# Tworzenie warstwy zasobów – projektowanie metodą strukturalną

Autor
Zofia Kruczkiewicz

Programowanie i wdrażanie systemów informatycznych

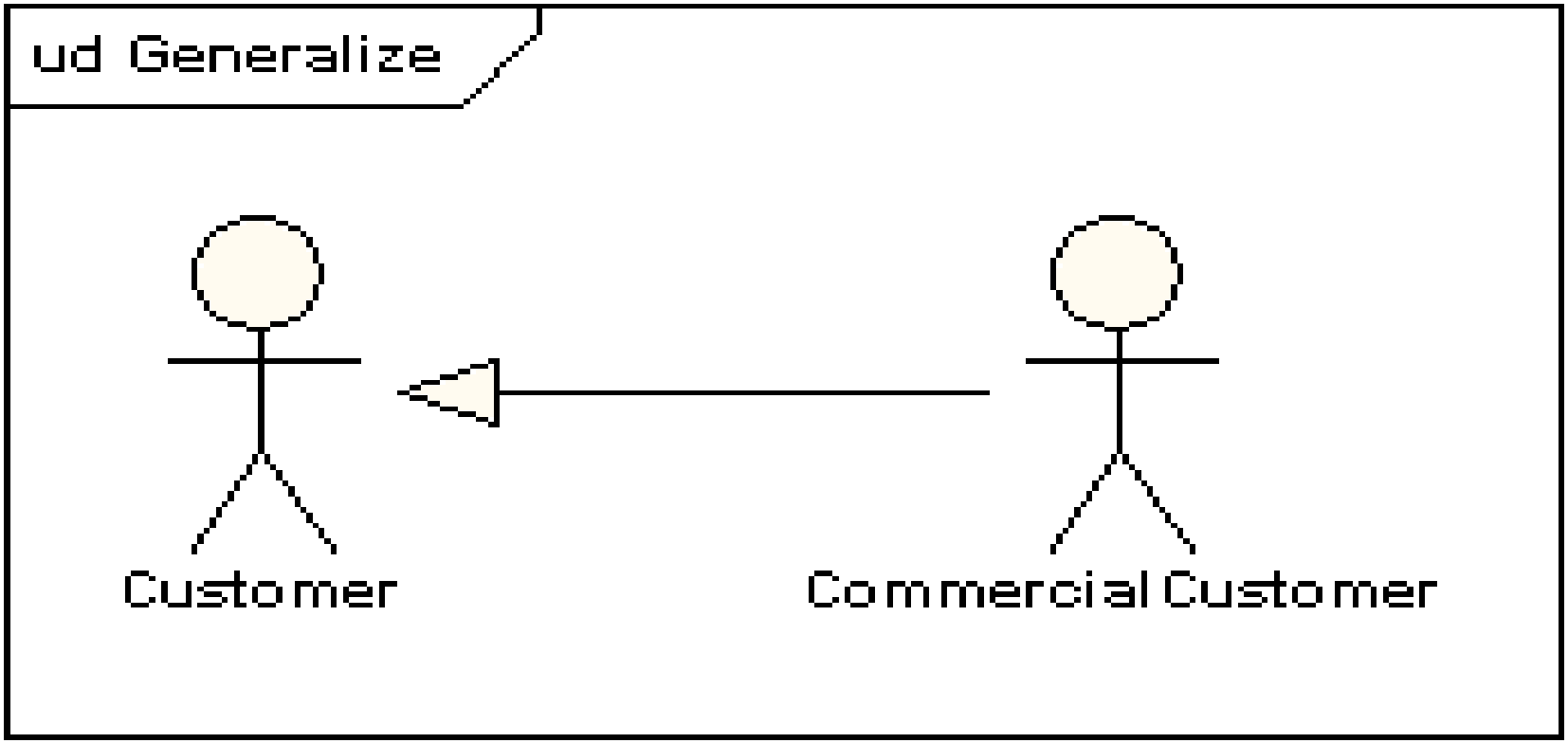
# 1. Zasady modelowania wymagań funkcjonalnych systemu za pomocą przypadków użycia

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/analizasi/Wykladasi3.pdf

## Identyfikacja aktorów i przypadków użycia – przypadek prostego systemu

- Należy wyznaczyć granice systemu w ramach środowiska
- Należy zdefiniować wymagania systemu:
  - należy podać użytkowników - aktorów systemu, zależności między aktorami typu dziedziczenie (***generalization***) lub powiązanie (***association***),
  - oczekiwane funkcje systemu (przypadki użycia), powiązania między aktorami i przypadkami użycia oraz zależności między przypadkami użycia
- Należy opisać każdy przypadek użycia np. wg szablonu podanego dalej

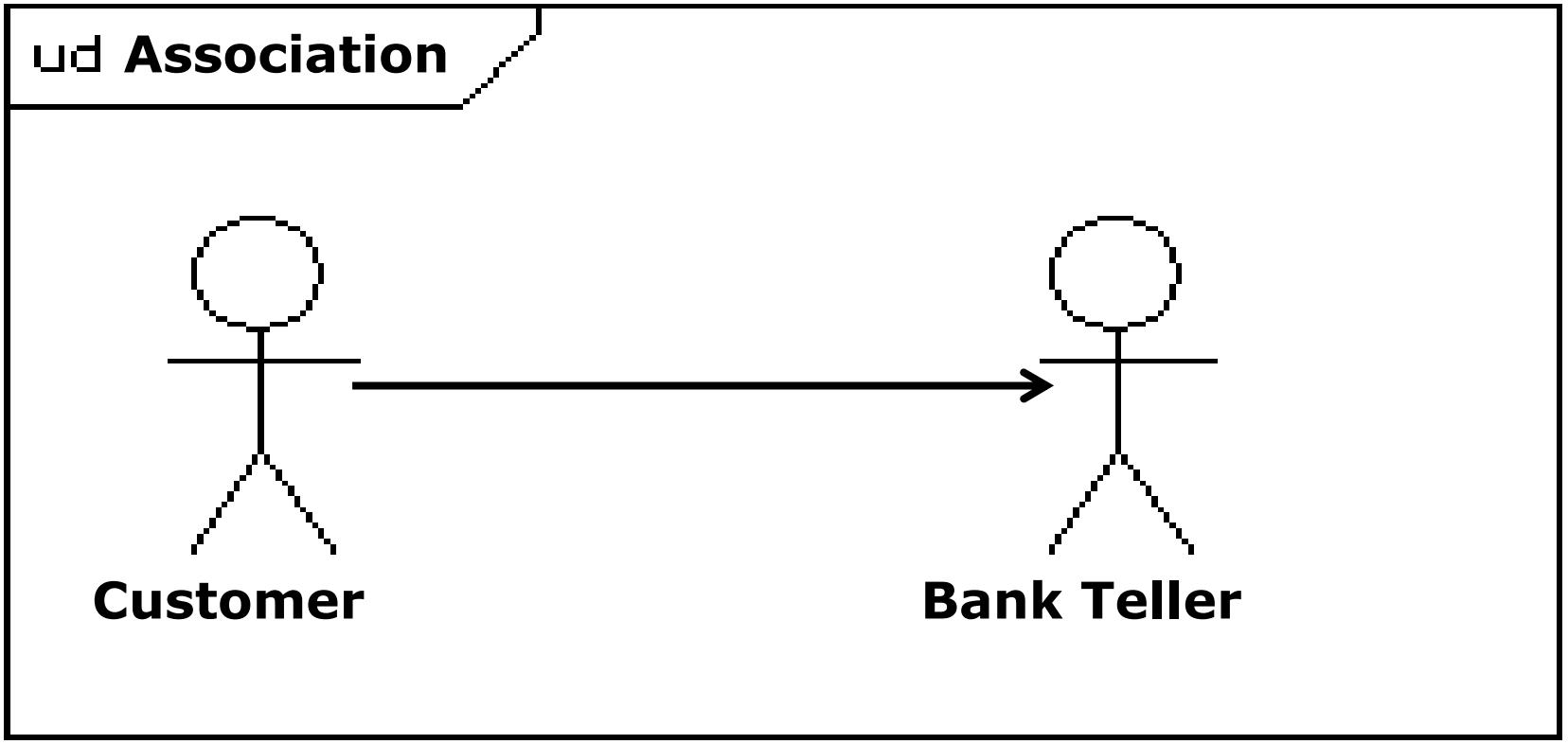
# Powtórka: Diagramy przypadków użycia UML




