Inżynieria Oprogramowania

Modelowanie Systemu Informatycznego

prezentacja 10

Modelowanie zachowania – diagram komunikacji, diagram sekwencji, diagram przeglądu interakcji i diagram czasowy

wersja 1.0

dr inż. Paweł Głuchowski

Wydział Informatyki i Telekomunikacji, Politechnika Wrocławska

Treść prezentacji

1

Diagram komunikacji

Diagram Komunikacji /communiaction diagram/
(dawniej zwany diagramem współpracy /collaboration/)

- Modeluje **interakcję** – przepływ sterowania między jej uczestnikami, np.:
  - złożony przypadek użycia lub operację jego realizacji,
  - algorytm wykonania operacji klasy.
- **Uczestnik interakcji** – aktor /actor/ lub **linia życia** /lifeline/: klasa, instancja klasy (obiekt), komponent, system itd.
- Nazwa linii życia:
  - dowolna, ogólna – dla diagramu konceptualnego;
  - ***instancja:klasa*** – uczestnikiem jest konkretna instancja klasy;
  - ***:klasa*** – uczestnikiem jest anonimowa instancja klasy lub sama klasa;
  - ***self*** – uczestnikiem jest posiadacz tego diagramu.
- Interakcję można umieścić w torach określających role jej uczestników (odpowiednik partycji z diagramu czynności).

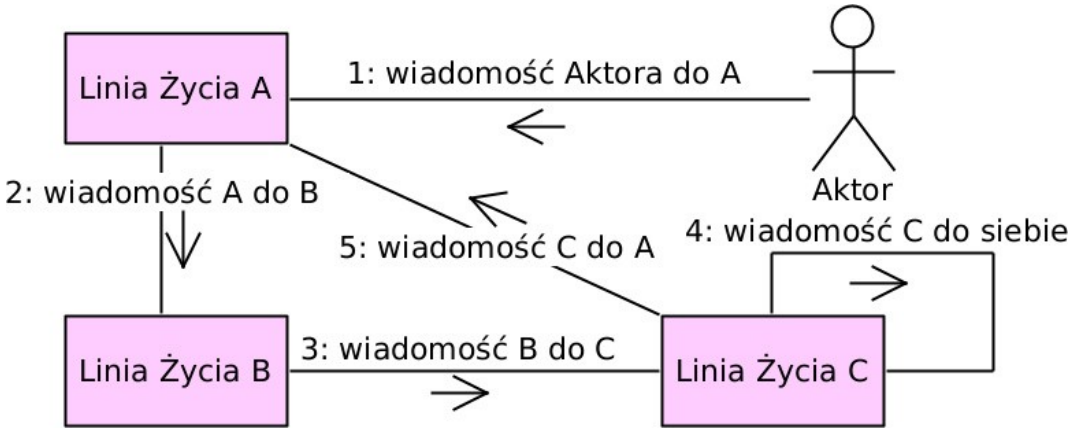
Diagram komunikacji

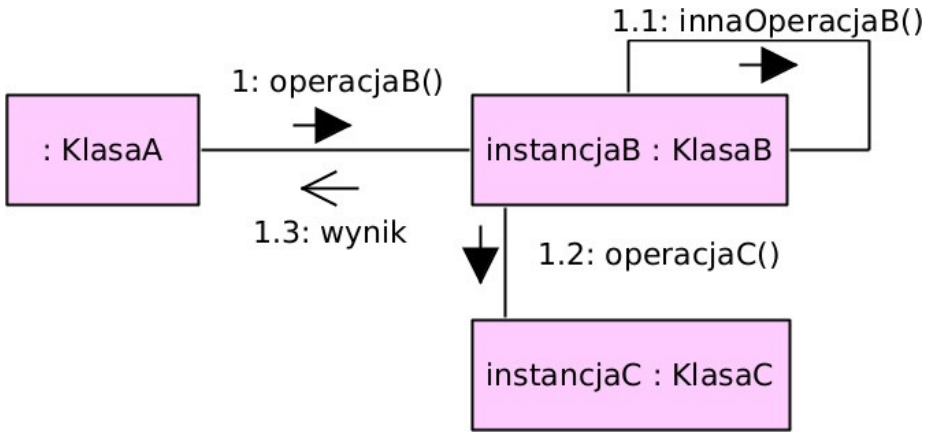
Wiadomość /message/

- **Relacja wiadomości**:
  - przekazanie sterowania (i opcjonalnie obiektu) między uczestnikami interakcji: od nadawcy do odbiorcy;
  - wykonanie operacji przez odbiorcę w odpowiedzi na żądanie nadawcy.
- **Odbiorcą** może być nadawca lub inny uczestnik interakcji.
- **Numer wiadomości** – kolejność jej wykonania;
  - pierwsza jest wiadomość nr 1,
  - numery mogą mieć podnumery itd. (np. 1.1).
- **Opis wiadomości** – zwykle operacja do wykonania przez odbiorcę.
- **Symbol strzałki** – wskazuje odbiorcę i rodzaj wiadomości.

Wiadomość

- **Wiadomość asynchroniczna** /asynchronous/ – asynchroniczne wywołanie operacji odbiorcy:
  - NIE towarzyszy jej wiadomość zwrotna,
  - grot relacji jest otwarty.
- **Wiadomość synchroniczna** /synchronous/ – synchroniczne wywołanie operacji odbiorcy:
  - może jej towarzyszyć asynchroniczna wiadomość zwrotna (otwarty grot),
  - grot relacji jest zamknięty.





2

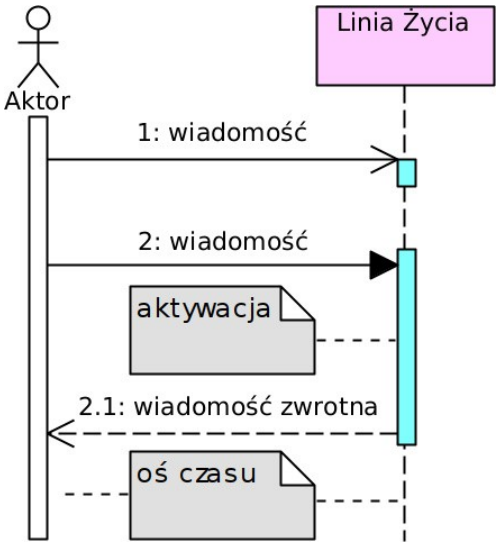
Diagram sekwencji

Diagram Sekwencji /sequence diagram/

- Modeluje **interakcję** – przepływ sterowania między jej uczestnikami, np.:
  - złożony przypadek użycia lub operację jego realizacji,
  - algorytm wykonania operacji klasy.
- **Uczestnik interakcji** – aktor /actor/ lub **linia życia** /lifeline/: klasa, instancja klasy (obiekt), komponent, system itd.
- Nazwa linii życia:
  - ogólna – dla diagramu konceptualnego;
  - ***instancja:klasa*** – uczestnikiem jest konkretna instancja klasy;
  - ***:klasa*** – uczestnikiem jest anonimowa instancja klasy lub sama klasa;
  - ***self*** – uczestnikiem jest posiadacz tego diagramu.

Linia życia /lifeline/

- Uczestnik interakcji + oś czasu jego funkcjonowania (im niżej, tym później).
- Do linii życia wchodzą relacje **wiadomości**, powodując:
  - utworzenie tej linii życia,
  - zakończenie tej linii życia,
  - przyjęcie informacji przez tę linię życia,
  - rozpoczęcie wykonywania operacji przez tę linię życia,
  - otrzymanie wyniku operacji wykonanej przez tę lub inną linię życia.



- Czas wykonywania operacji przez linię życia pokazuje **aktywacja** /activation, execution specification/ – wąski prostokąt na jej osi czasu (i opcjonalnie wymiarowanie z wartościami czasu).
- Gdy uczestnikiem interakcji jest **aktor** (byt spoza modelowanego systemu):
  - ikona i nazwa aktora są nad osią czasu,
  - wysyłanie i przyjmowanie wiadomości jest ograniczone.

Wiadomość /message/

- Przekazanie sterowania (i opcjonalnie obiektu) między uczestnikami interakcji: od nadawcy do odbiorcy;
- Wykonanie operacji przez odbiorcę w odpowiedzi na żądanie nadawcy.
- Wiadomości NIE dotyczące tej samej linii życia mogą być wykonywane w dowolnej kolejności.
- **Główne rodzaje wiadomości:**
  - asynchroniczna, – własna, – znaleziona, –trwająca.
  - synchroniczna, – rekursywna, – tworząca obiekt,
  - zwrotna, – zgubiona, – usuwająca obiekt,
- **Numer** wiadomości – kolejność jej wykonania:
  - pierwsza jest wiadomość nr 1,
  - numery mogą mieć podnumery itd. (np. 1.1).
- **Opis wiadomości** – zwykle operacja do wykonania przez odbiorcę.
  - Zwrot relacji wiadomości wskazuje odbiorcę wiadomości.

