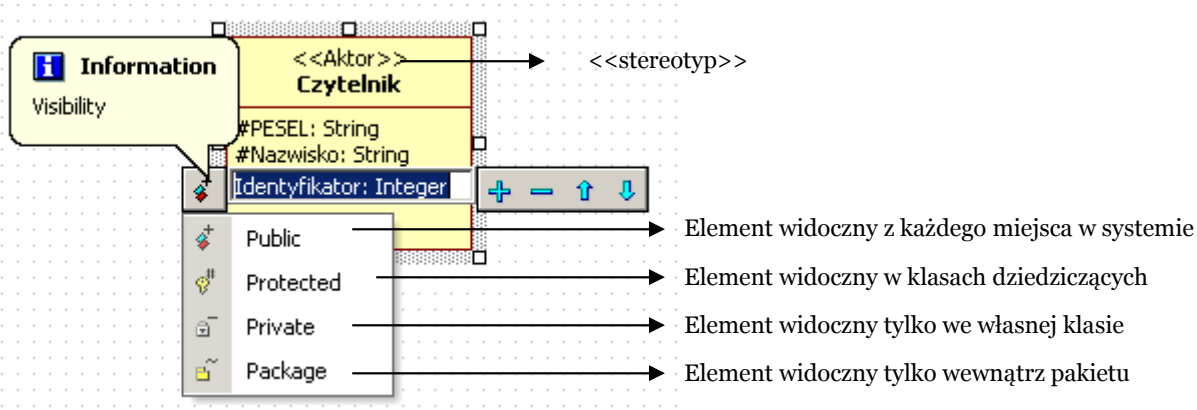
Krzysztof Regulski
AGH, WIMiIP

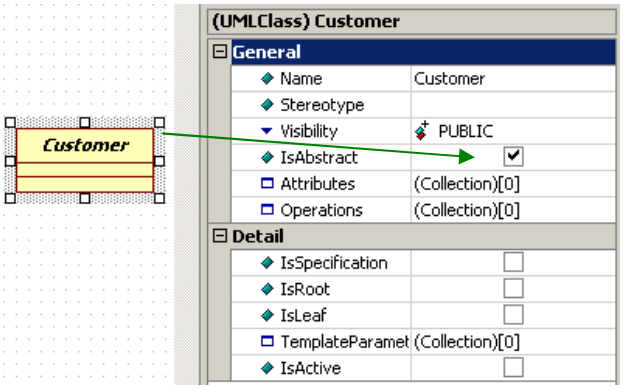
DIAGRAM KLAS

1) Zastosowanie

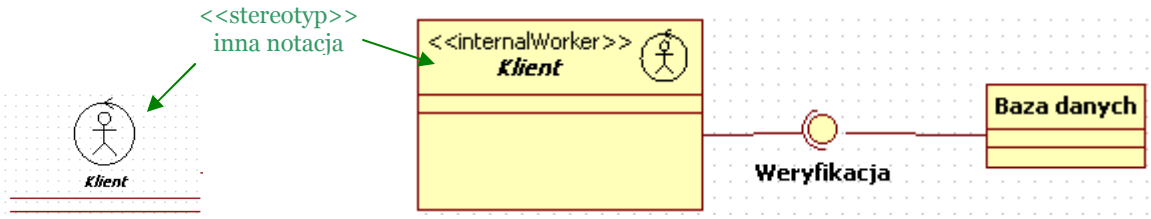
- Diagram klas jest abstrakcyjnym modelem dziedzinowym, ale może być także modelem logicznym systemu, co zależy od przyjętego poziomu abstrakcji.
- Diagram klas reprezentuje różne rodzaje bytów – klasy obiektów, które są potrzebne w systemie aby wykonać zadania zadeklarowane w Przypadkach Użycia.
- Klasy są typowym „przejawem" programowania obiektowego. Zapewniają podstawową korzyść technik obiektowych: hermetyzację. Klasy posiadają atrybuty (zestaw danych) oraz operacje (metody) pozwalające na operowanie tymi danymi. Atrybuty, operacje i klasy mogą zostać udostępnione lub ukryte w zależności od potrzeb.



- Diagram klas przedstawia również powiązania statyczne pomiędzy klasami. Głównym celem diagramu jest zobrazowanie współpracy klas.
- Ilość informacji i szczegóły techniczne na poziomie implementacyjnym pozwalają na automatyczne generowanie kodu na podstawie dobrze sformułowanych (projektowych) diagramów klas.
- **Klasy abstrakcyjne** służą do modelowania rzeczywistości, którą chcemy przenieść do logiki swojej aplikacji. Klasa abstrakcyjna definiuje w dużej ogólności pewien podstawowy model i zachowanie obiektu. Klasa abstrakcyjna nie posiada instancji, jedynie klasy dziedziczące mogą posiadać instancje. Oznaczamy ją kursywą.