**Związek miedzy aktorami typu Generalization**

**Actors** mogą genaralizować innych **Actors.**

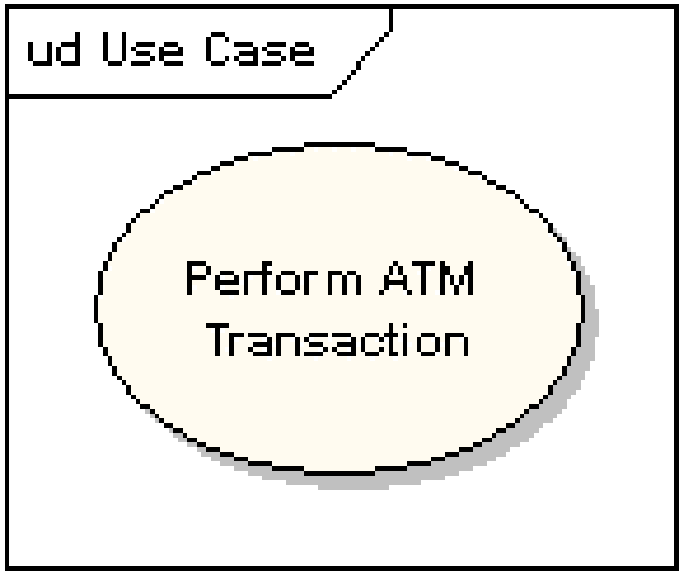
Oznacza to dziedziczenie funkcji (przypadków użycia) przez aktora **CommercialCustomer** od aktora **Customer** oraz korzystanie z nowych przypadków użycia

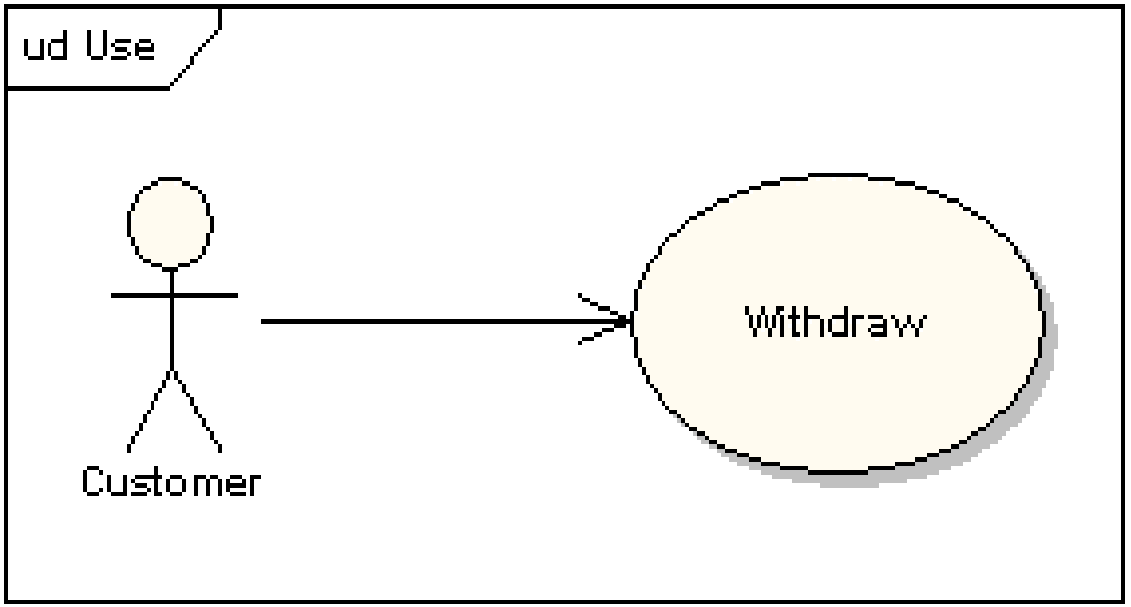



**Związek miedzy aktorami typu Association**

**Actors** mogą być powiązani z innymi **Actors**, czyli mogą pośredniczyć w dostępie do przypadków użycia.

Oznacza to, że aktor **Customer** za pośrednictwem aktora **BankTeller** korzysta z jego przypadków użycia




## Przypadek użycia (Use Cases)

- Jednostka pracy
- Wysoki poziomom zewnętrznej obserwacji systemu
- Notacja – elipsa
- <<>> znak stereotypu oznaczającego właściwości związku

## Związek użycia przypadku użycia <<use>>

•Np. aktor *Customer* używa przypadku użycia *Withdraw* (pobiera pieniądze np. z konta)

# Opis aktorów

| AKTOR | OPIS | PRZYPADKI UŻYCIA |
|---|---|---|
| Nazwa | *Opis celu i obowiązków* | Lista używanych przypadków użycia powiązanych wprost lub przez dziedziczenie od innych aktorów |

# Szablon opisu przypadku użycia:

- **Nazwa i opis**
- **Wymagania** funkcjonalne spełniane dla użytkownika
- **Ograniczenia** – warunki przed- po- przypadku użycia oraz nie zmieniające się na skutek wykonania przypadku użycia
- **Scenariusze** – sekwencja zdarzeń między systemem i zewnętrznymi użytkownikami (opis tekstowy)
- **Diagramy scenariuszy** – diagramy sekwencji
- **Dodatkowe informacje** – np. identyfikacja karty płatniczej przed dokonaniem wyciągu z konta

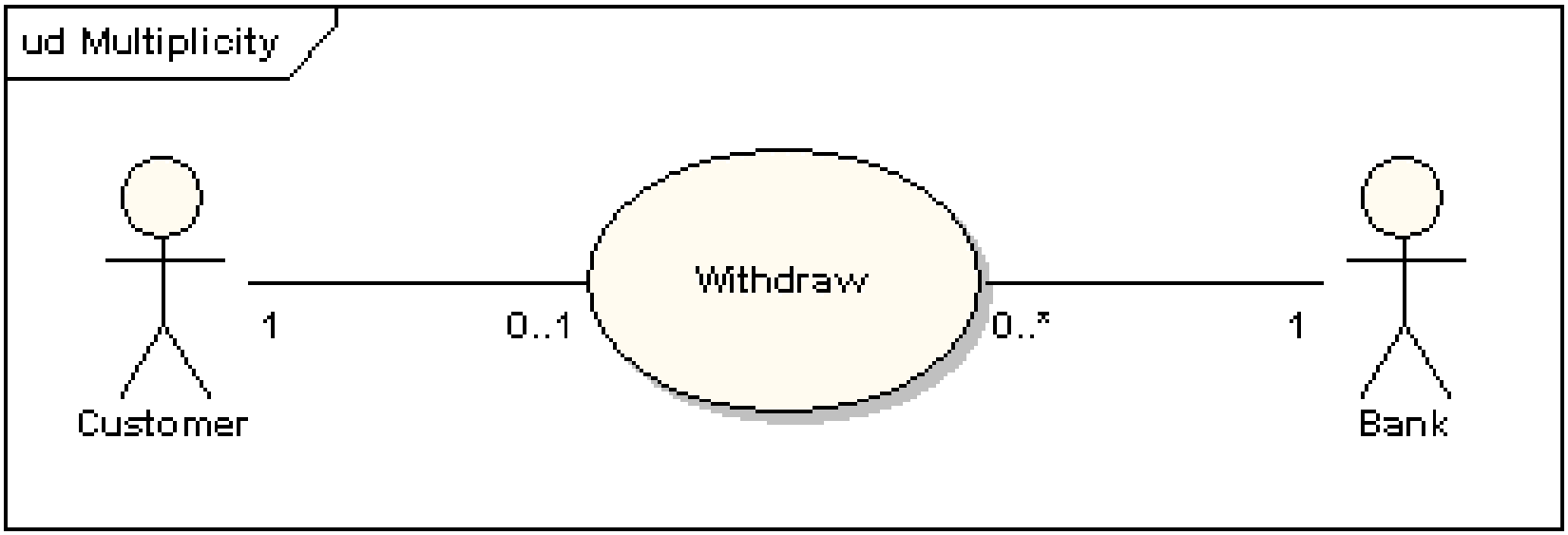
# Test PU (przykład): dodawanie nowego zakupu

