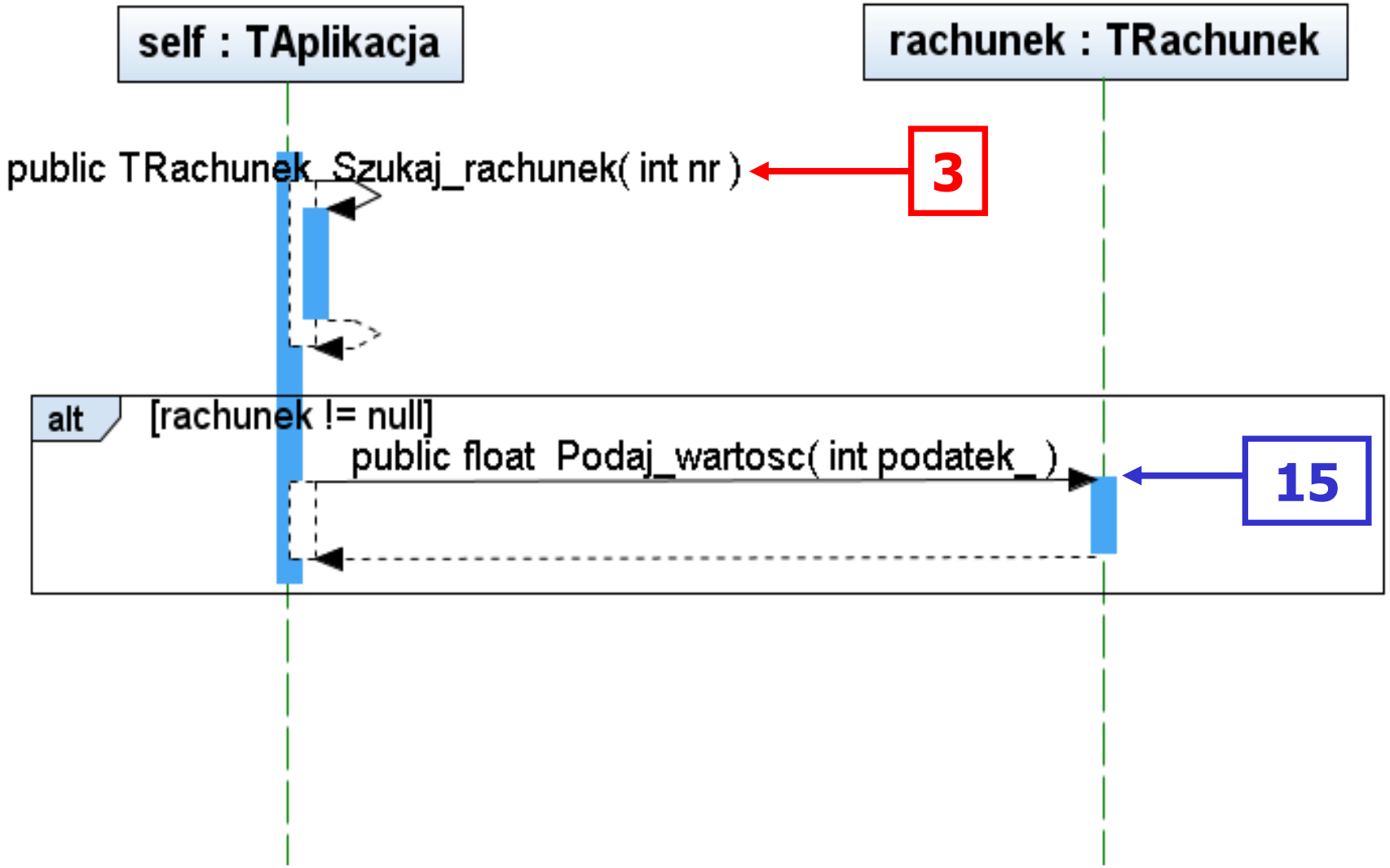
Projekt przypadku użycia

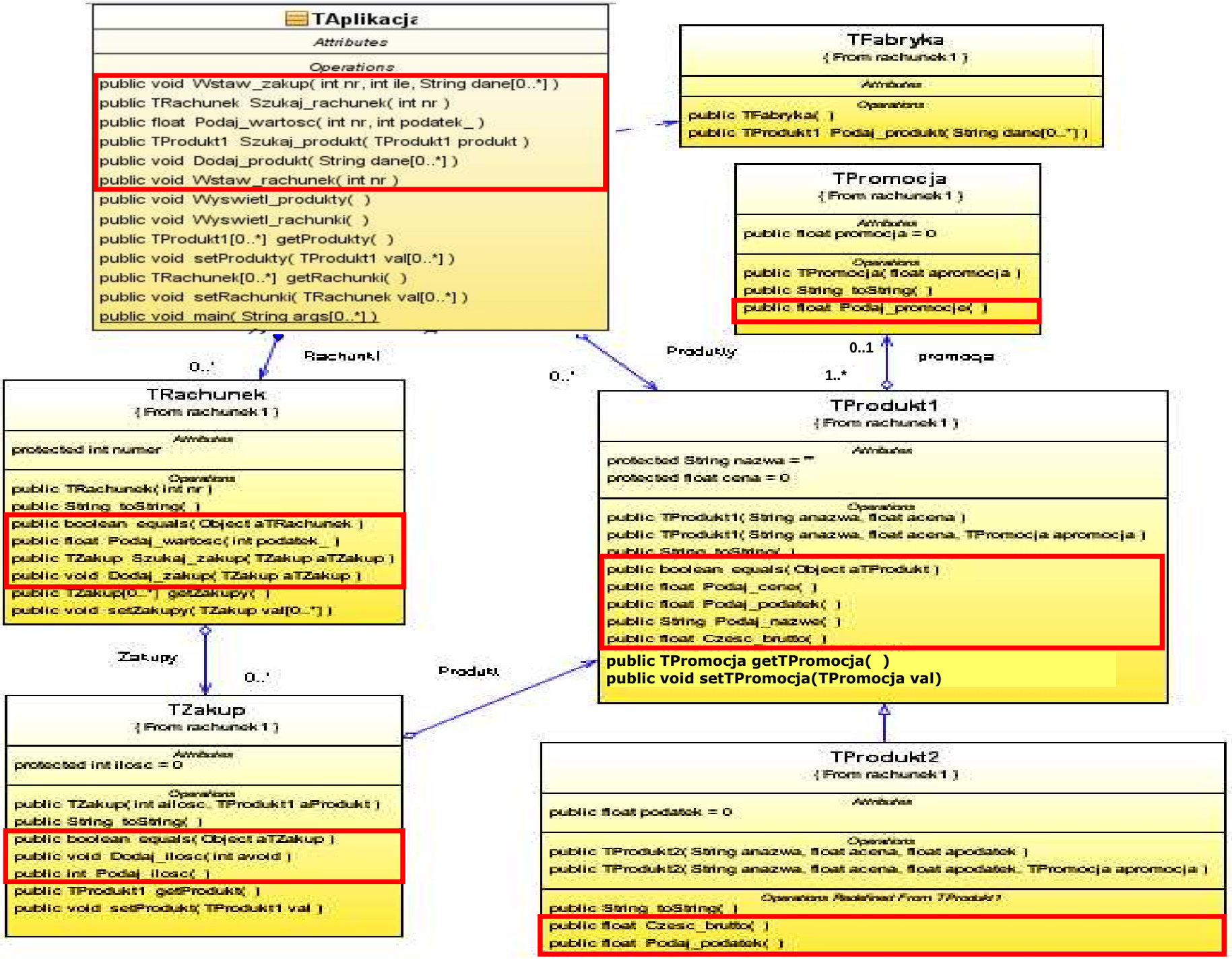
„Obliczanie wartości rachunku”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