Opis wiadomości żądania wykonania operacji

- **Składnia:**

```
<request-message-label> ::= <message-name> [‘(‘[<input-argument-list>] ’)’]

<input-argument-list> ::= <input-argument> [‘,’<input-argument>]*

<input-argument> ::= [<in-parameter-name> ‘=’] <value-specification> | ‘-’
```

  - **<message-name>** – nazwa wiadomości (np. operacji),
  - **<in-parameter-name>** – nazwa wejściowego parametru wiadomości,
  - **<value-specification>** – parametr wiadomości.
- **Przykłady:**
  - `wiadomość`
  - `operacja()`
  - `operacja(parametr)`
  - `operacja(parametr1, parametr2)`
  - `operacja(nazwa1=parametr1, nazwa2=parametr2)`
- Często dodaje się na koniec typ wiadomości zwrotnej `:<return-type>`, np.: `operacja():int`

Opis wiadomości zwrotnej

- **Składnia:**

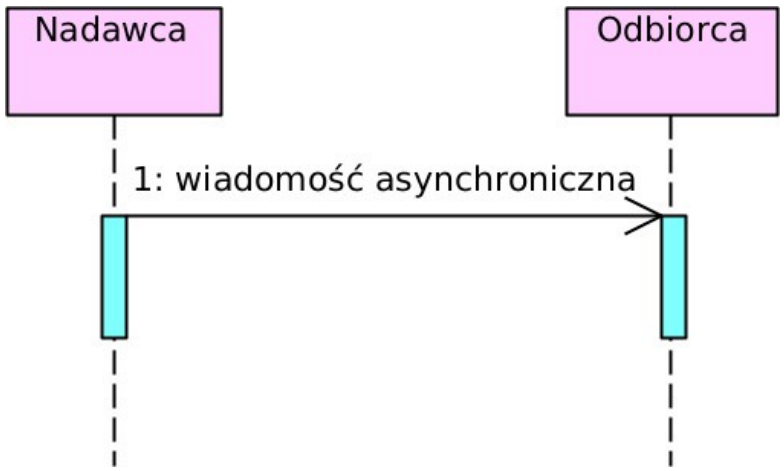
```
<reply-message-label> ::= [<assignment-target> ‘=’] <message-name>
        [‘(’ [<output-argument-list>] ‘)’] [‘:’ <value-specification>]

<output-argument-list> ::= <output-argument> [‘,’<output-argument>]*

<output-argument> ::= <out-parameter-name> ‘:’ <value-specification> |
<assignment-target> ‘=’ <out-parameter-name> [‘:’ <value-specification>]
```

  - **<assignment-target>** (dla wiadomości) – wyjściowy parametr wiadomości żądania, z którą związana jest wiadomość zwrotna (dla wiadomości zwrotnej lub jej parametru);
  - **<value-specification>** – typ wiadomości zwrotnej lub jej parametr;
  - **<message-name>** – nazwa lub treść wiadomości (np. nazwa operacji);
  - **<out-parameter-name>** – nazwa wyjściowego parametru wiadomości.

- **Przykłady:**
  - `wynik`
  - `wynik:string`
  - `out = wynik(-):45`
    gdzie *wynik()* to operacja, która kończy się tą wiadomością zwrotną
  - `out = wynik(nazwa1=parametr1:5, nazwa2=parametr2:10)`

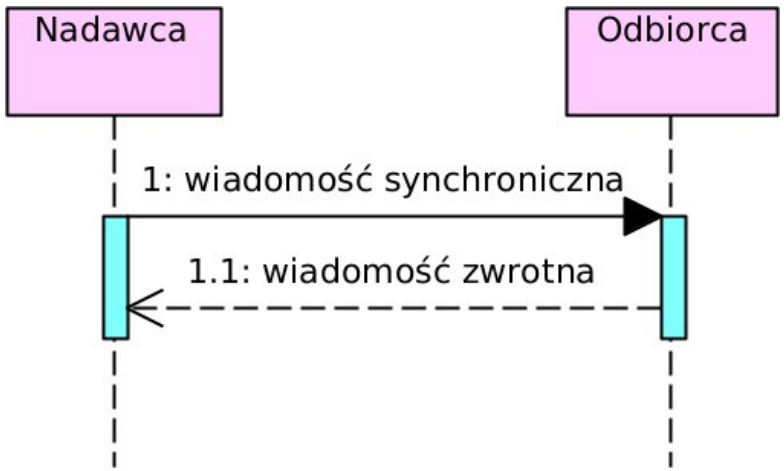
Wiadomość asynchroniczna /asynchronous/

- Asynchroniczne wywołanie operacji odbiorcy przez nadawcę:
  - nadawca NIE oczekuje na zakończenie tej operacji,
  - nadawca NIE otrzymuje wiadomości zwrotnej od odbiorcy.
- **Odbiorcą** może być nadawca lub inny uczestnik interakcji.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.
- **Linia relacji wiadomości** – ciągła z otwartym grotem wskazującym wykonawcę operacji.



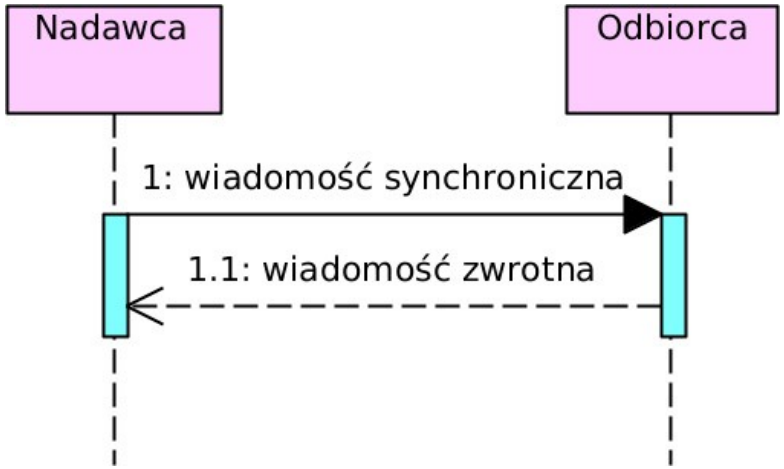
Wiadomość synchroniczna /synchronous/

- Synchroniczne wywołanie operacji odbiorcy przez nadawcę:
  - nadawca oczekuje na zakończenie tej operacji,
  - nadawca może otrzymać wiadomość zwrotną od odbiorcy.
- **Odbiorcą** może być nadawca lub inny uczestnik interakcji.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.
- **Linia relacji wiadomości** – ciągła z zamkniętym grotem wskazującym wykonawcę operacji.



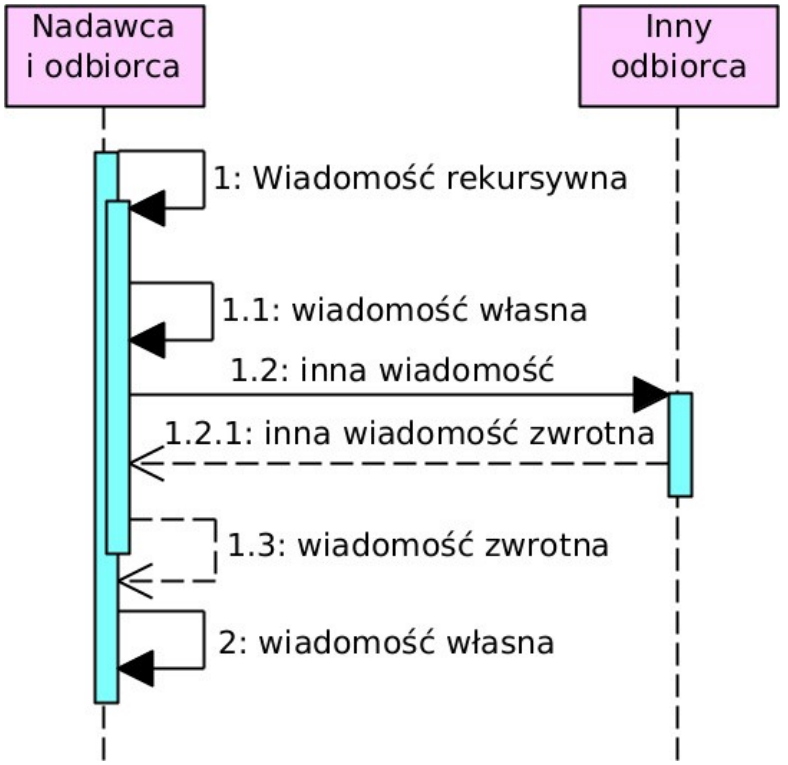
Wiadomość zwrotna /reply/

- Kończy wykonywanie operacji przez nadawcę:
  - nadawca zwraca sterowanie odbiorcy,
  - może też przekazać odbiorcy wynik kończonej operacji.
- **Odbiorcą** jest nadawca synchronicznej wiadomości, która wywołała kończoną operację (wiadomość zwrotna wchodzi do aktywacji odbiorcy).
- Aktywacja nadawcy kończy się po wysłaniu wiadomości zwrotnej.
- **Opis relacji wiadomości** – nic lub dane zwracane przez operację.
- **Linia relacji wiadomości** – przerywana z otwartym grotem wskazującym odbiorcę.