**Dane wejściowe:**

Dane rachunku, dane produktu oraz jego ilość

**Dane wyjściowe:**

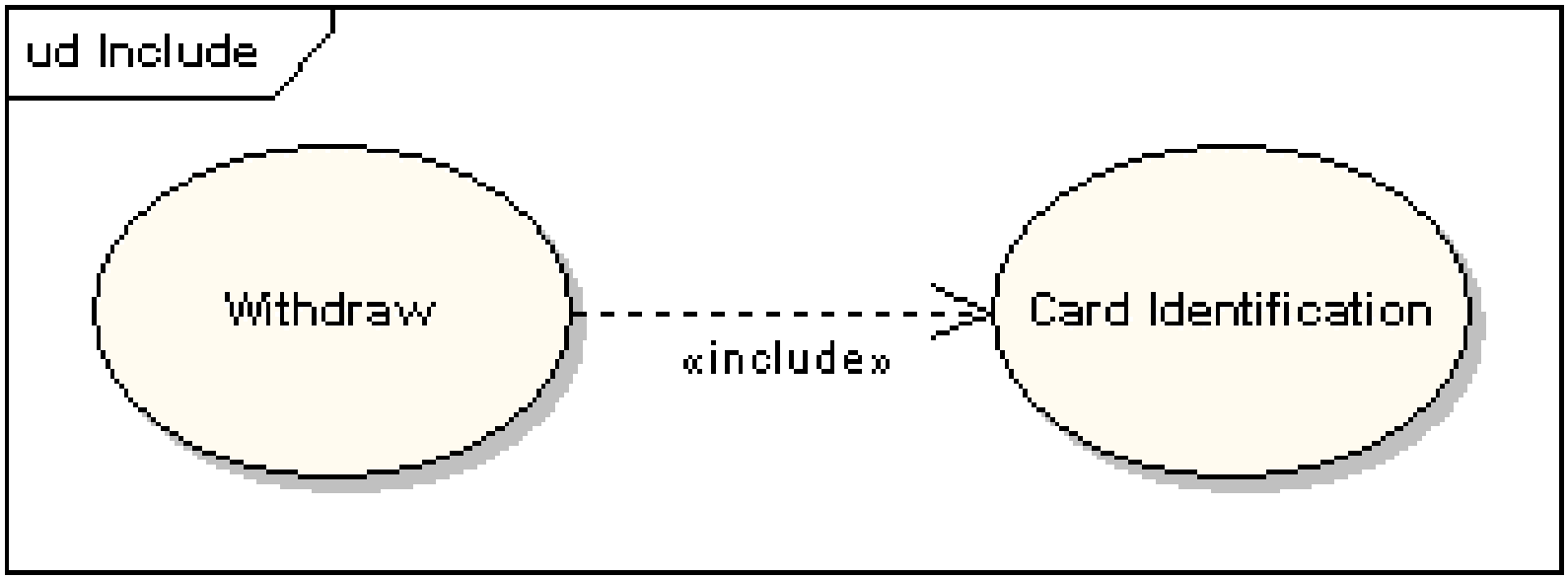
- Powstanie nowego zakupu, jeśli rachunek nie zawierał zakupu tego samego produktu
- Zwiększenie ilości zakupionego produktu o nową ilość, jeśli rachunek zawierał już zakup tego samego produktu




## Powiązania (Association) – liczność związku

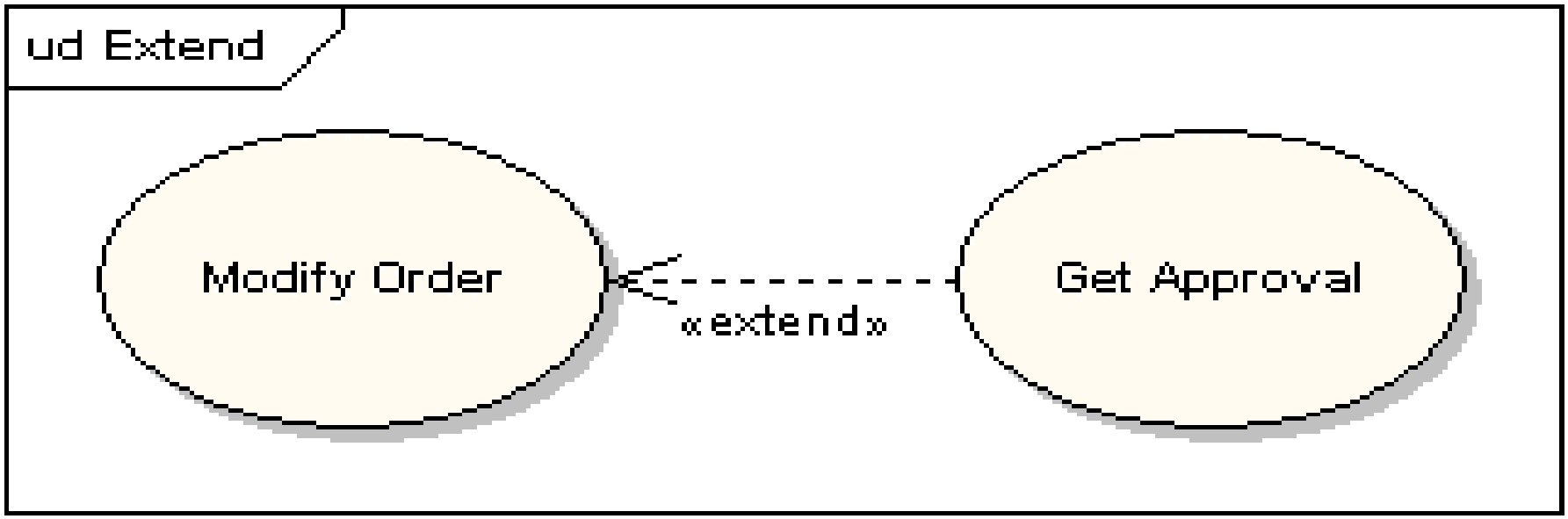
- Liczność instancji na końcu połączenia

Np. aktor *Klient (Customer)* ma tylko jedną sesję wypłacania pieniędzy w danym momencie (*Withdraw*) natomiast *Bank* może mieć ich wiele w tym samym czasie



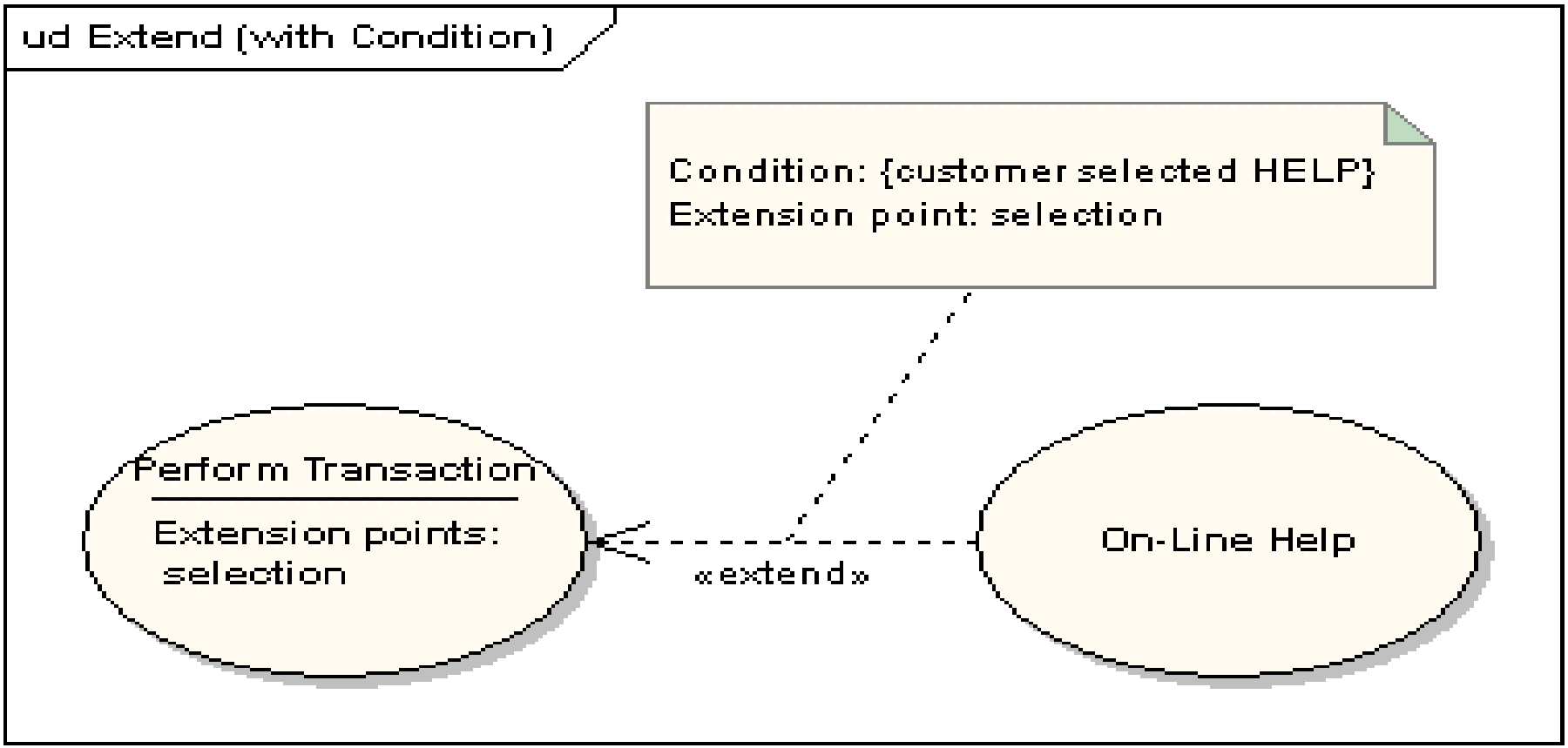

## Zawieranie <<includes>>

Przypadek użycia zawiera jeden lub wiele innych przypadków użycia eliminując powtarzanie funkcjonalności systemu dzięki tej wielożywalności, czyli zawieraniu
np.Pobranie z konta (*Withdraw*) zawsze zawiera identyfikację karty (*Card identification*)



