

Literatura

- K. Subieta „Obiektowość w projektowaniu i bazach danych”
- <http://www.omg.com/uml/>
- <http://www.rational.com/uml/>
- http://www.cetus-links.org/oo_uml.html

Universal Modeling Language

- graficzny język obiektowego modelowania oprogramowania
- wspomaga proces tworzenia oprogramowania, umożliwiając:
 - wizualizację
 - specyfikację
 - konstrukcję
 - dokumentacjęsystemu i jego części składowych

Elementy specyfikacji UML

rzeczy (ang. *things*) – abstrakcje mające sens same w sobie

np. klasa, interfejs, komponent, komunikat, stan, pakiet, notatka

relacje (ang. *relationships*) – semantyczne powiązania między rzeczami

np. zależność, związek, generalizacja

diagramy (ang. *diagrams*) – grupy rzeczy i relacji (utożsamiane z reprezentacją graficzną w postaci grafu, którego węzłami są rzeczy, a krawędziami – relacje)

np. diagram klas, obiektów, sekwencji, współdziałania, komponentów

Podstawowe założenia UML

1. graficzna reprezentacja \neq specyfikacja
2. ozdobniki – opcjonalne właściwości
3. typowe dychotomie:
 - klasa — obiekt
 - interfejs — implemetacja
4. rozszerzalność: stereotypy, znaczniki, ograniczenia

Modelowanie struktury

diagramy klas: klasy, interfejsy, pakiety, współdziałania (oraz relacje)

diagramy obiektów: obiekty (oraz relacje)

diagramy komponentów: komponenty (oraz relacje)

diagramy rozmieszczenia: węzły (oraz relacje)

Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Modelowanie zachowań

diagramy przypadków użycia: przypadki użycia, aktorzy oraz relacje

diagramy sekwencji: instancje (obiekty) oraz komunikaty

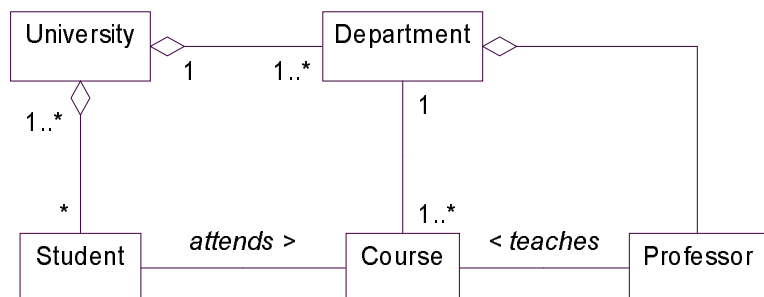
diagramy współdziałania: obiekty, połączenia oraz komunikaty

diagramy stanów: stany, przejścia oraz zdarzenia

diagramy aktywności: akcje, przepływ sterowania, obiekty

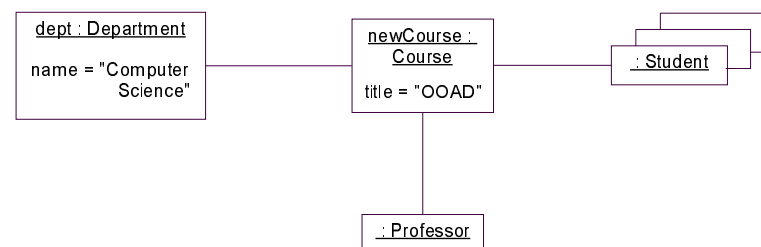
Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Przykładowy diagram klas



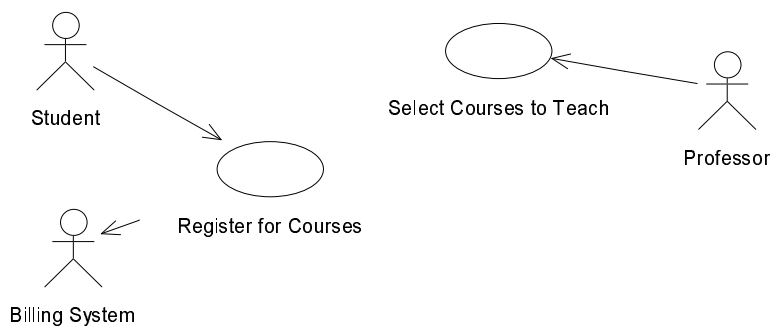
Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Przykładowy diagram obiektów