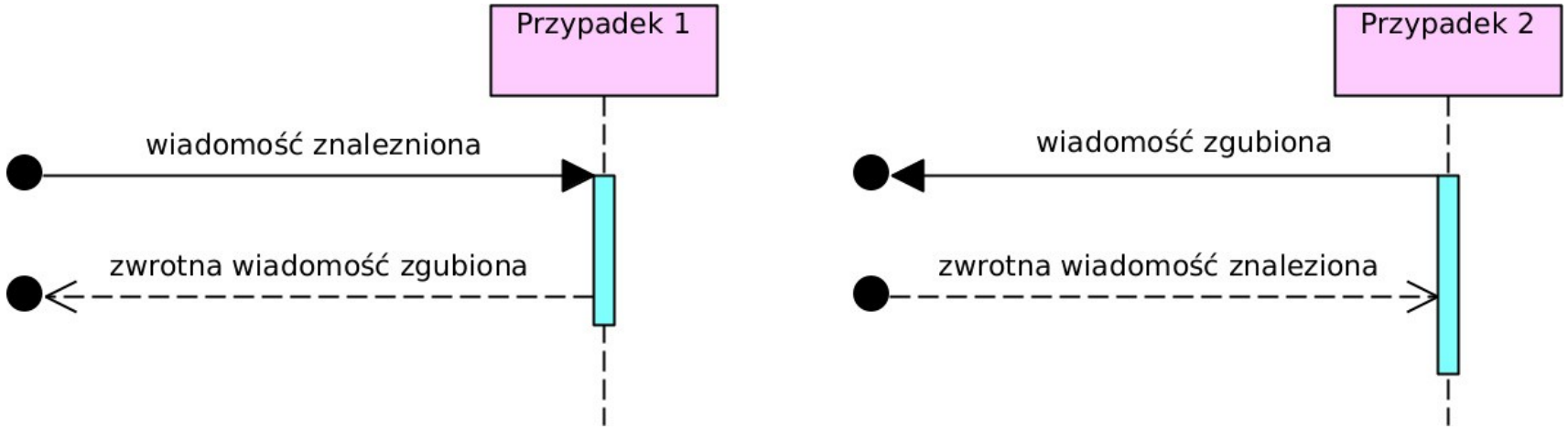
Wiadomość własna i rekursywna

- Wywołanie operacji odbiorcy przez nadawcę – nadawca jest odbiorcą.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.
- **Linia relacji wiadomości** – zgięta w kształt ⊐, ciągła, wskazująca wykonawcę operacji.
- **Wiadomość własna** /self/
  - Wykonywana operacja NIE powoduje wysłania innych wiadomości
    - nawet jej wiadomości zwrotnej.
  - NIE umieszcza na osi czasu odbiorcy aktywacji wykonania tej operacji.
- **Wiadomość rekursywna** /recursive/
  - Wykonywana operacja powoduje wysłanie innych wiadomości.
  - Umieszcza na osi czasu odbiorcy aktywację wykonania tej operacji.



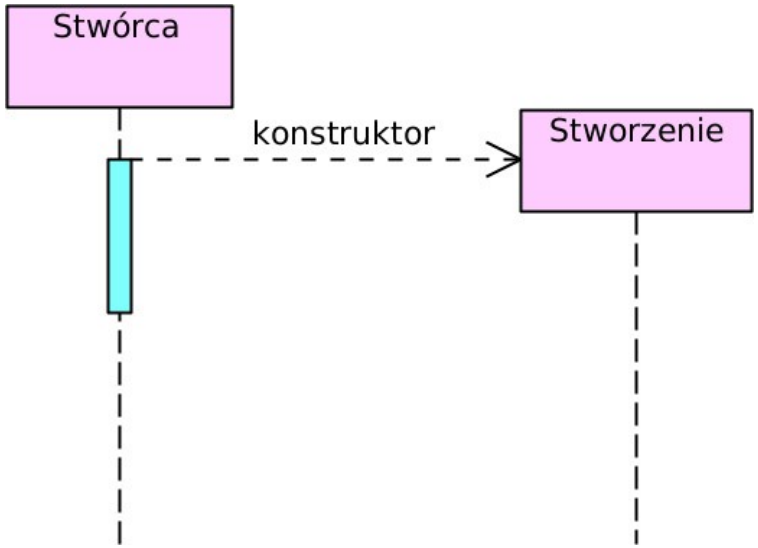
Wiadomość zgubiona i znaleziona

- **Wiadomość zgubiona** /lost/
  - Odbiorca wiadomości nie jest znany lub nie istnieje.
  - **Linia relacji wiadomości** – wychodzi z osi czasu nadawcy i wskazuje na węzeł *czarne koło*.
- **Wiadomość znaleziona** /found/
  - Nadawca wiadomości nie jest znany.
  - **Linia relacji wiadomości** – wychodzi z węzła *czarne koło* i wskazuje na oś czasu odbiorcy.
- **Opis relacji wiadomości** – operacja do wykonania przez odbiorcę.



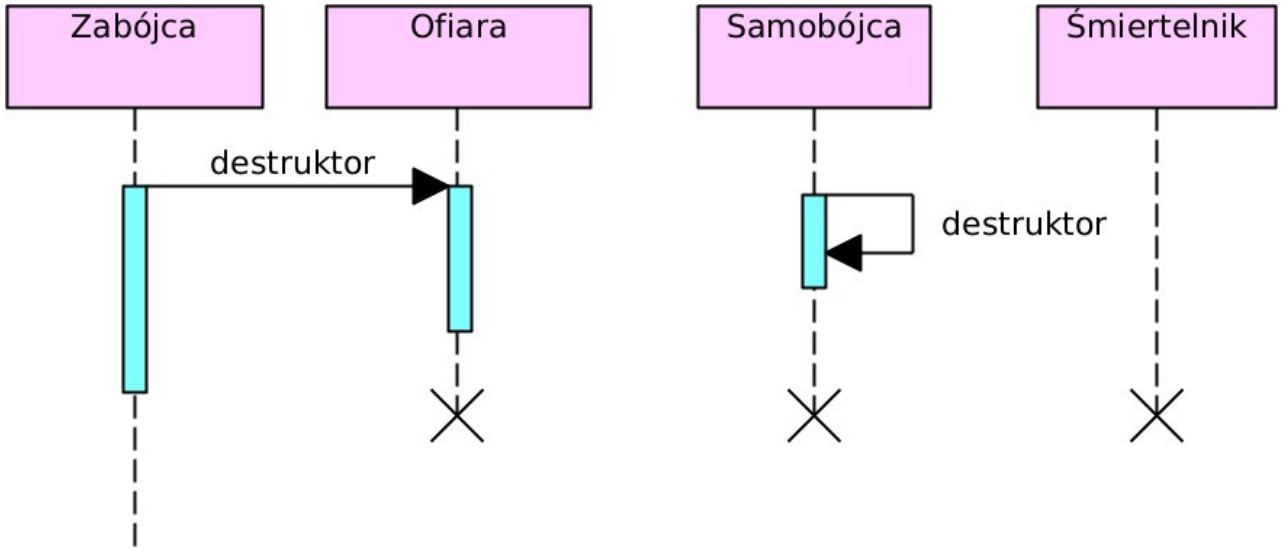
Wiadomość tworząca obiekt /object creation/

- Utworzenie nowej linii życia (obiektu, np. instancji klasy) przez inną.
- **Linia relacji wiadomości** – przerywana z otwartym grotem wskazującym nazwę tworzonej linii życia.
- **Nadawcą** NIE jest odbiorca.
- Nadawca NIE otrzymuje wiadomości zwrotnej od odbiorcy.
- **Opis relacji wiadomości** – operacja utworzenia obiektu (konstruktor).



Wiadomość usuwająca obiekt /object deletion/

- Usunięcie linii życia obiektu (np. instancji klasy) przez nią samą lub inną.
- **Linia relacji wiadomości** – ciągła z grotem wskazującym oś czasu usuwanej linii życia.
- Następnie usuwana linia życia kończy się znakiem X.
- **Nadawcą** może być odbiorca.
- Nadawca może otrzymać wiadomość zwrotną od odbiorcy (jeśli nie jest odbiorcą).
- **Opis relacji wiadomości** – operacja usunięcia obiektu (destruktor).



Parametry czasowe

- Określają ilościowo w nawiasach { } i przy pomocy „wymiarowania”:
  - moment wysłania lub otrzymania wiadomości,
  - długość czasu wysyłania wiadomości,
  - odstęp czasu między zdarzeniami wysłania lub otrzymania wiadomości.
- Konkretne liczby lub symbolizujące je zmienne, które mogą być zależne od:
  - ***now*** – zaobserwowany moment czasu,
  - ***duration*** – zaobserwowana długość czasu.

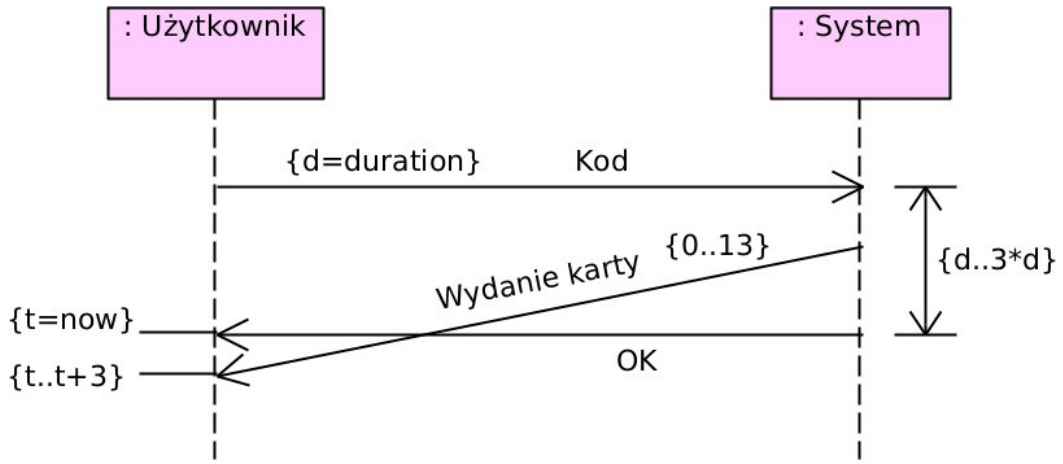
Wiadomość trwająca /duration message/

- Otrzymanie wiadomości następuje zauważalny czas po jej nadaniu.
- **Relacja wiadomości:**
  - jest skierowana pochyło w dół (nie jest pozioma),
  - może przecinać inne relacje wiadomości.