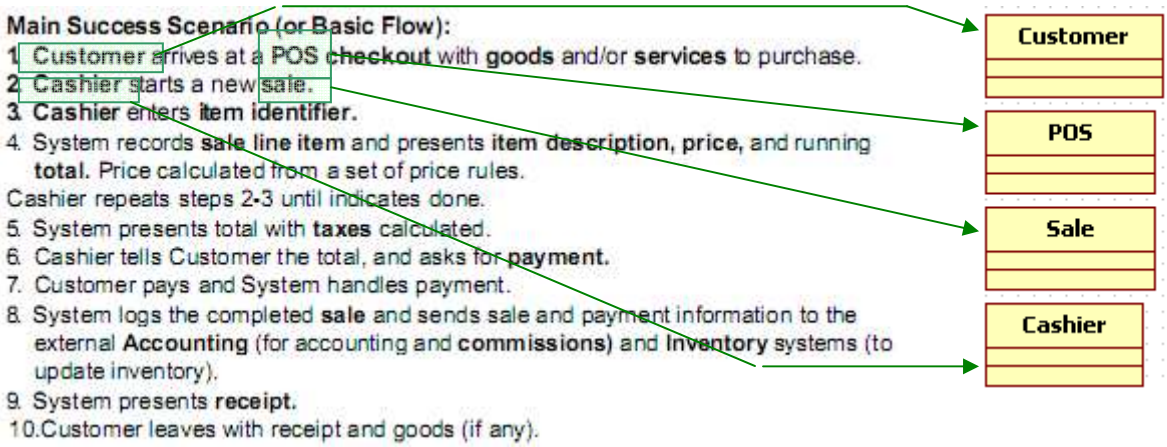
- Klasa abstrakcyjna jest dość mocno związana logicznie z obiektami z niej dziedziczącymi. **Interfejs** natomiast jedynie narzuca klasie jakie metody powinna posiadać, interfejs definiuje pewne właściwości, które klasa musi spełniać. Sam w sobie nie definiuje logicznej zależności pomiędzy sobą a klasą go implementującą, definiuje natomiast powiązanie pomiędzy wszystkimi klasami go implementującymi. Agreguje klasy obiektów na których można wykonać pewne operacje.



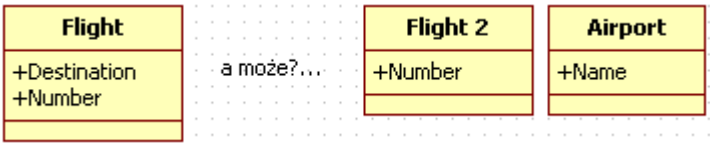
- **Stereotypy** wprowadzają meta-klasyfikację obiektów. Oznaczane przez «podwójne nawiasy» pozwalają nadać wspólną nazwę dla wspólnych własności - upraszczają model.

2) Konstrukcja

- Pierwszym krokiem w procesie ustalania klas może być wychwycenie rzeczowników w przepływie czynności:

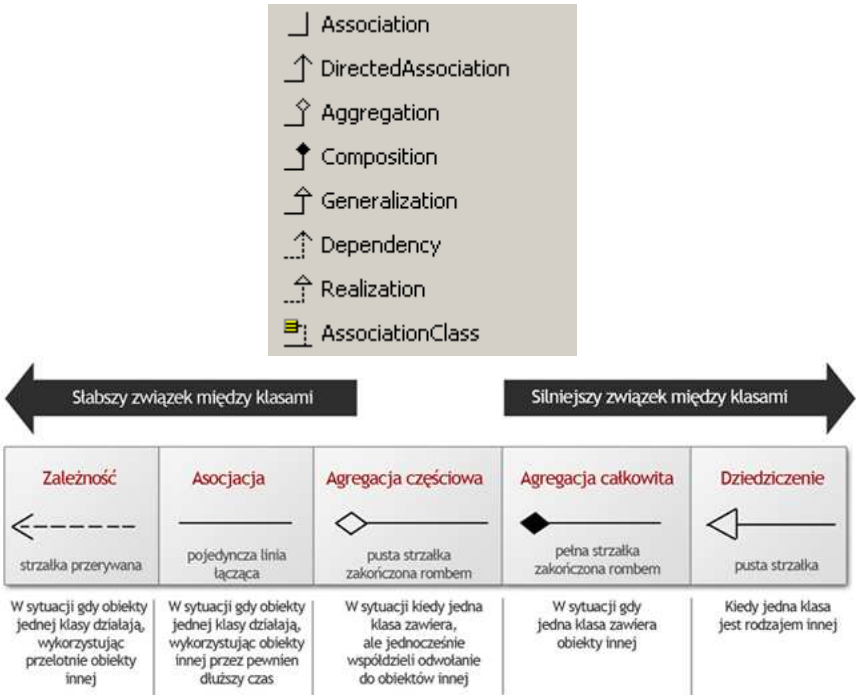


- Częstym błędem jest definiowanie elementów jako atrybuty, podczas gdy w rzeczywistości powinny być klasami. Praktyczna zasada: jeśli nie myślisz o czymś w kategoriach tekstu lub liczby to zapewne jest to klasa. Przykład:



- Proces budowania diagramów klas składa się z etapów:

  - identyfikacja klas i obiektów
  - identyfikacja związków pomiędzy klasami
  - identyfikacja i definiowanie atrybutów
  - identyfikacja i definiowanie operacji (metod) i komunikatów



- **Zależność** (dependenci) – kiedy klasa musi posiadać wiedzę o innej klasie aby używać obiektów danej klasy. Zależność oznacza, że obiekty będą jedynie **współpracować** ze sobą.

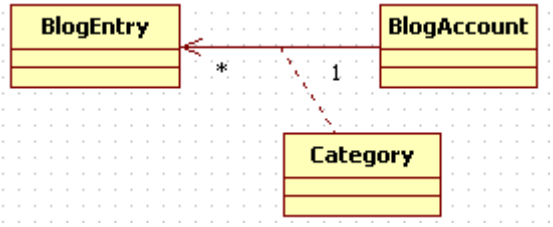
- **Asocjacja** – kiedy klasa wykorzystuje obiekty innej klasy jako atrybuty, kiedy przechowuje referencje do innych klas. Najczęściej mówimy, że jedna klasa „używa" innej.

- **Nawigowalność** asocjacji – określa wiedzę o sobie nawzajem obiektów uczestniczących w relacji.

konto „zna" wszystkie swoje wpisy, wpisy „znają" konto autora

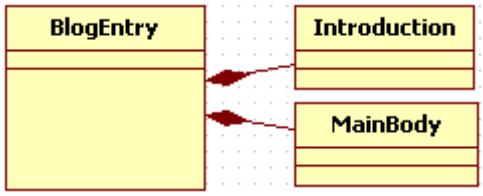
konto „zna" wszystkie swoje wpisy, wpisy nie odwołują się do konta autora

- **Klasa asocjacyjna** umożliwia opisanie za pomocą atrybutów i operacji nie obiektu, ale właśnie samej asocjacji pomiędzy klasami. Informacje przechowywane w klasie asocjacyjnej nie są związane z żadną z klas uczestniczących w asocjacji, dlatego wygodnie jest stworzyć dodatkową klasę i powiązać ją z relacją.

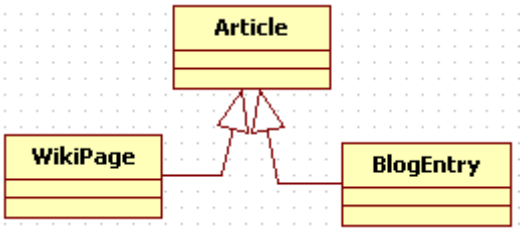


- **Agregacja** – kiedy klasa posiada i współdzieli obiekty innej klasy. Istnieje właściciel i obiekt podrzędny, które są ze sobą powiązane czasem swojego życia. Właściciel jednak nie jest wyłącznym właścicielem obiektu podrzędnego, zwykle też nie tworzy i nie usuwa go.

- **Kompozycja** jest najsilniejszą relacją łączącą klasy. Reprezentuje relacje całość-część, w których części są tworzone i zarządzane przez obiekt reprezentujący całość. Ani całość, ani części nie mogą istnieć bez siebie, dlatego czasy ich istnienia są bardzo ściśle ze sobą związane i pokrywają się: w momencie usunięcie obiektu całości obiekty części są również usuwane. Typowa fraza związana z taką relacją to "...jest częścią...".



- **Uogólnienie** (generalization, dziedziczenie) – tworzy hierarchię klas od ogólnych do bardziej szczegółowych. Pozwala wyłączyć części wspólne klas. Można powiedzieć, że jedna klasa „jest typem” innej.



- Instrukcje, jak rysować diagramy klas w StarUML znajdziecie w podręczniku użytkownika: http://staruml.sourceforge.net/docs/user-guide(en)/ch05_2.html