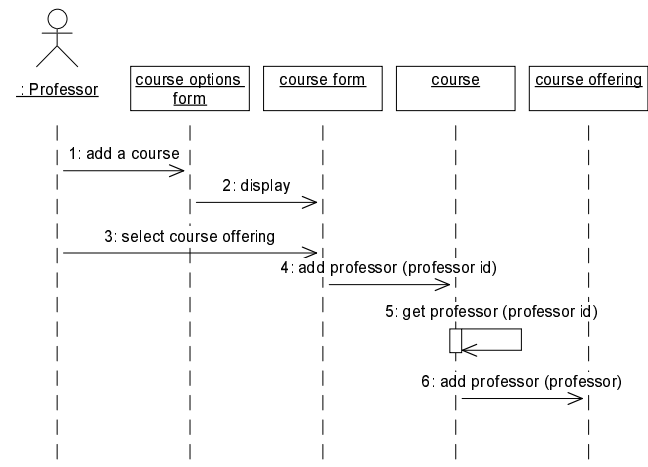
Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Przykładowy diagram przypadków użycia



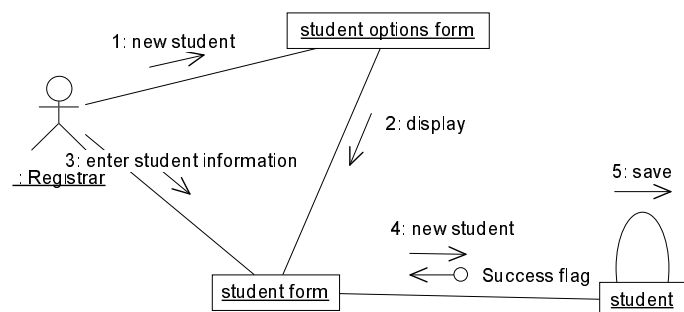
Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Przykładowy diagram sekwencji



Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Przykładowy diagram współdziałania

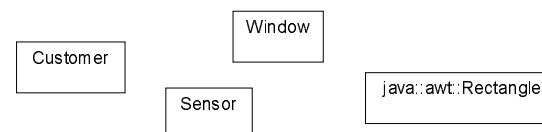


Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Klasy

klasa – opis obiektów o takich samych atrybutach, operacjach (usługach), relacjach i znaczeniu

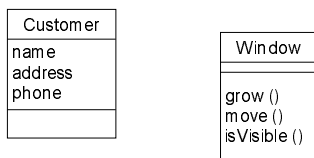
nazwy proste i kwalifikowane:



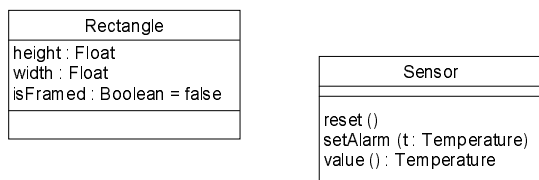
Copyright (c) 1997–2000 Marek Kisiel-Dorohinicki

Atrybuty i operacje

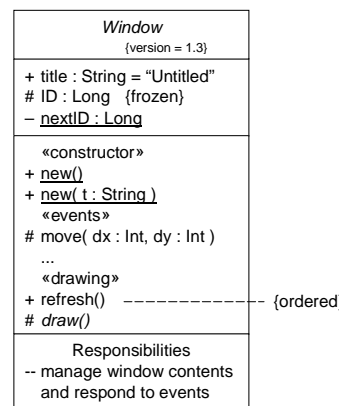
nazwy atrybutów i operacji:



typy i wartości początkowe atrybutów i sygnatury operacji:



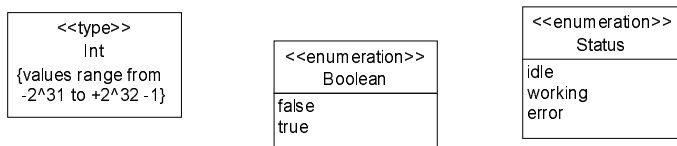
Inne właściwości



- ukrywanie pojedynczych atrybutów i usług lub całych przedziałów
- kategorie atrybutów i usług – **stereotypy**
- zakres i dostęp – **dekoracje**
- abstrakcyjne i konkretne klasy oraz operacje – **dekoracje**
- dodatkowe właściwości atrybutów (*changable*, *addOnly*, *frozen*) oraz operacji (*isQuery*) – **znaczniki**
- dodatkowe przedziały (obowiązki, sygnały, wyjątki) – **stereotypy**