Przykład parametrów czasowych i wiadomości trwającej

na podst. Unified Modeling Language (UML)

- **Użytkownik** wysyła do **Systemu** wiadomość **Kod**:
  - czas wysyłania **Kod** jest mierzony i zapisywany jako **d**.
- **System** odpowiada wysłaniem do **Użytkownika** wiadomości trwającej **Wydanie karty**, a następnie wiadomości **OK**:
  - czas wysyłania **Wydanie karty** trwa od 0 do 13 jednostek;
  - odstęp czasu między wysłaniem (i odebraniem) **Kod**, a wysłaniem (i odebraniem) **OK** wynosi od **d** do **3*d**;
  - moment wysłania (i otrzymania) **OK** jest zapisywany jako **t**.
  - moment otrzymania **Wydanie karty** wynosi od **t** do **t+3**.



Połączony fragment /combined fragment/

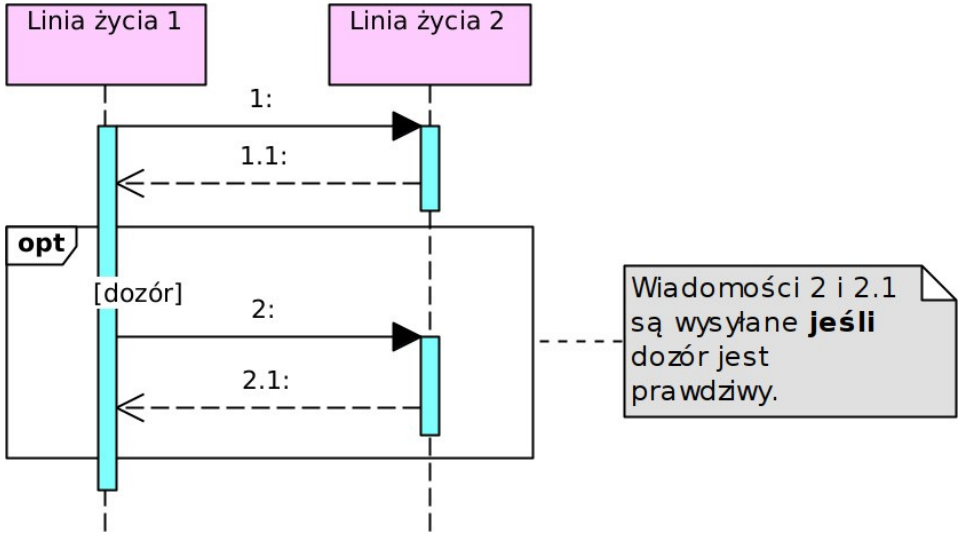
- Otacza część diagramu, aby nadać jej określone znaczenie:
  - obejmuje określony czas (przez rozciągnięcie w pionie),
  - obejmuje określone linie życia (przez rozciągnięcie w poziomie),
  - obejmuje relacje wiadomości przechodzące między tymi liniami życia.
- **Operator** /operator/ – określa interpretację zawartości połączonego fragmentu.
- **Operand** /operand/ – część połączonego fragmentu (ich liczba zależy od operatora).
  - Operandy rozdzielane są poziomą przerywaną linią.
  - Może mieć **dozór** /guard/ w nawiasach [ ]:
    - logiczny warunek uwzględnienia zawartych relacji wiadomości,
    - brak dozoru oznacza dozór [*true*].
  - Dotyczy tylko tych relacji wiadomości, które są wysyłane lub otrzymywane przez linię życia objętą tym operandem.

Rodzaje operatorów

- ***opt*** – opcja (1 operand w połączonym fragmencie),
- ***alt*** – alternatywy (2 lub więcej),
- ***par*** – współbieżność (2 lub więcej),
- ***seq*** – słaba kolejność (2 lub więcej),
- ***strict*** – ścisła kolejność (2 lub więcej),
- ***neg*** – niepoprawność (1),
- ***critical*** – region krytyczny (1),
- ***ignore*** – pominięcie (1),
- ***consider*** – uwzględnienie (1),
- ***assert*** – założenie (1),
- ***loop*** – pętla (1),
- ***break*** – opuszczenie (1),
- ***ref*** – użycie interakcji (1).

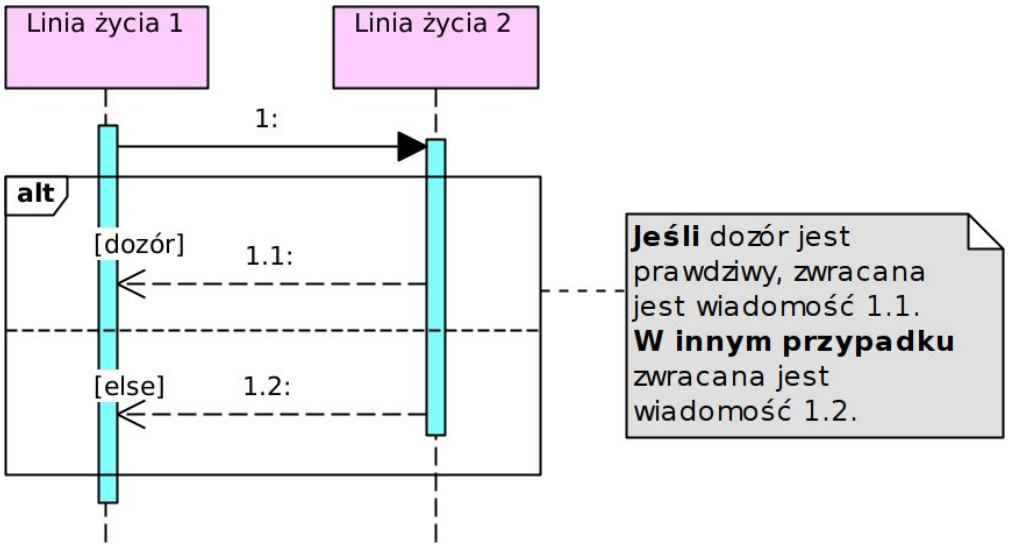
opt (opcja /option/)

- ***opt*** (1 operand):
  - jeśli dozór operandu jest prawdziwy,
    wykonywane są interakcje zawarte w operandzie;
  - w innym przypadku są pomijane.
- Odpowiada konstrukcji *if* (BEZ *else*).



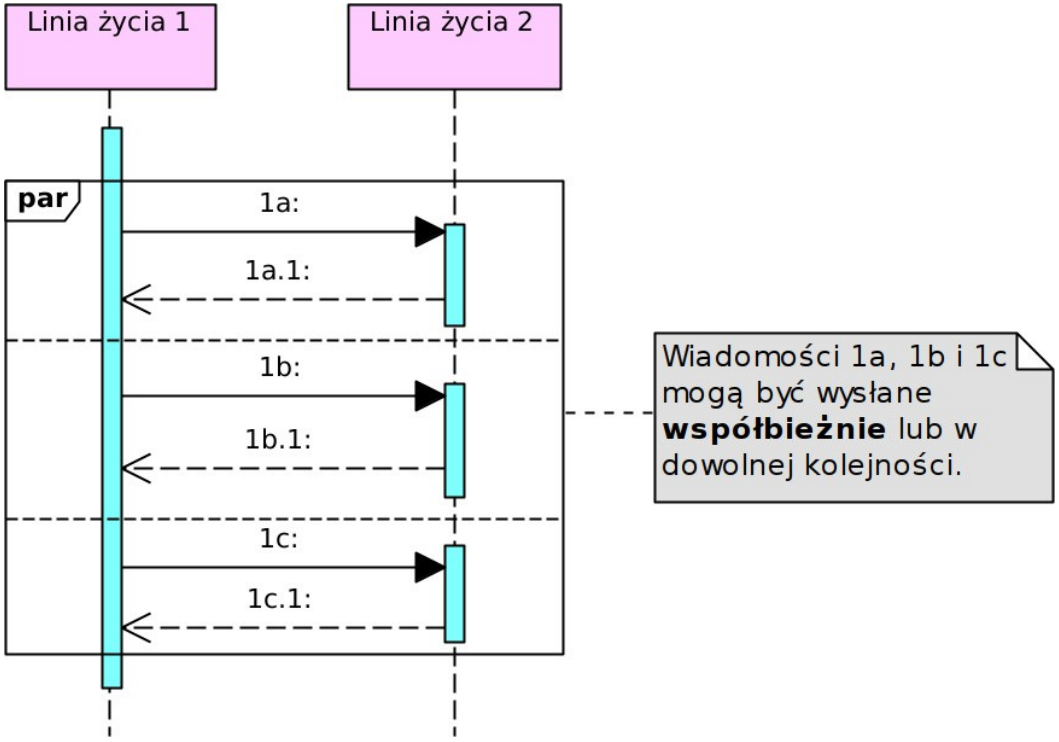
alt (alternatywy /alternatives/)