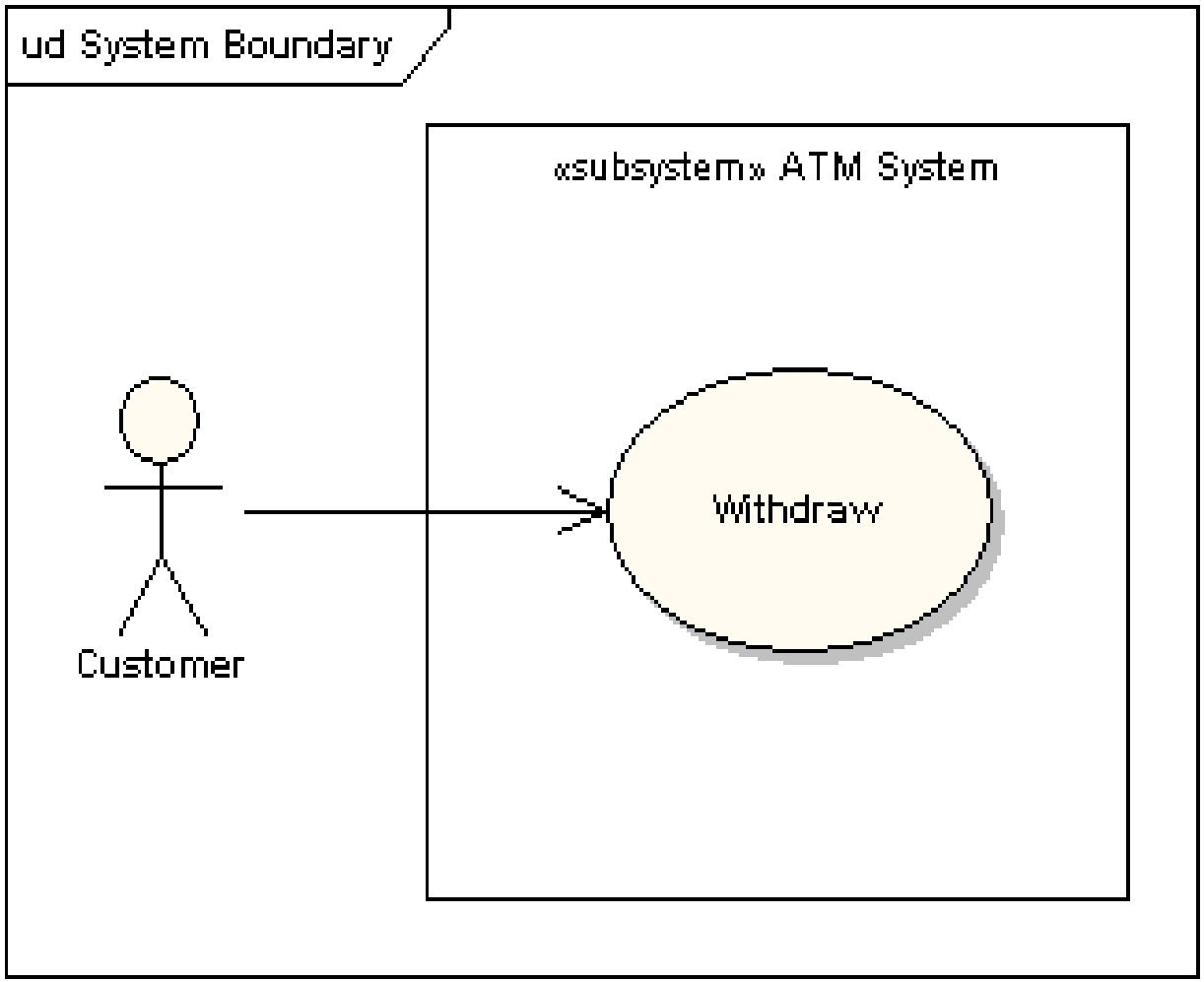

## Rozszerzanie <<extends>>

Jeden przypadek użycia może być użyty do rozszerzenia właściwości drugiego przypadku użycia
Np. Przypadek użycia *Zezwolenie (Get Approval)* opcjonalnie rozszerza właściwości przypadku użycia *Modyfikuj zlecenie (Modify Order)*



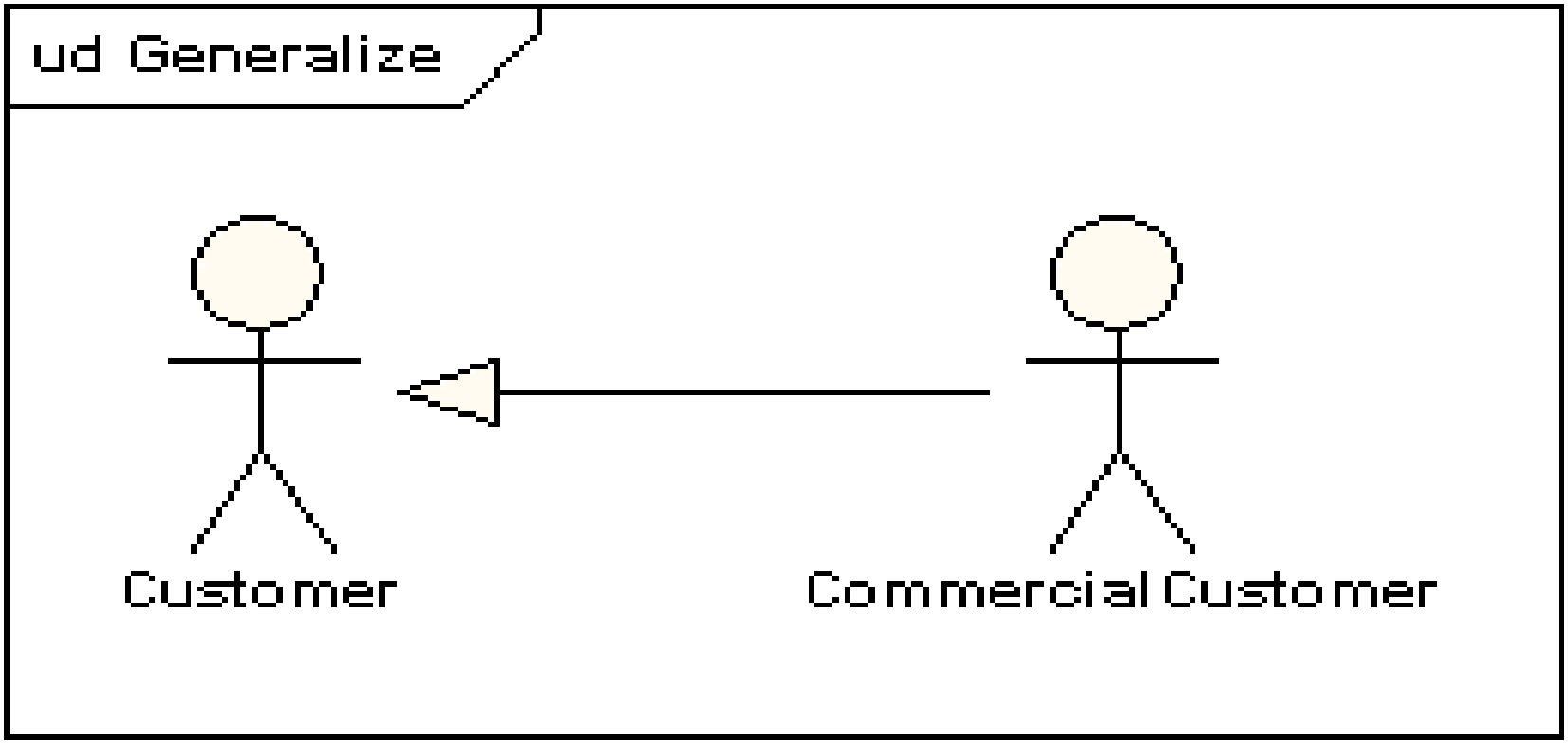

## Punkty rozszerzające (Extension Points)

np..Punkt, w którym rozszerzany przypadek użycia *Wykonanie tranzakcji* (*Perform Transaction*) jest rozszerzany przez rozszerzajacy przypadek użycia *Pomoc* (*On-Line Help*) zgodnie ze znaczeniem punktu rozszerzania np. przez *wybór* (*selection*)