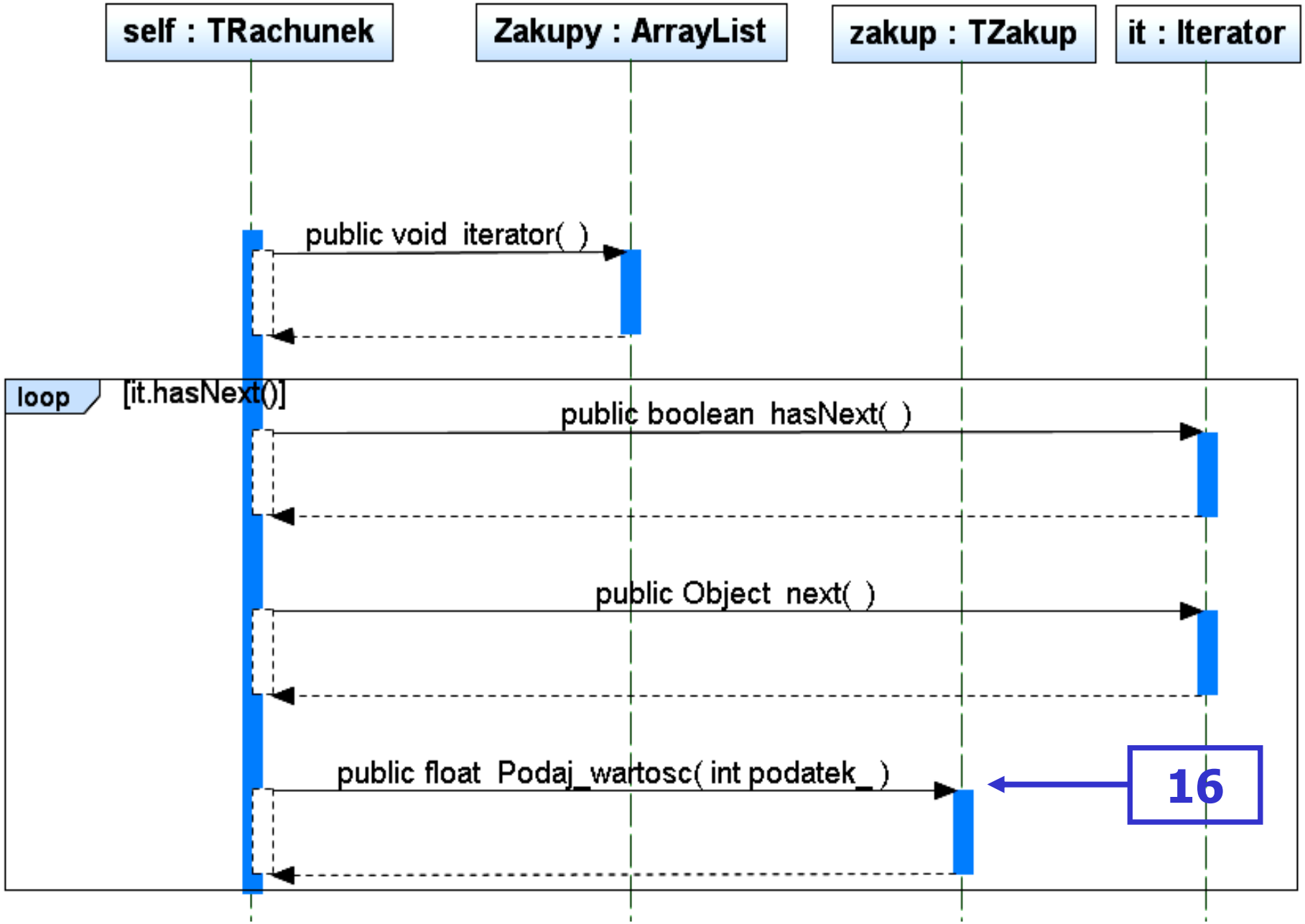
(6) Obliczanie wartosci rachunku

(float TAplikacja::Podaj_wartosc(int nr, int podatek_))