- ***alt*** (2 lub więcej operandów):
  - sprawdzana jest prawdziwość dozorów kolejnych operandów (od góry);
  - jeśli dozór danego operandu jest prawdziwy:
    - wykonywane są interakcje zawarte w tym operandzie,
    - interakcje zawarte w pozostałych operandach są pomijane;
  - w innym przypadku jego interakcje są pomijane;
  - dozór [*else*] w ostatnim operandzie oznacza wszystkie inne przypadki.
- Odpowiada konstrukcjom *if*… *else* i *switch* (z *break*).



par (współbieżność /parallel/)

- ***par*** (2 lub więcej operandów):
  - interakcje zawarte w tym samym operandzie wykonywane są normalnie,
  - wszystkie operandy są współbieżne:
    - interakcje zawarte w różnych operandach mogą być wykonywane współbieżnie lub w dowolnej kolejności.



seq (słaba kolejność /weak sequencing/)

- ***seq*** (2 lub więcej operandów):
  - interakcje zawarte w tym samym operandzie wykonywane są normalnie,
  - wszystkie operandy są częściowo współbieżne:
    - interakcje zawarte w różnych operandach i dotyczące różnych linii życia mogą być wykonywane współbieżnie lub w dowolnej kolejności;
    - interakcje zawarte w różnych operandach i dotyczące tej samej linii życia wykonywane są normalnie i w kolejności ich operandów.

strict (ścisła kolejność /strict sequencing/)

- ***strict*** (2 lub więcej operandów):
  - interakcje zawarte w tym samym operandzie wykonywane są normalnie,
  - interakcje zawarte w różnych operandach wykonywane są w kolejności ich operandów, nawet jeśli dotyczą różnych linii życia.

neg (niepoprawność /negative/)

- ***neg*** (1 operand):
  - interakcje zawarte w operandzie są niepoprawne (nie powinny być wykonane).

critical (region krytyczny /critical region/)

- ***critical*** (1 operand):
  - podczas wykonywania interakcji zawartych w operandzie, nie mogą być wykonywane interakcje spoza niego, które dotyczą tych samych linii życia
  - także wtedy, gdy połączony fragment *critical* znajduje się w połączonym fragmencie *par*.

ignore (pominięcie)

- ***ignore{wiadomości}*** (1 operand):
  - podaje wiadomości niepokazywane przez operand:
    - mogą być przekazywane, ale są nieistotne;
    - ich przekazywanie jest ignorowane.
  - `wiadomości ::= nazwa-wiadomości[‘,’ nazwa-wiadomości]*`
- Operator *ignore* po nazwie diagramu dotyczy całego diagramu.
- Przydatne w specyfikacji testu systemu (w tym oprogramowania).

consider (uwzględnienie)

- ***consider{wiadomości}*** (1 operand):
  - podaje jedynie wiadomości pokazywane przez operand:
    - INNE wiadomości mogą być przekazywane, ale są nieistotne;
    - przekazywanie INNYCH wiadomości jest ignorowane.
  - `wiadomości ::= nazwa-wiadomości[‘,’ nazwa-wiadomości]*`
- Operator *consider* po nazwie diagramu dotyczy całego diagramu.
- Przydatne w specyfikacji testu systemu (w tym oprogramowania).

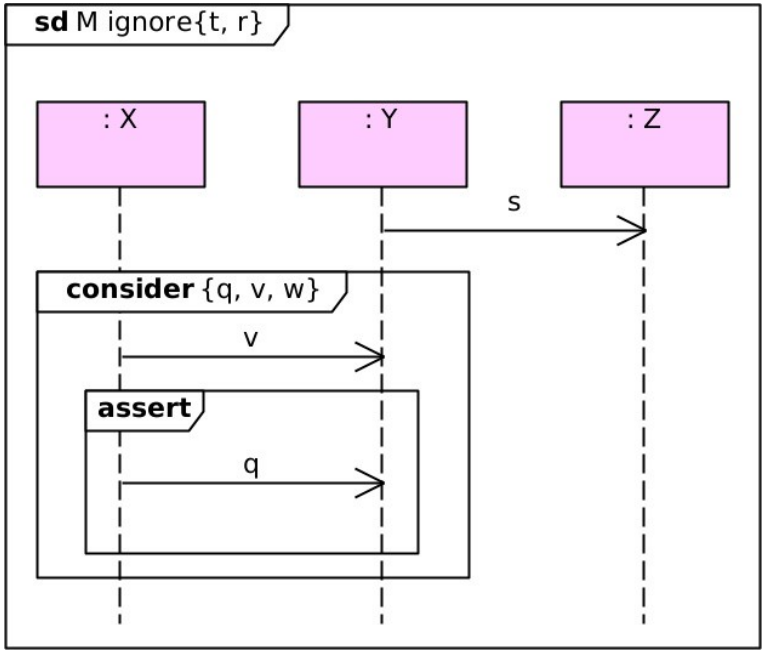
assert (założenie /assertion/)

- ***assert*** (1 operand):
  - zakłada (narzuca) wykonanie interakcji zawartych w operandzie:
    - w miejscu położenia tego połączonego fragmentu dozwolone są TYLKO interakcje zawarte w operandzie,
    - wszystkie inne WTEDY NIE są wykonywane.
- Uzupełniają połączone fragmenty *ignore* i *consider*.
- Przydatne w specyfikacji testu systemu (w tym oprogramowania).

Przykład połączonych fragmentów *ignore*, *consider* i *assert* w modelu testu systemu

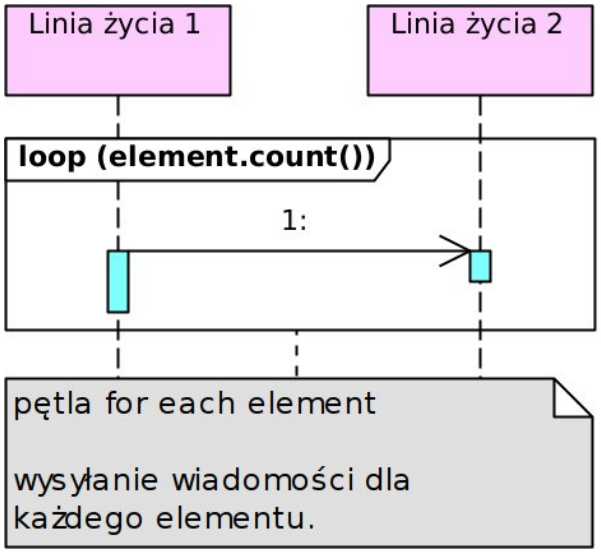
na podst. Unified Modeling Language (UML)

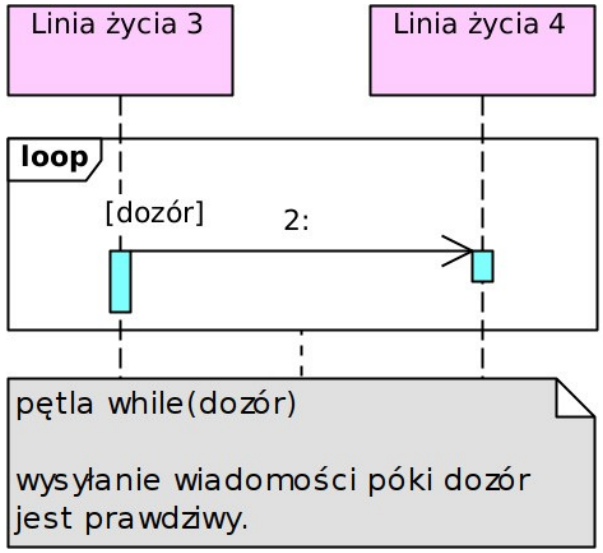
- Diagram **M** modeluje test systemu z liniami życia **X**, **Y** i **Z**, ignorując wykonywanie wiadomości **t** i **r**.
  - Dla testu nie jest ważne, jak system obsługuje ich wykonywanie.
- Po wykonaniu wiadomości **s** mogą być wykonywane dowolne wiadomości (poza **t** i **r**), ale:
  - tylko wykonywanie wiadomości **q**, **v** i **w** jest dalej uwzględniane w teście;
  - zaraz po wykonaniu wiadomości **v**:
    - zakłada się, że nastąpi wykonanie wiadomości **q**;
    - wykonanie innej wiadomości zamiast niej jest niepoprawne.

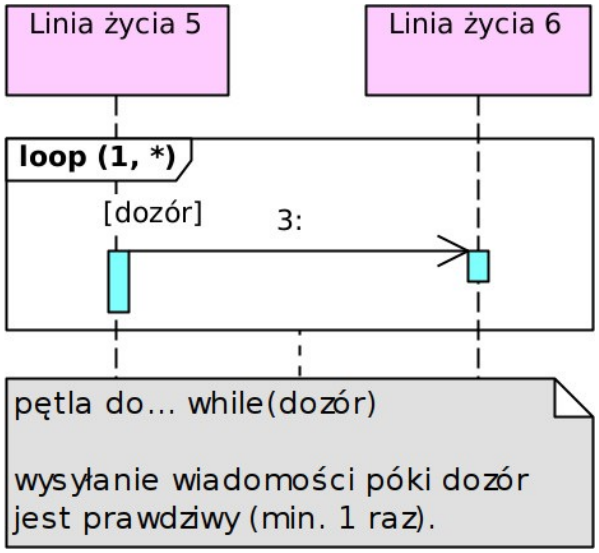


loop (pętla)

- ***loop(min,max)*** (1 operand):
  - interakcje zawarte w operandzie wykonywane są normalnie;
  - to wykonanie odbywa się przynajmniej *min* razy i najwyżej *max* razy:
    - *loop(ile)* oznacza *loop(ile,ile)* – dokładnie *ile* razy,
    - brak *(min,max)* oznacza *loop(0,*)* – 0 lub dowolnie wiele razy,
    - nie powtórzy się, gdy odbyło się choć *min* razy i dozór jest fałszywy.
  - w innym przypadku są pomijane.
- Odpowiada konstrukcjom *for, do… while(…)* i *while(…)*.