## Granice systemu (System Boundary)

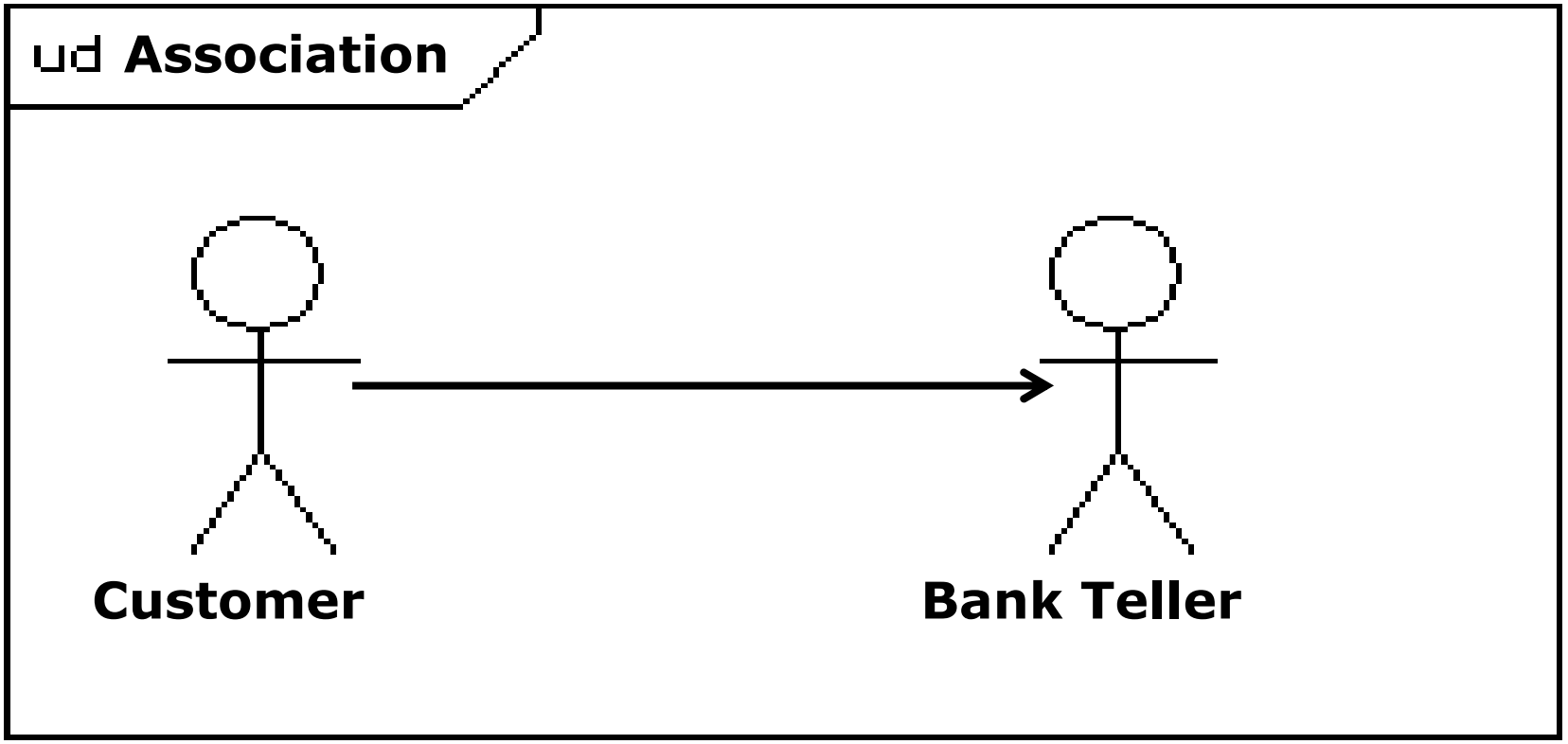
Zazwyczaj aktorzy są na zewnątrz systemu, a przypadki użycia wewnątrz systemu.




**Związek miedzy aktorami typu Generalization**

**Actors** mogą genaralizować innych **Actors.**

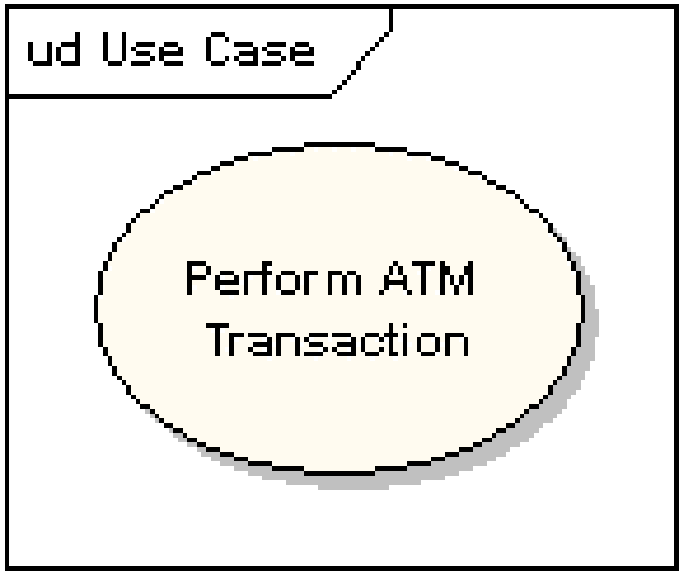
Oznacza to dziedziczenie funkcji (przypadków użycia) przez aktora **CommercialCustomer** od aktora **Customer** oraz korzystanie z nowych przypadków użycia

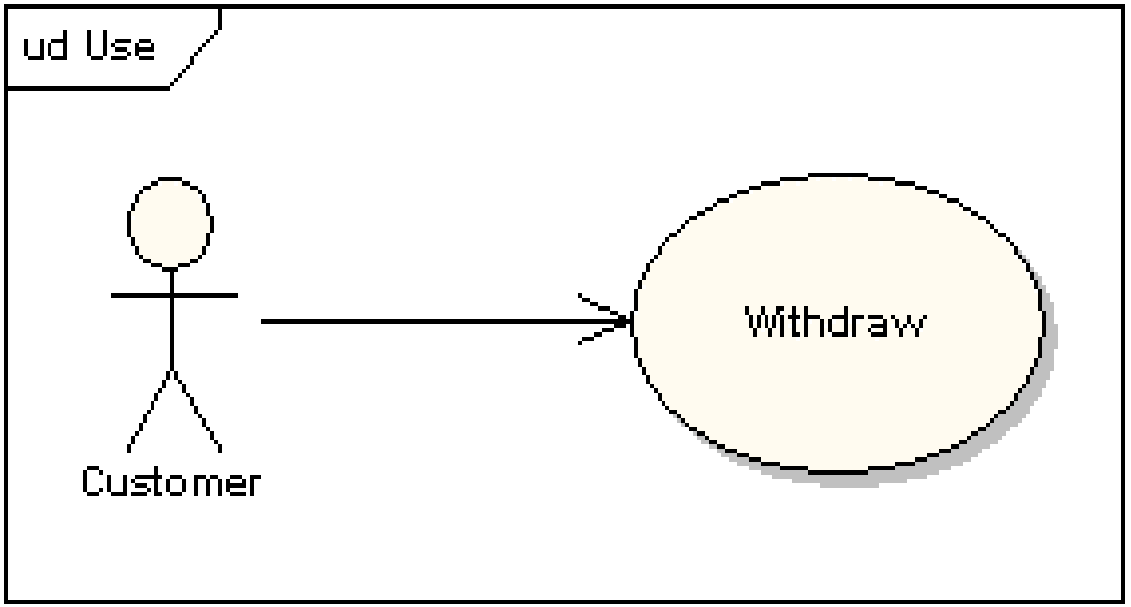



**Związek miedzy aktorami typu Association**

**Actors** mogą być powiązani z innymi **Actors**, czyli mogą pośredniczyć w dostępie do przypadków użycia.

Oznacza to, że aktor **Customer** za pośrednictwem aktora **BankTeller** korzysta z jego przypadków użycia