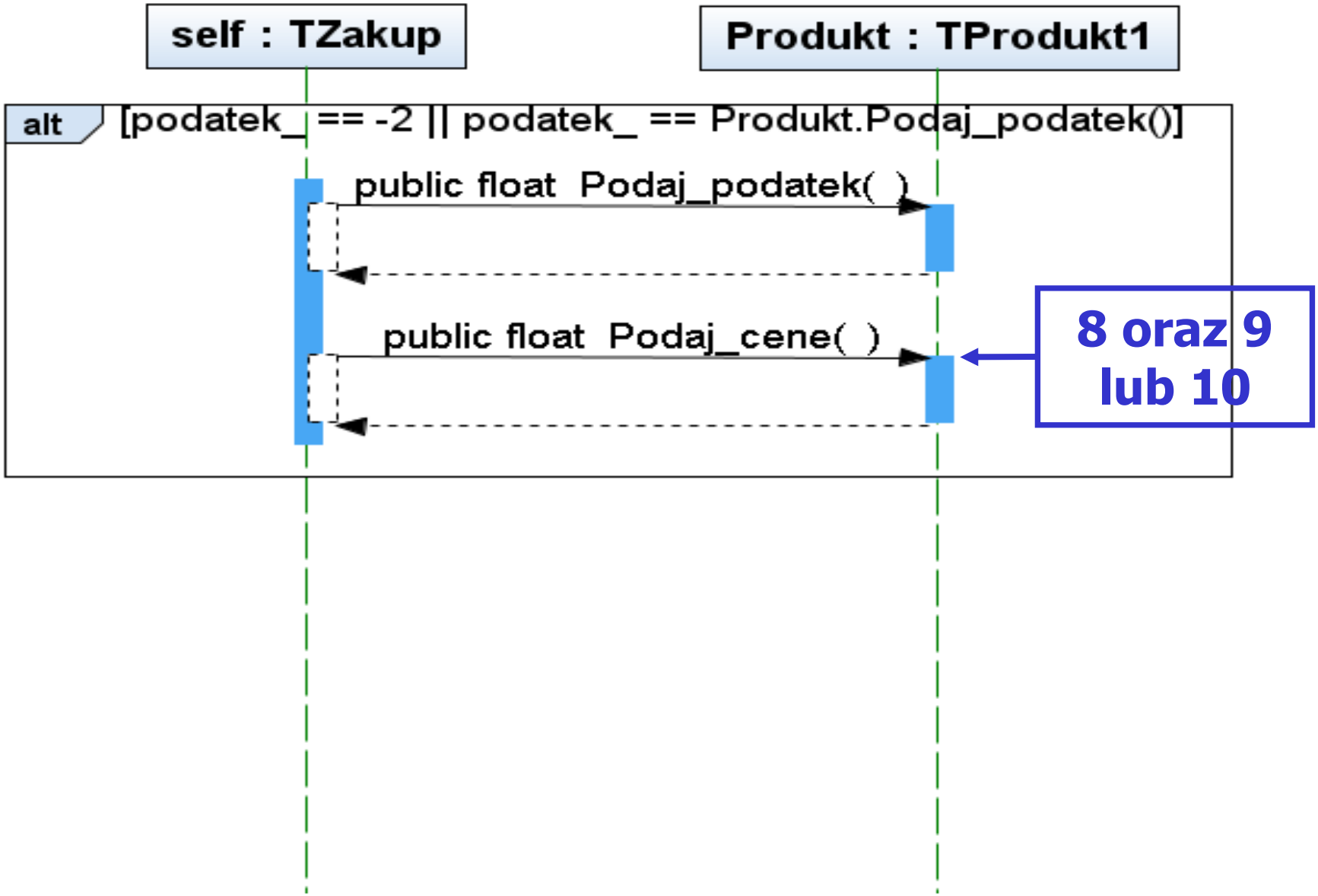


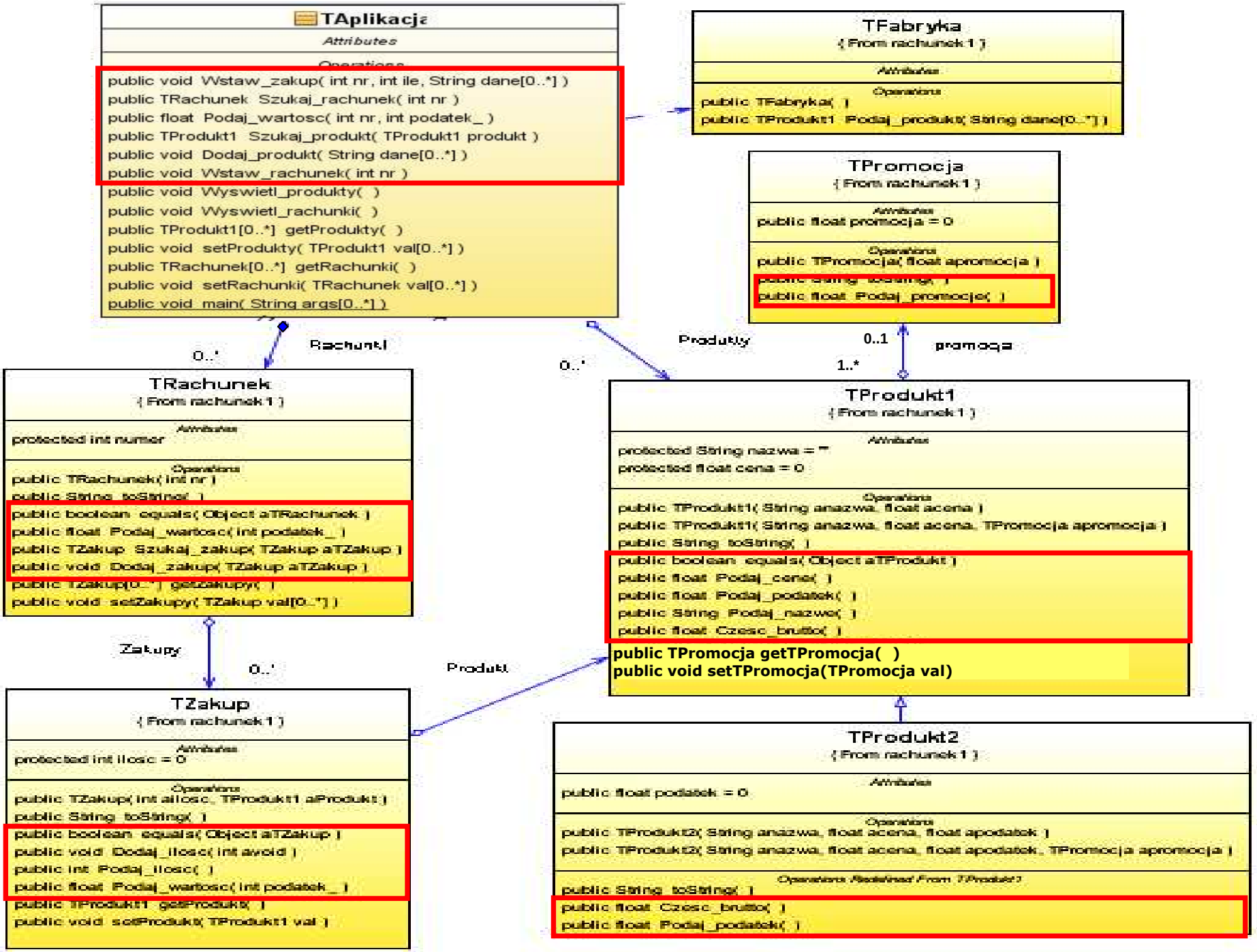

TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public String toString( )
public float Podaj_promocje( )
Rachunki
0..*
0..*
Produkty
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public boolean equals(Object aTRachunek)
public float Podaj_wartosc(int podatek_)
public TZakup Szukaj_zakup(TZakup aTZakup)
public void Dodaj_zakup(TZakup aTZakup)
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public boolean equals(Object aTZakup)
public void Dodaj_ilosc(int avoid)
public int Podaj_ilosc( )
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )

(9) float TRachunek::Podaj_wartosc(int podatek_)



(10) float TZakup::Podaj_wartosc(int podatek_)




TAplikacja
Attributes
Operations
public void Wstaw_zakup(int nr, int ile, String dane[0..*])
public TRachunek Szukaj_rachunek(int nr)
public float Podaj_wartosc(int nr, int podatek_)
public TProdukt1 Szukaj_produkt(TProdukt1 produkt)
public void Dodaj_produkt(String dane[0..*])
public void Wstaw_rachunek(int nr)
public void Wyswietl_produkty( )
public void Wyswietl_rachunki( )
public TProdukt1[0..*] getProdukty( )
public void setProdukty(TProdukt1 val[0..*])
public TRachunek[0..*] getRachunki( )
public void setRachunki(TRachunek val[0..*])
public void main(String args[0..*])
TFabryka
{ From rachunek1 }
Attributes
Operations
public TFabryka( )
public TProdukt1 Podaj_produkt(String dane[0..*])
TPromocja
{ From rachunek1 }
Attributes
public float promocja = 0
Operations
public TPromocja(float apromocja)
public float Podaj_promocje( )