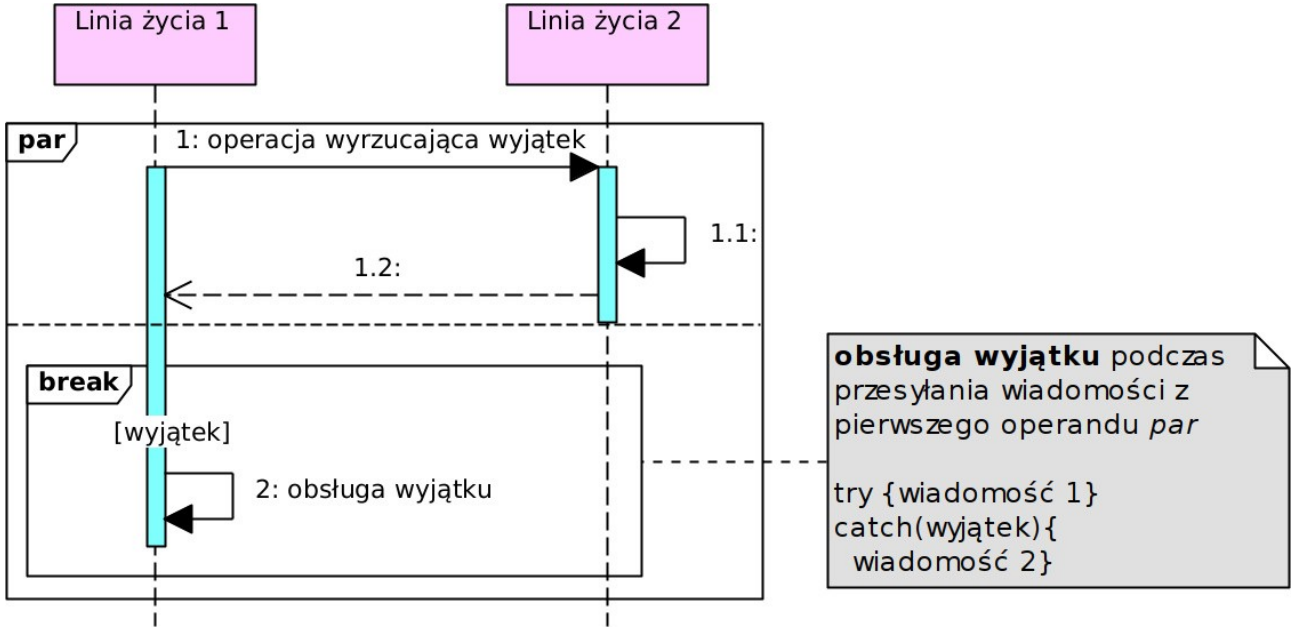






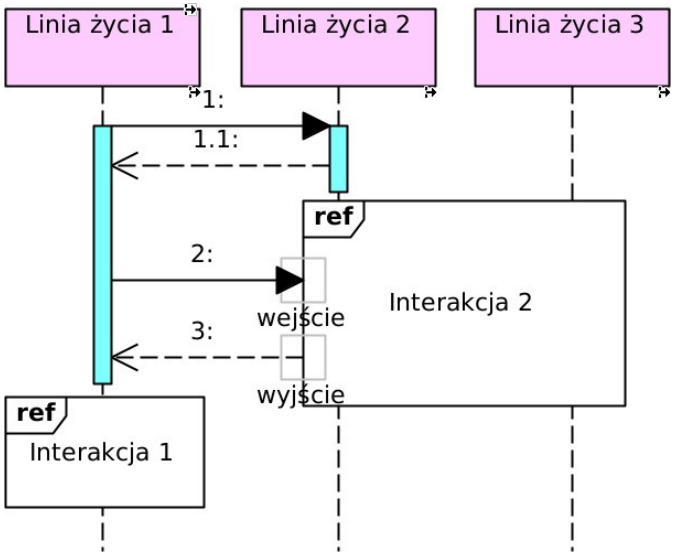
break (opuszczenie)

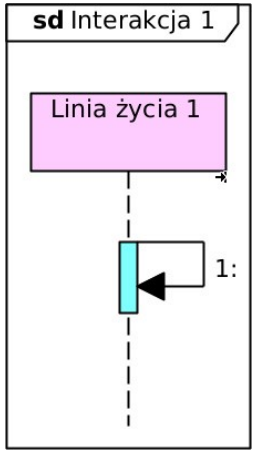
- ***break*** (1 operand):
  - całkowicie zawiera się w innym połączonym fragmencie;
  - kiedy dozór operandu jest prawdziwy:
    - inne interakcje zawierającego fragmentu są pomijane,
    - wykonywane są interakcje zawarte w operandzie;
  - w innym przypadku są pomijane;
  - brak dozoru oznacza jego losową prawdziwość.
- Odpowiada konstrukcjom *break* wyjścia z pętli i *catch* w obsłudze wyjątków.

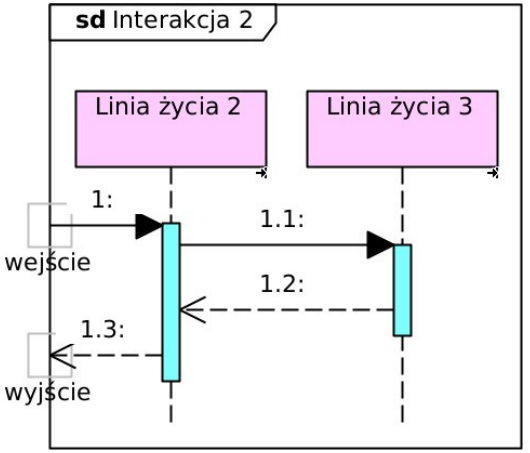


ref (użycie interakcji /interaction use/)

- ***ref*** (1 operand):
  - zastępuje część interakcji diagramu:
    - która znajduje się na innym diagramie;
    - obejmuje linie życia, które tam też są;
    - ma nazwę (w tym nazwę tego diagramu);
    - NIE zawiera żadnych interakcji.
- Pozwala wielokrotnie użyć tę samą interakcję i oczyścić diagram ze „zbędnych” interakcji.
- Do połączonego fragmentu *ref* mogą wchodzić i wychodzić wiadomości:
  - przez nazwane **bramki**,
  - diagram związany z tym fragmentem ma tak samo nazwane bramki,
  - lub bezpośrednio.







ref (użycie interakcji)

- **Składnia nazwy:**

```
<name> ::= [<attribute-name> '=' ] [<collaboration-use> '.']
<interaction-name> ['(' <io-argument> [',' <io-argument>]* ')']
[':' <return-value>]

<io-argument> ::= <in-argument> | 'out' <out-argument>
```

<interaction-name> – nazwa interakcji i związanego z nią diagramu;

<attribute-name> – atrybut linii życia (z diagramu zawierającego ten połączony fragment *ref*), do którego trafi wartość zwracana przez interakcję;

<collaboration-use> – określenie części większej interakcji;

<in-argument> – wejściowy parametr interakcji;

<out-argument> – wyjściowy parametr interakcji;