Szczególne przypadki

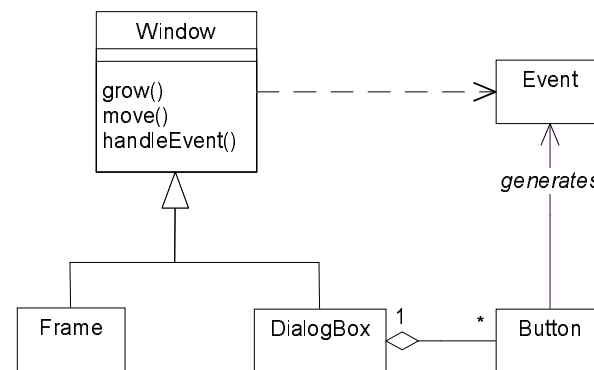
typy podstawowe:



elementy rzeczywistości spoza systemu:



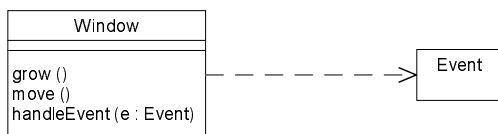
Relacje między klasami



Relacje zależności

pokazują, że jedna rzecz (klasa) „używa” innej, co oznacza, że zmiana specyfikacji jednej rzeczy (klasy) może wpływać na inną

- najczęściej: klasa jako argument operacji

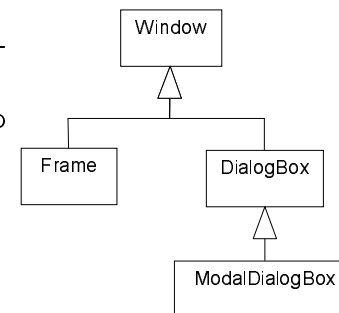


- mogą mieć nazwy, ale rzadko jest to konieczne
- stereotypy: *bind*, *derive*, *friend*, *instanceOf*, *instantiate*, *powertype*, *refine*, *use*

Relacje generalizacji

pokazują, że jedna rzecz (podklasa) jest rodzajem innej (nadklasa), co oznacza, że posiada (dziedziczy) wszystkie jej cechy i może być wszędzie zamiast niej używana

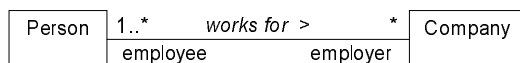
- pojedyncze i wielokrotne dziedziczenie
- mogą mieć nazwy, ale rzadko jest to konieczne
- stereotyp: *implementation*
- ograniczenia:
 - *complete/incomplete*
 - *disjoint/overlapping*



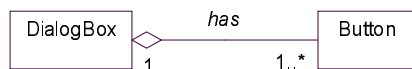
Powiązania

pokazują, że rzeczy (obiekty klas) wiążą się ze sobą, co oznacza, że od jednej rzeczy da się przejść do innej (wskazać inną) i vice versa

- dekoracje: nazwa, rola, krotność

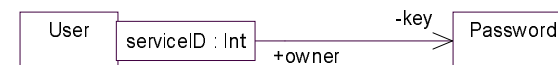


- dekoracje: agregacja i kompozycja

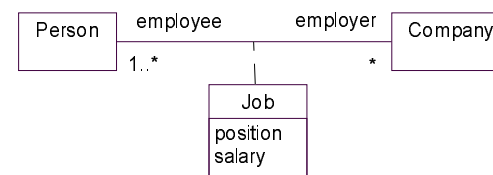


Inne właściwości powiązań

- dekoracje: nawigacja, dostęp, kwalifikacja



- dekoracje: klasa powiązania

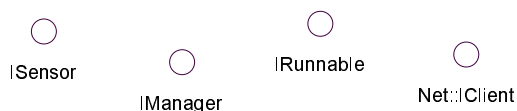


- ograniczenia:
 - implicit*, *ordered*, *changable*, *addOnly*, *frozen*, *xor*

Interfejsy

interfejs – zbiór operacji opisujących wymagane zachowanie

nazwy proste i kwalifikowane:



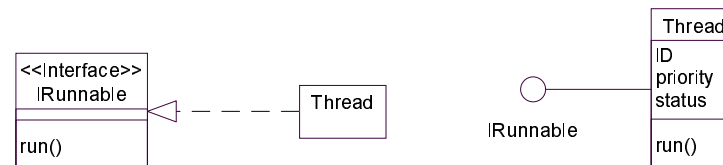
operacje (bez implementacji!):



Relacje realizacji

pokazują, że rzeczy (klasy) gwarantują pewne zachowania (interfejsy), czyli implementują zbiór operacji