Rachunki
0..*
Produkty
0..*
0..1
promocja
1..*
TRachunek
{ From rachunek1 }
Attributes
protected int numer
Operations
public TRachunek(int nr)
public String toString( )
public boolean equals(Object aTRachunek)
public float Podaj_wartosc(int podatek_)
public TZakup Szukaj_zakup(TZakup aTZakup)
public void Dodaj_zakup(TZakup aTZakup)
public TZakup[0..*] getZakupy( )
public void setZakupy(TZakup val[0..*])
TProdukt1
{ From rachunek1 }
Attributes
protected String nazwa = ""
protected float cena = 0
Operations
public TProdukt1(String anazwa, float acena)
public TProdukt1(String anazwa, float acena, TPromocja apromocja)
public String toString( )
public boolean equals(Object aTProdukt)
public float Podaj_cene( )
public float Podaj_podatek( )
public String Podaj_nazwe( )
public float Czesc_brutto( )
public TPromocja getTPromocja( )
public void setTPromocja(TPromocja val)
Zakupy
0..*
Produkt
TZakup
{ From rachunek1 }
Attributes
protected int ilosc = 0
Operations
public TZakup(int ailosc, TProdukt1 aProdukt)
public String toString( )
public boolean equals(Object aTZakup)
public void Dodaj_ilosc(int avoid)
public int Podaj_ilosc( )
public float Podaj_wartosc(int podatek_)
public TProdukt1 getProdukt( )
public void setProdukt(TProdukt1 val)
TProdukt2
{ From rachunek1 }
Attributes
public float podatek = 0
Operations
public TProdukt2(String anazwa, float acena, float apodatek)
public TProdukt2(String anazwa, float acena, float apodatek, TPromocja apromocja)
Operations Redefined From TProdukt1
public String toString( )
public float Czesc_brutto( )
public float Podaj_podatek( )