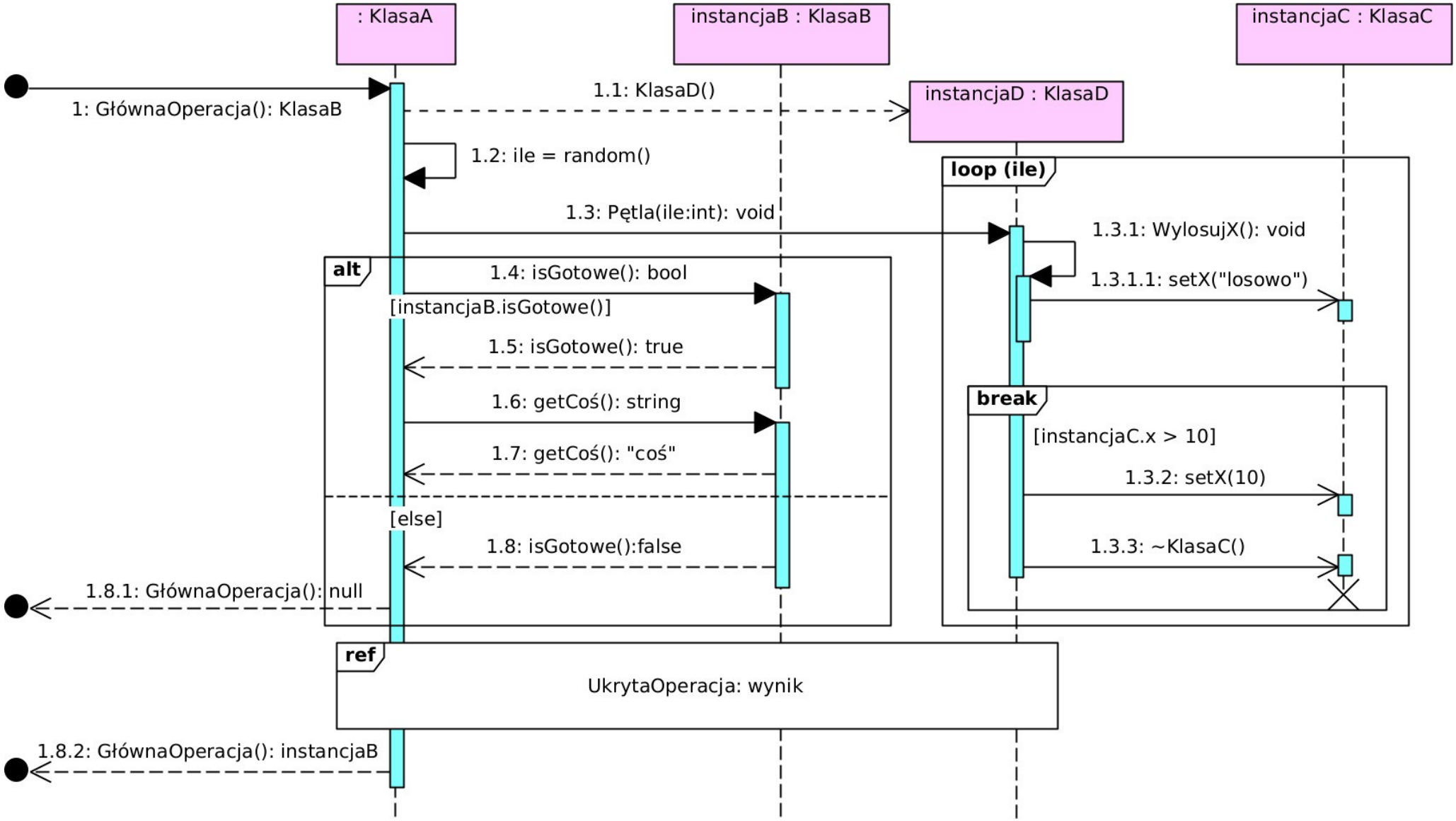
<return-value> – wartość zwracana przez interakcję.

- **Przykłady:**

```
Nazwa diagramu

Operacja(parametr1, parametr2)

liniaŻycia.atrybut = Operacja(parametr): wynikInterakcji
```

Przykład implementacyjnego modelu operacji klasy

- Wykonanie operacji **GłównaOperacja()** klasy **KlasaA**.



3

Diagram przeglądu interakcji

Diagram przeglądu interakcji
/interaction overview diagram/

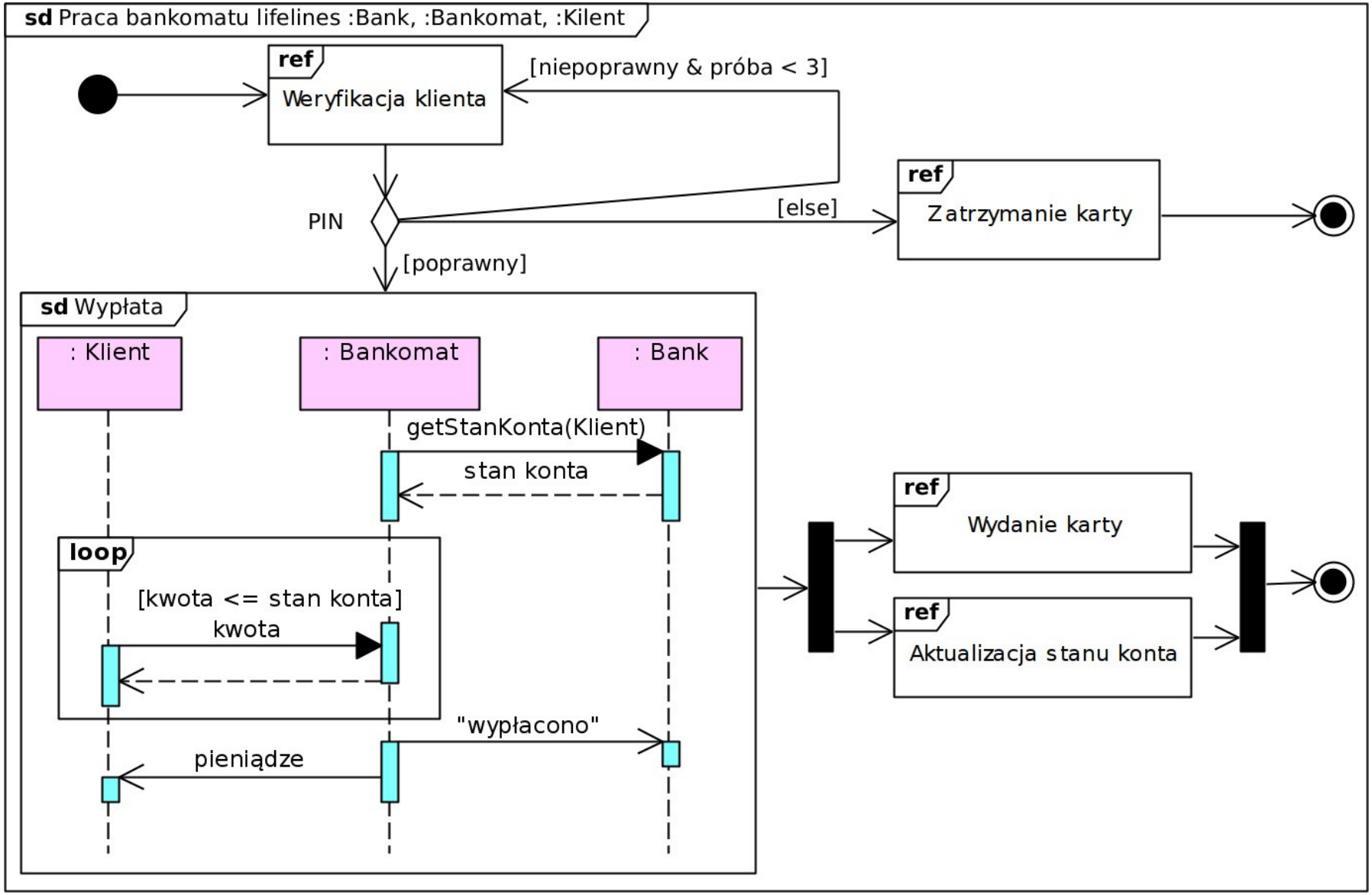
- Modeluje **interakcję** – przepływ sterowania między jej uczestnikami, np.:
  - złożony przypadek użycia lub operację jego realizacji,
  - algorytm wykonania operacji klasy.
- Diagram czynności:
  - wykorzystuje m.in. węzły decyzji i współbieżności,
  - zawiera zamiast akcji i czynności:
    - użycie interakcji (ramkę *ref* – referencja do innego diagramu);
    - diagram interakcji (ramkę z diagramem interakcji, np. sekwencji).
  - wymienia uczestników interakcji (po nazwie diagramu):

```
sd <diagram-name> lifelines <lifelines>
<lifelines> ::= <lifeline> [‘,’ <lifeline>]*
```

- **Uczestnik interakcji** – aktor /actor/ lub **linia życia** /lifeline/: klasa, instancja klasy (obiekt), komponent, system itd.
  - Może być pokazany tylko w ramce zawartego diagramu.

Przykład diagramu przeglądu interakcji

- Interakcje między liniami życia **:Bank**, **:Bankomat** i **:Klient**.



4

Modelowanie sekwencji

Elementy modelowania diagramu sekwencji

1. **Zdefiniuj uczestników interakcji – linie życia:**
   - np. dla aktorów, klas, obiektów.
2. **Ułóż linie życia w odpowiedniej kolejności:**
   - według najczęściej występującej sekwencji wiadomości między nimi.
3. **Połącz odpowiednie linie życia odpowiednimi wiadomościami:**
   - wiadomość modeluje czynność albo operację klasy lub obiektu;
   - wiadomości synchronicznej powinna towarzyszyć wiadomość zwrotna (między tymi samymi liniami życia, przeciwnie zwrócona), jeśli:
     - niesie informację zwrotną,
     - nie niesie informacji zwrotnej, ale poprawia czytelność interakcji.

Elementy modelowania diagramu sekwencji

4. **Ponumeruj wiadomości według kolejności ich wysyłania:**
   - stosuj wielopoziomową numerację.
5. **Dla wiadomości synchronicznej lub asynchronicznej umieść jej blok aktywacji na linii życia nadawcy wiadomości:**
   - od momentu wysłania wiadomości,
   - do momentu odebrania wiadomości zwrotnej dla wiadomości synchronicznej lub zakończenia operacji asynchronicznej.
   - zwykle z wyjątkiem: wiadomości własnej, konstruktora i destruktora.
6. **Zastosuj odpowiednie fragmenty połączone:**
   - obejmij nimi wszystkie linie życia, których dotyczą;
   - jeśli to możliwe, nie obejmuj nimi innych linii życia.
7. **Dodaj notatki:**
   - aby wyjaśnić niejasności interakcji.

Uwaga!