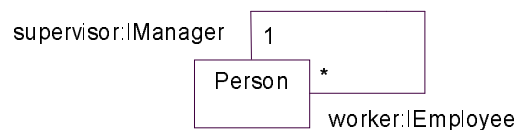
- postać podstawowa (rozwinięta) i uproszczona:



- klasa może realizować wiele interfejsów

Interfejsy a powiązania

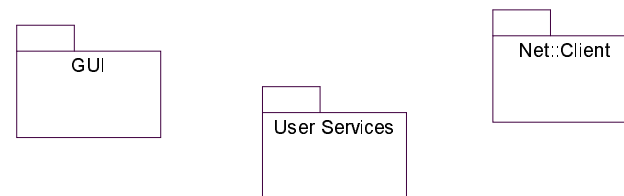
- w kontekście określonego powiązania klasa może prezentować jedynie wybrany interfejs
- dekoracje: specyfikacja interfejsu dla roli powiązania



Pakiety

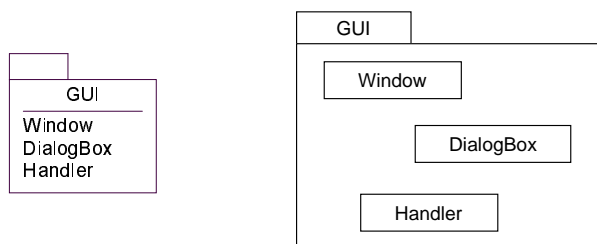
służą organizacji modelu przez łączenie elementów specyfikacji w grupy

nazwy proste i kwalifikowane:



Zawartość pakietu

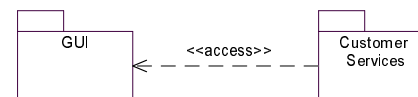
- specyfikacja tekstowa i graficzna:



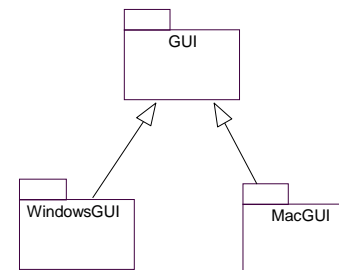
- ozdobniki: dostęp (eksport)
- stereotypy: *facade*, *framework*, *stub*, *subsystem*, *system*

Relacje między pakietami

- zależności (stereotypy: *import*, *access*)



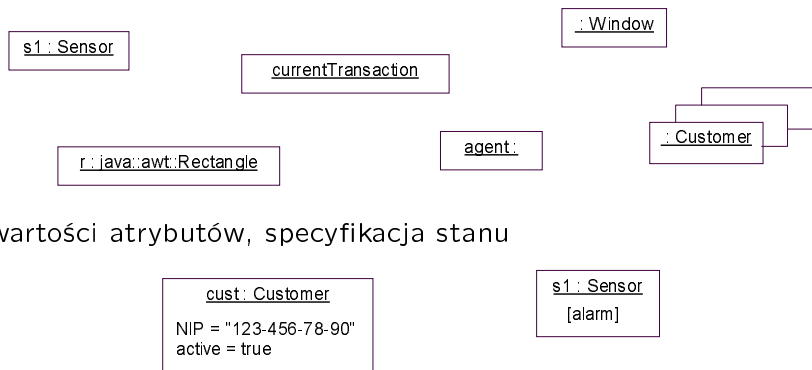
- generalizacje



Obiekty

obiekt — egzemplarz klasy

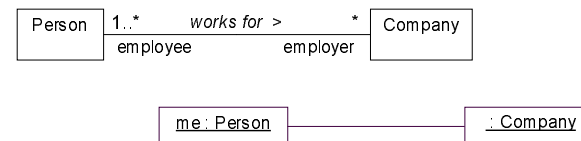
obiekty nazwane, anonimowe (wielokrotne), nieokreślonej klasy



wartości atrybutów, specyfikacja stanu

Połączenia

- realizacje powiązań



- wskazują drogi przesyłania komunikatów
- ozdobniki: takie jak dla powiązań (z wyłączeniem krotności)
- stereotypy: *association*, *self*, *global*, *local*, *parameter*
- ograniczenia: *new*, *destroyed*, *transient*

Komunikaty

- wywołania (także lokalne), powroty, sygnały
- oznaczanie wątków i kolejności wywołań
- sekwencje płaskie i zagnieżdżone (proceduralne)
- stereotypy: *create*, *destroy*