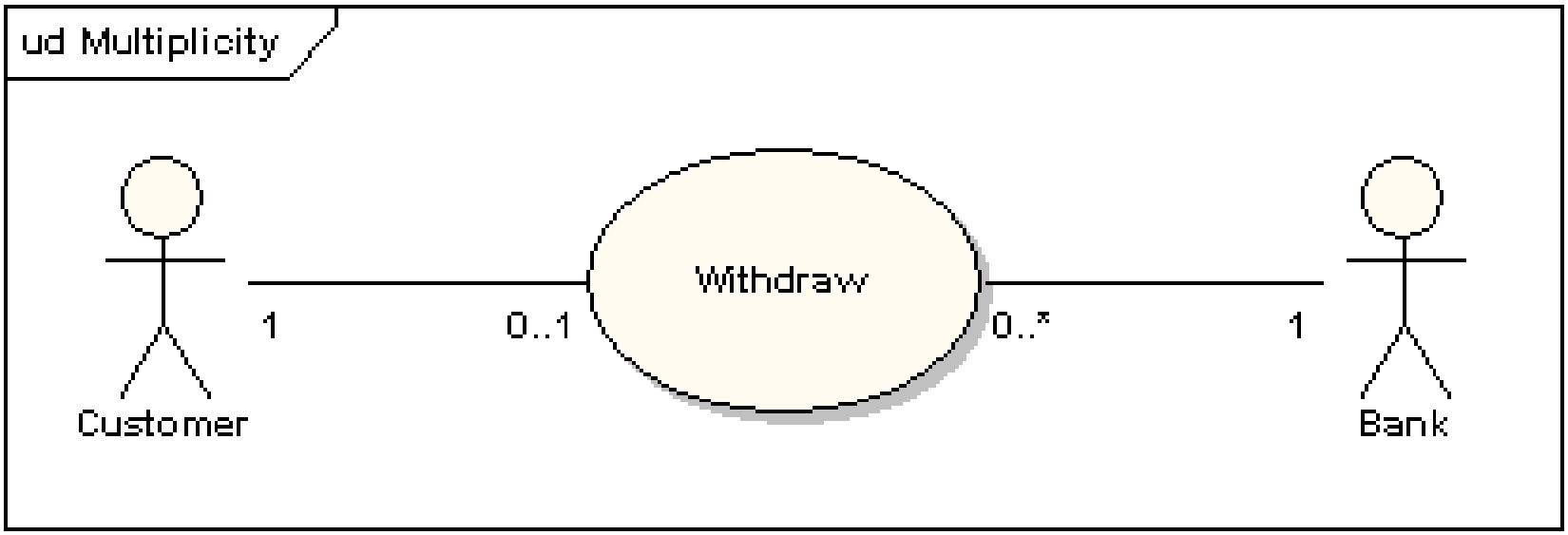




## Przypadek użycia (Use Cases)

- Jednostka pracy
- Wysoki poziomom zewnętrznej obserwacji systemu
- Notacja – elipsa
- <<>> znak stereotypu oznaczającego właściwości związku

## Związek użycia przypadku użycia <<use>>

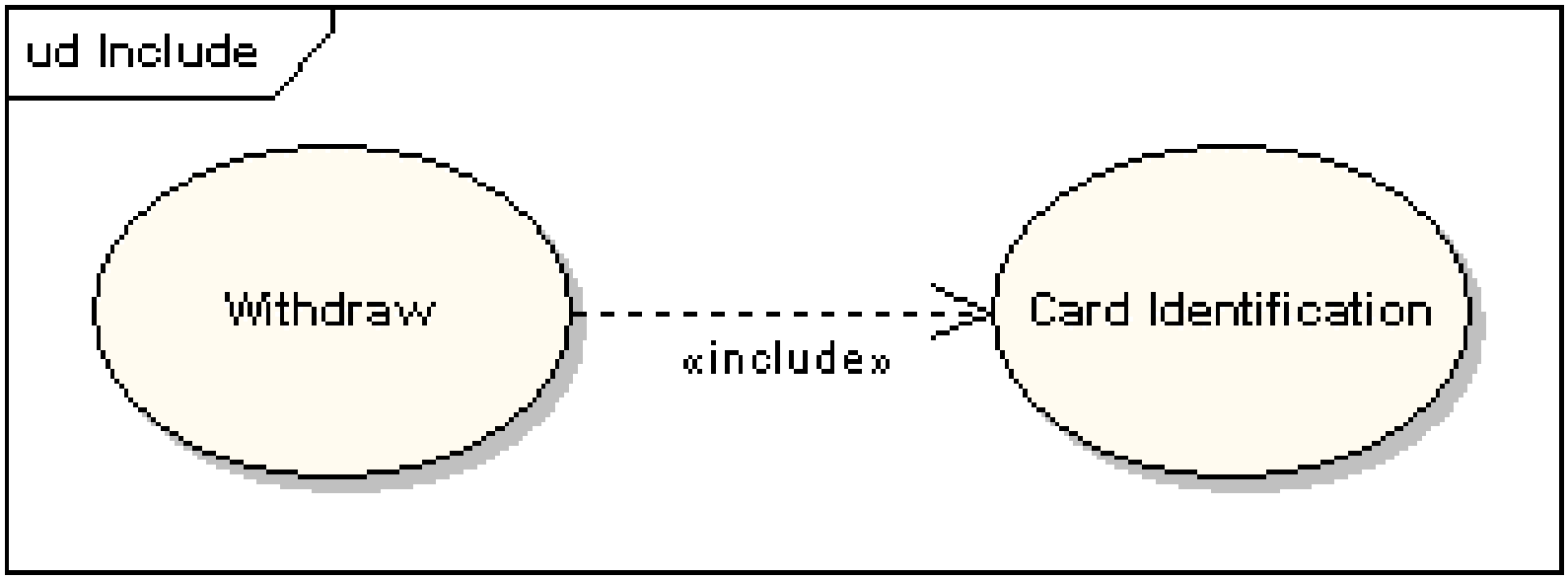
•Np. aktor *Customer* używa przypadku użycia *Withdraw* (pobiera pieniądze np. z konta)




## Powiązania (Association) – liczność związku

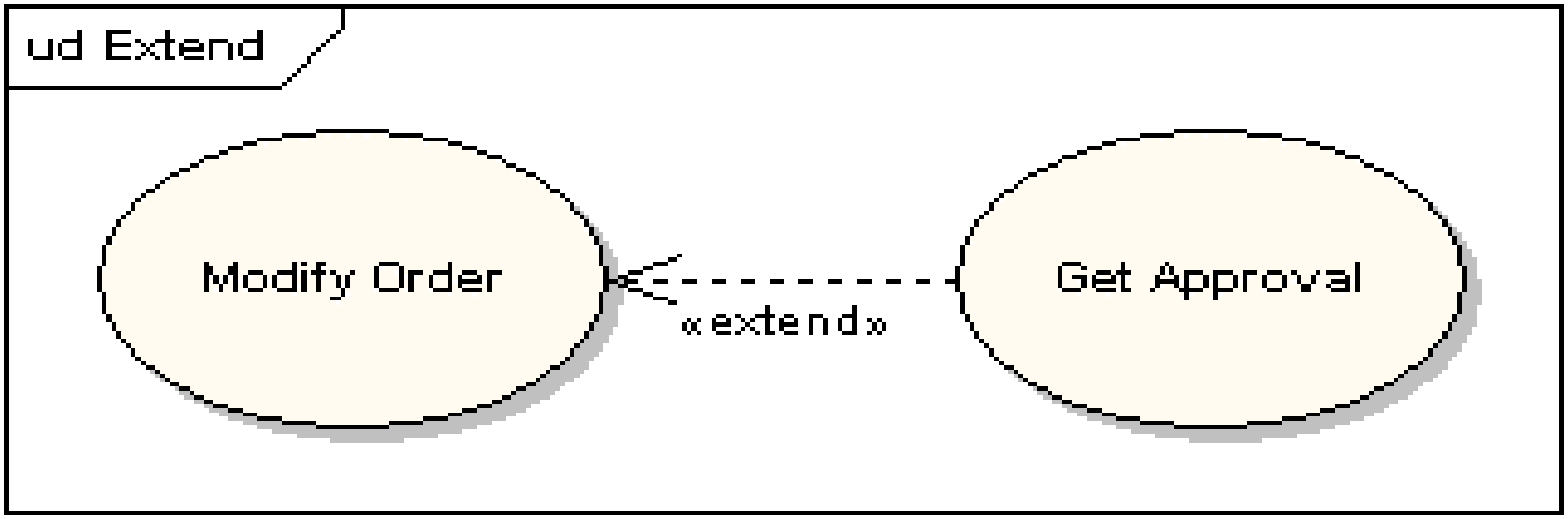
- Liczność instancji na końcu połączenia

Np. aktor *Klient (Customer)* ma tylko jedną sesję wypłacania pieniędzy w danym momencie (*Withdraw*) natomiast *Bank* może mieć ich wiele w tym samym czasie