- **Definiowanie linii życia i wiadomości oraz używanie połączonych fragmentów musi być zrozumiałe, ustandaryzowane i wewnętrznie spójne:**
  - np. sposób modelowania współbieżności.
- **Diagram sekwencji powinien być możliwie jak najprostszy i wizualnie czytelny:**
  - skupiony na przedstawieniu określonego procesu biznesowego albo określonej implementacji operacji klasy lub obiektu;
  - ograniczony do wyjaśnienia interakcji między liniami życia czyli przepływu sterowania i danych między nimi;
  - bez zbędnych szczegółów, w tym implementacyjnych (w przypadku modelowaniu operacji klasy lub obiektu).

5

Przykłady

Przykłady

zobacz: Inżynieria Oprogramowania, Z. Kruczkiewicz, PWr, wykład 4

- Projekt przypadku użycia „Wstawianie nowego produktu”. (strona 20—37).
- Projekt przypadku użycia „Dodawanie rachunku”. (strona 38—47).
- Projekt przypadku użycia „Dodawanie zakupu”. (strona 48—60).
- Projekt przypadku użycia „Obliczanie wartości rachunku”. (strona 61—74).

6

Diagram czasowy

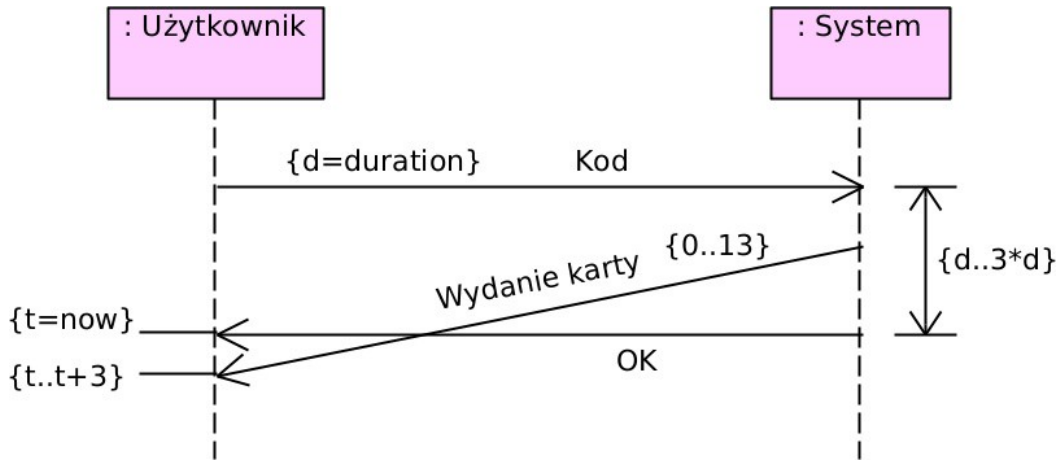
Diagram czasowy /timing diagram/

- Pokazuje **zmiany stanów linii życia w czasie:**
  - pionowa oś pokazuje różne stany,
  - pozioma oś pokazuje zdarzenia zmiany stanów (czas biegnie w prawo) oraz ich parametry czasowe.
- Nazwa linii życia:
  - ogólna – dla diagramu konceptualnego;
  - ***instancja:klasa*** – uczestnikiem jest konkretna instancja klasy;
  - ***:klasa*** – uczestnikiem jest anonimowa instancja klasy lub sama klasa;
  - ***self*** – uczestnikiem jest posiadacz tego diagramu.
- Linie życia umieszcza się w osobnych torach (odpowiednik partycji z diagramu czynności).
- **Wiadomość** może łączyć zdarzenia zmiany stanu dwóch linii życia:
  - zdarzenie wysłania wiadomości → zdarzenie otrzymania wiadomości.

Przykład z diagramem sekwencji ze str. 21

na podst. Unified Modeling Language (UML)

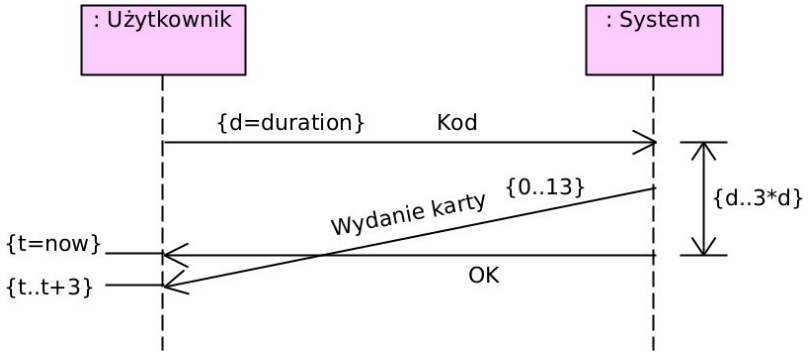
- **Użytkownik** wysyła do **Systemu** wiadomość **Kod**:
  - czas wysyłania **Kod** jest mierzony i zapisywany jako **d**.
- **System** odpowiada wysłaniem do **Użytkownika** wiadomości trwającej **Wydanie karty**, a następnie wiadomości **OK**:
  - czas wysyłania **Wydanie karty** trwa od 0 do 13 jednostek;
  - odstęp czasu między wysłaniem (i odebraniem) **Kod**, a wysłaniem (i odebraniem) **OK** wynosi od **d** do **3*d**;
  - moment wysłania (i otrzymania) **OK** jest zapisywany jako **t**.
  - moment otrzymania **Wydanie karty** wynosi od **t** do **t+3**.

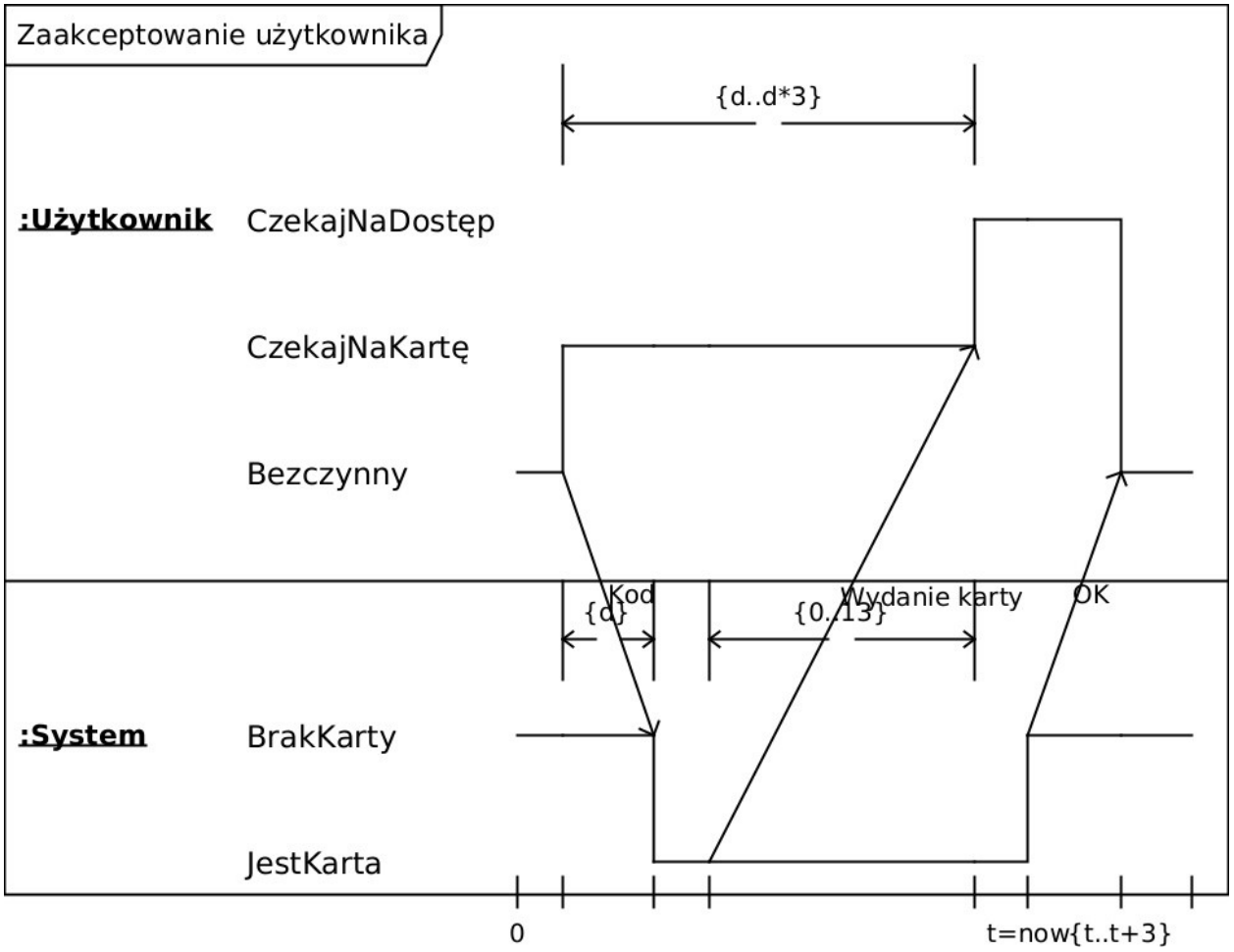


Przykład diagramu z 2 liniami życia i wiadomościami

na podst. Unified Modeling Language (UML)

- Ta sama interakcja między **Użytkownikiem** i **Bankiem**.
- Stany Użytkownika:
  - **CzekajNaDostęp**,
  - **CzekajNaKartę**,
  - **Bezczynny**.
- Stany Systemu:
  - **BrakKarty**,
  - **JestKarta**.





KONIEC