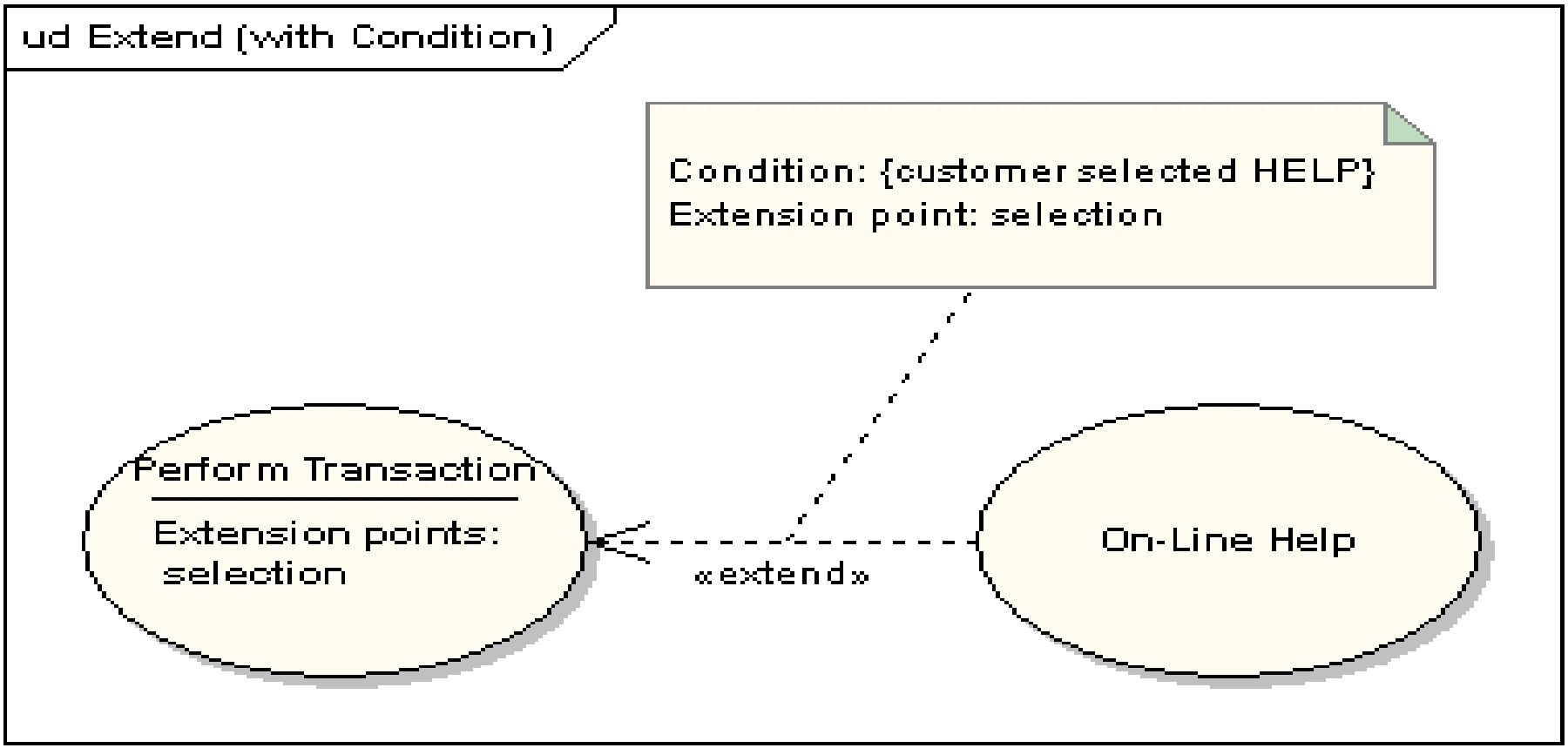

## Zawieranie <<includes>>

Przypadek użycia zawiera jeden lub wiele innych przypadków użycia eliminując powtarzanie funkcjonalności systemu dzięki tej wielożywalności, czyli zawieraniu
np.Pobranie z konta (*Withdraw*) zawsze zawiera identyfikację karty (*Card identification*)



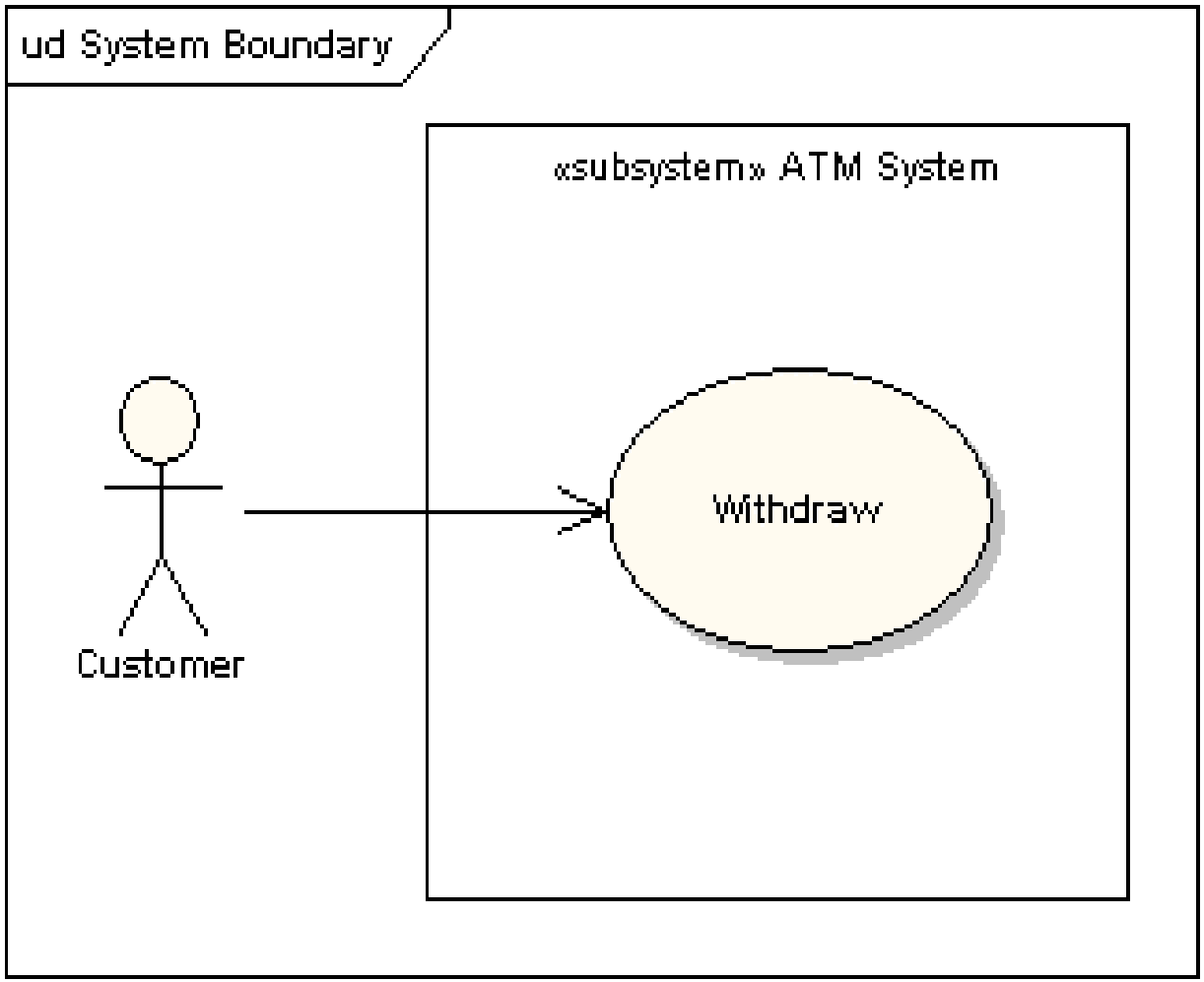

## Rozszerzanie <<extends>>

Jeden przypadek użycia może być użyty do rozszerzenia właściwości drugiego przypadku użycia
Np. Przypadek użycia *Zezwolenie (Get Approval)* opcjonalnie rozszerza właściwości przypadku użycia *Modyfikuj zlecenie (Modify Order)*




## Punkty rozszerzające (Extension Points)

np..Punkt, w którym rozszerzany przypadek użycia *Wykonanie tranzakcji* (*Perform Transaction*) jest rozszerzany przez rozszerzajacy przypadek użycia *Pomoc* (*On-Line Help*) zgodnie ze znaczeniem punktu rozszerzania np. przez *wybór* (*selection*)




## Granice systemu (System Boundary)

Zazwyczaj aktorzy są na zewnątrz systemu, a przypadki użycia wewnątrz systemu.

# Identyfikacja aktorów i przypadków użycia – przypadek prostego systemu

1. Należy wyznaczyć granice systemu w ramach środowiska
2. Należy zdefiniować wymagania systemu: należy podać użytkowników - aktorów systemu, zależności między aktorami typu dziedziczenie (***generalization***) lub powiązanie (***association***), oczekiwane funkcje systemu (przypadki użycia), powiązania między aktorami i przypadkami użycia oraz zależności między przypadkami użycia
3. Należy opisać każdy przypadek użycia wg szablonu

# Odp.1: Wytyczne przy modelowaniu granic systemu

- Należy zidentyfikować aktorów działających wokół systemu. Oznacza to wyznaczenie grup użytkowników korzystających w określonym celu z projektowanego systemu (zarządzanie, pielęgnacja, usługi)
- Należy uporządkować aktorów wg zależności typu powiązanie np. Klient korzysta z usług Sprzedawcy lub dziedziczenia: Komercyjny Klient dziedziczy przypadki użycia od Klienta
- Należy powiązać aktorów z przypadkami użycia za pomocą powiązań nadając im zidentyfikowane znaczenie za pomocą stereotypu wg podanych definicji

# Odp.2: Wytyczne przy modelowaniu wymagań stawianych systemowi

- Należy określić otoczenie systemu, czyli zidentyfikować aktorów
- Dla każdego aktora należy podać działania, jakie każdy aktor oczekuje od systemu (osiągnięcie celu przez aktora)
- Działania należy zapisać jako przypadki użycia
- Należy wyłączyć powtarzające się ciągi działań i zastąpić je jednym nowym połączonym relacją **<<include>>** lub **<<extends>>** lub **<<use>>** lub zwykłym powiązaniem typu ***association*** bez stereotypu
- Należy uwzględnić tych aktorów, przypadki użycia oraz zidentyfikowane powiązania między nimi
- Można dodać do każdego aktora i przypadku użycia notatkę opisującą wymagania niefunkcjonalne (np. sprzęt, język, korzystanie z Internetu)

# Odp.3: Wytyczne przy modelowaniu przypadków użycia

- Należy opisać główny i nadzwyczajne ciągi zdarzeń każdego przypadku użycia podając: czynności i dane używane podczas działania przypadku użycia
- Należy zdefiniować testy systemu w odniesieniu do wybranego aktora i powiązanego za nim jednego lub grupy przypadków użycia podając stan początkowy i końcowy oznaczający powodzenie testu przypadku użycia

  (np. *Wstawianie nowej książki jest możliwe tylko wtedy, gdy istnieje już jej tytuł w katalogu oraz posiada unikatowy numer. Po wstawieniu tej książki nie może być dwóch książek o tym samym numerze*)

# Przykłady

- Przykłady wymagań i opisu przypadków użycia (slajdy 11-20)

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/analizasi/Wykladasi5_1.pdf

- Przykłady opisu biznesowego, wymagań i opisu przypadków użycia (slajdy 3 - 13)

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/analizasi/Laboratorium2_3_4.pdf

# 2. Identyfikacja encji w diagramie związków encji na podstawie opisów przypadków użycia

(na podstawie Lech Banachowski, BazyDanych, Tworzenie aplikacji)

- Dane przechowywane w bazie danych służą do odpowiedzi na polecenia (pytania, wstawianie, usuwanie i modyfikacja) wynikające z przypadków użycia.
- W tym celu należy zidentyfikować dane występujące w przypadkach użycia – zarówno te, które należy dostarczyć oraz te, które należy uzyskać.
- Należy zaprojektować pośredni model między wymaganiami przedstawionymi za pomocą przypadków użycia a docelowym schematem bazy danych **– czyli diagram związków encji ERD**.
  - Ten pośredni model powinien w *sposób jednoznaczny* wyrażać wymagania użytkowników, umożliwiające klientowi sprawdzenie, czy analityk systemu dobrze zrozumiał ich intencje i specyfikę działania firmy.
  - Ten *pośredni model jest prostszy* od schematu bazy danych, ponieważ abstrahuje od szczegółów implementacyjnych, które muszą być później opracowane przez projektanta bazy danych, aby baza danych mogła powstać i spełniać pozostawione przed nią zadania.

# 2.1. Definicje związane z diagramem związków encji:

- Encja (obiekt) – coś, co istnieje, co jest odróżniane od innych, o czym informację trzeba znać lub przechowywać. Encje o tych samych własnościach tworzą zbiory encji. Na diagramie encje prezentuje się za pomocą prostokąta z podaną nazwą – oznacza to **typ encji,** gdyż encja jest traktowana jako konkretny **egzemplarz encji** np., *Tytul_ksiazki*
- Atrybut jest to własność encji danego typu, reprezentowana pewną wartością np. łańcuch znaków, liczba całkowita, liczba rzeczywista itp. **Atrybut powinien opisywać encję**, przy której się go umieszcza. Należy odróżnić atrybuty, które:
  - atrybuty opisujące dane encję (są jej atrybutami „rodzonymi”),
  - atrybuty reprezentujące związki danej encji z innymi encjami.

- W pierwszej fazie projektowania identyfikuje się tylko atrybuty opisujące daną encję.
  - Dla każdego egzemplarza encji każdy jej atrybut opisujący powinien przyjmować pojedynczą, atomową wartość.
  - W jednej chwili encja powinna mieć tylko jedną wartość dla każdego atrybutu.
- Związek jest to uporządkowana lista encji, poszczególne encje mogą występować wielokrotnie.
  - Każdy związek określa pewną relację między zbiorami egzemplarzy encji wchodzącymi w skład związku.
  - Relację nazywamy instancją związku.
  - Związek jest opisywany przy użyciu zdania zawierającego nazwy encji, których dotyczy i określającego, na czym ten związek polega.

# Typy związków

- **związek wieloznaczny**, czyli wiele do wiele
  - np. Na podstawie związku między Pracownikiem i Projektem można powiedzieć, że wielu Pracownikach pracuje w jednym Projekcie i jeden Pracownik pracuje w wielu Projektach
- **związek jednoznaczny**, czyli wiele do jeden, jeden do wiele
  - Przykładem związku jeden do wiele: Tytul jest na wielu Książkach
  - Przykładem związku wiele do jeden: Wiele Książek ma jeden Tytul
- **związek jedno-jednoznaczny**, czyli jeden do jeden
  - Każdy Produkt może być kupiony, ale każdy Zakup dotyczy tylko jednego Produktu

# Informacje o relacjach

- Każdy fakt przechowywany w bazie danych powinien być wyrażany w niej tylko na jeden sposób
  - Informacja o Tytule powinna być zapisana tylko w jednym miejscu, a nie przy każdej książce, która ma ten Tytul.
  - Jeśli zapisujemy informacje o Wypożyczeniu Książki, to nie ma potrzeby zapisywania informacji o Tytule, gdyż można te informację wyprowadzić z Książki
  - Jeśli istnieją dwie odrębne encje w systemie sprzedaży: Pracownik i Klient, i jeśli Pracownik jest jednocześnie Klientem firmy, wówczas jego dane będą zapisane w dwóch różnych miejscach
- Każda dana w modelu powinna być istotna z punktu widzenia np. firmy
- Każda dana ważna w firmie, powinna być uwzględniona w modelu

# 2.2. Identyfikacja encji oraz ich atrybutów opisujących encje i związki

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/analizasi/Wykladasi5_1.pdf

Przykłady opisu biznesowego, wymagań i opisu przypadków użycia (slajdy 3 - 13)

| **Nazwa przypadków użycia – PU (podstawowe operacje na danych)** | **Atrybuty „rodzone”** | **Atrybuty reprezentujące związki** | **Atrybut identyfikujący encję** | **Nazwa Encji** |
|---|---|---|---|---|
| Szukanie produktu | Nazwa<br>Cena<br>Podatek<br>Promocja | | Id_produktu | Produkt |
| Wstawianie nowego produktu | Nazwa<br>Cena<br>Podatek<br>Promocja | | Id_produktu | Produkt |

| | | | | |
|---|---|---|---|---|
| Szukanie rachunku | Numer_rachunku | | Numer_rachunku | Rachunek |
| Wstawianie nowego rachunku | Numer_rachunku | | Numer_rachunku | Rachunek |
| Wstawianie nowego zakupu | Numer_rachunku<br>Ilosc_produktu<br>Nazwa<br>Cena<br>Podatek<br>Promocja | Numer_rachunku<br>Id_produktu | Numer_rachunku | Rachunek |
| | | | Id_zakupu | Zakup |
| | | | Id_produktu | Produkt |
| Obliczanie wartosci rachunku | *Wartosc_rachunku*<br>(atrybut wyliczeniowy= Cena_brutto, Ilosc_produktu)<br><br>*Cena_brutto*<br>(atrybut wyliczeniowy= Cena, Podatek, Promocja)<br>Podatek | Numer_rachunku | Numer_rachunku | Rachunek |
| | | | Id_zakupu | Zakup |
| | | | Id_produktu | Produkt |

# Identyfikacja liczności związków

- Rachunek zawiera informację o wielu Zakupach (pozycja rachunku) – jeden do wiele, Rachunek może istnieć bez Zakupow jeden do wiele..0
- Zakup należy do Rachunku – wiele do jeden,
- Zakup nie może istnieć bez Rachunku – wiele..0 do jeden
- Zakup zawiera dane o zakupionym Produkcie - jeden do jeden, w tym samym Rachunku może wystąpić tylko jeden Zakup z danym typem Produktu, można zwiększyć Ilosc_produktu,
- Produkt może istnieć bez Zakupu – jeden do jeden..0

# Diagram reprezentujący encje, atrybuty encji oraz związki między encjami