Analiza i projektowanie aplikacji Java

Modele analityczne a projektowe

- Modele analityczne (konceptualne) pokazują dziedzinę problemu.
- Modele projektowe (fizyczne) pokazują system informatyczny.
- Utrzymanie spójności – modele projektowe powinny być rozwinięciem (uszczegółowieniem, uściśleniem) modeli analitycznych.

Przegląd modelu klas

- Czy wszystkie klasy analityczne będą implementowane?
- Uzupełnienie brakujących klas
- Uzupełnienie definicji klas
- Stworzenie i uściślenie struktury klas

Stosowane środki (1)

- Generalizacja
  - wydzielenie i przenoszenie wspólnych składowych z kilku klas do osobnej, wspólnej klasy, z której te klasy będą dziedziczyć.
- Abstrakcja
  - tworzenie klas abstrahujących od pewnych szczegółów implementacji
  - stosowanie interfejsów – abstrakcja od wszystkich szczegółów implementacji (rozwiązanie problemu braku dziedziczenia wielokrotnego)
- Uszczegóławianie
  - tworzenie klas potomnych oferujących dodatkowe właściwości i możliwości
- Agregacja
  - deklarowanie klas kontenerowych, które będą zawierały klasy "biznesowe" wynikające z analizy wymagań
  - przenoszenie operacji z klas składowych do klas kontenerowych (komponent klasy kontenerowej "zna" wszystkie komponenty składowe)

Stosowanie środki (2)

- Partycjonowanie - podział systemu:
  - podział na warstwy - oszacowanie liczby kontraktów (wymienianych danych i wykorzystywanych operacji) między klasami.
    - stosowanie warstw ułatwia wprowadzanie modyfikacji do systemu (np. wymianę bazy danych) bez naruszania innych elementów architektonicznych (np. interfejsu użytkownika)
  - podział na przestrzenie nazw – organizacja hierarchiczna złożoności, możliwość stosowania tych samych nazw w różnych przestrzeniach.

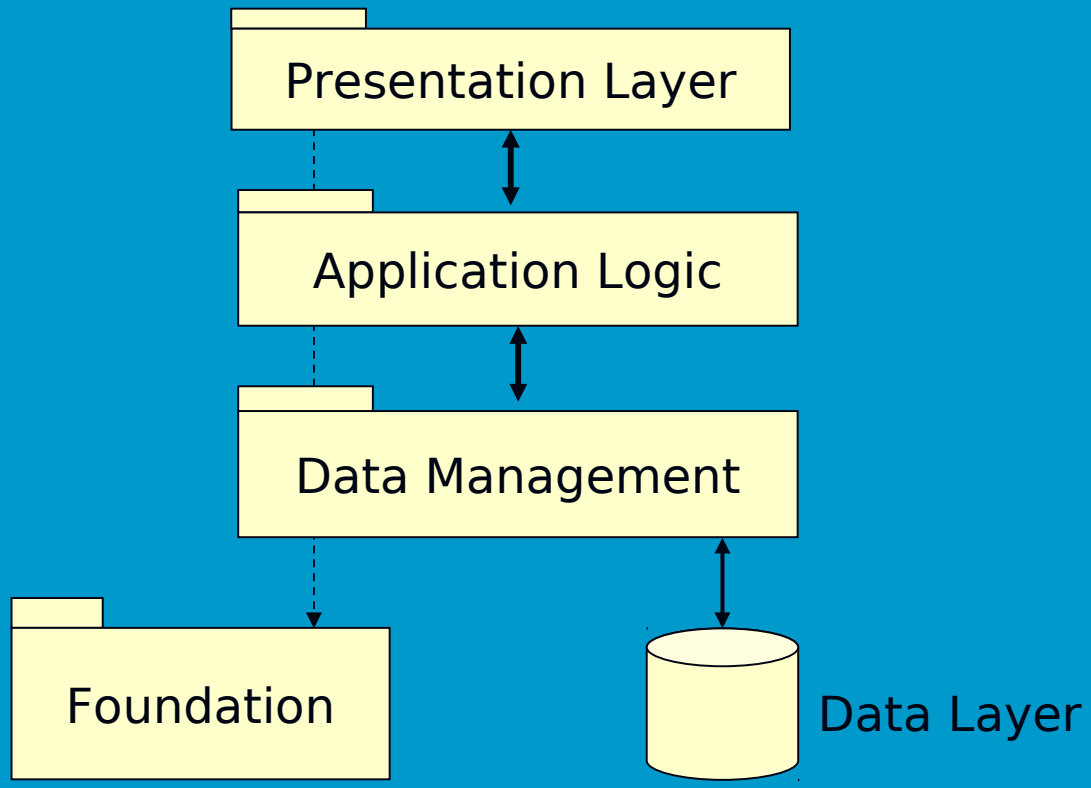
Stosowane środki (3)

- Wykorzystanie frameworków
  - Framework – gotowa biblioteka komponentów tworzących szkielet aplikacji i określających sposób jej działania.
  - Zaleta: ułatwia i przyspiesza projektowanie i implementację aplikacji.
  - Warunek: dobra znajomość wykorzystywanego frameworka.
  - Wady:
    - duża różnorodność frameworków przy braku standaryzacji
    - trudne lub niemożliwe wykorzystanie frameworka w sposób nieprzewidziany przez jego twórcę.

Stosowane środki (4)

- Faktoryzacja - wykorzystanie gotowych komponentów i szablonów projektowych z dopasowaniem do wymagań konkretnego projektu:
  - Zastosowanie kodu otwartego (open source).
  - Przeniesienie własnego kodu z innego projektu.
  - Adaptacja gotowych komponentów:
    - problemy: klasy zamknięte (sealed), właściwości i metody prywatne, brak wirtualizacji,
    - wrapper – klasa, której interfejs jest dopasowany do wymagań danego projektu, a implementacja korzysta w znacznym stopniu z innej, gotowej klasy (której nie można modyfikować).

Typowe warstwy projektowe



Warstwa podstawowa

- Warstwa podstawowa (*Foundation*) – obejmuje klasy wykorzystywane bezpośrednio we wszystkich innych warstwach:
  - definicje podstawowych typów danych (np. typy wyliczeniowe),
  - definicje podstawowych struktury danych (np. listy, drzewa, stosy),
  - użyteczne typy abstrakcyjne (data, czas, waluta)
  - operacje dodatkowe, które nie są dostarczane przez biblioteki

Warstwa danych

- Warstwa danych (*Data*) zawiera komponenty odpowiedzialne za przechowywanie (zapisywanie i odczytywanie) danych:
  - bazy danych (tabele, kwerendy)
  - repozytoria plików
- Wymaga określenia:
  - które klasy są trwałe (dane przechowywane między sesjami)
  - jaki sposób przechowywania będzie stosowany (baza danych, pliki)
  - jaki format zapisu danych będzie stosowany (tekstowy, binarny)
  - jaki typ bazy danych będzie stosowany (relacyjna, obiektowa)

Warstwa zarządzania danymi

- Warstwa zarządzania danymi (*Data Management*) zawiera klasy odpowiedzialne za dostęp do przechowywanych danych.
- Umożliwia:
  - ochronę danych przed nieupoważnionym dostępem
  - współdzielenie danych między wieloma użytkownikami

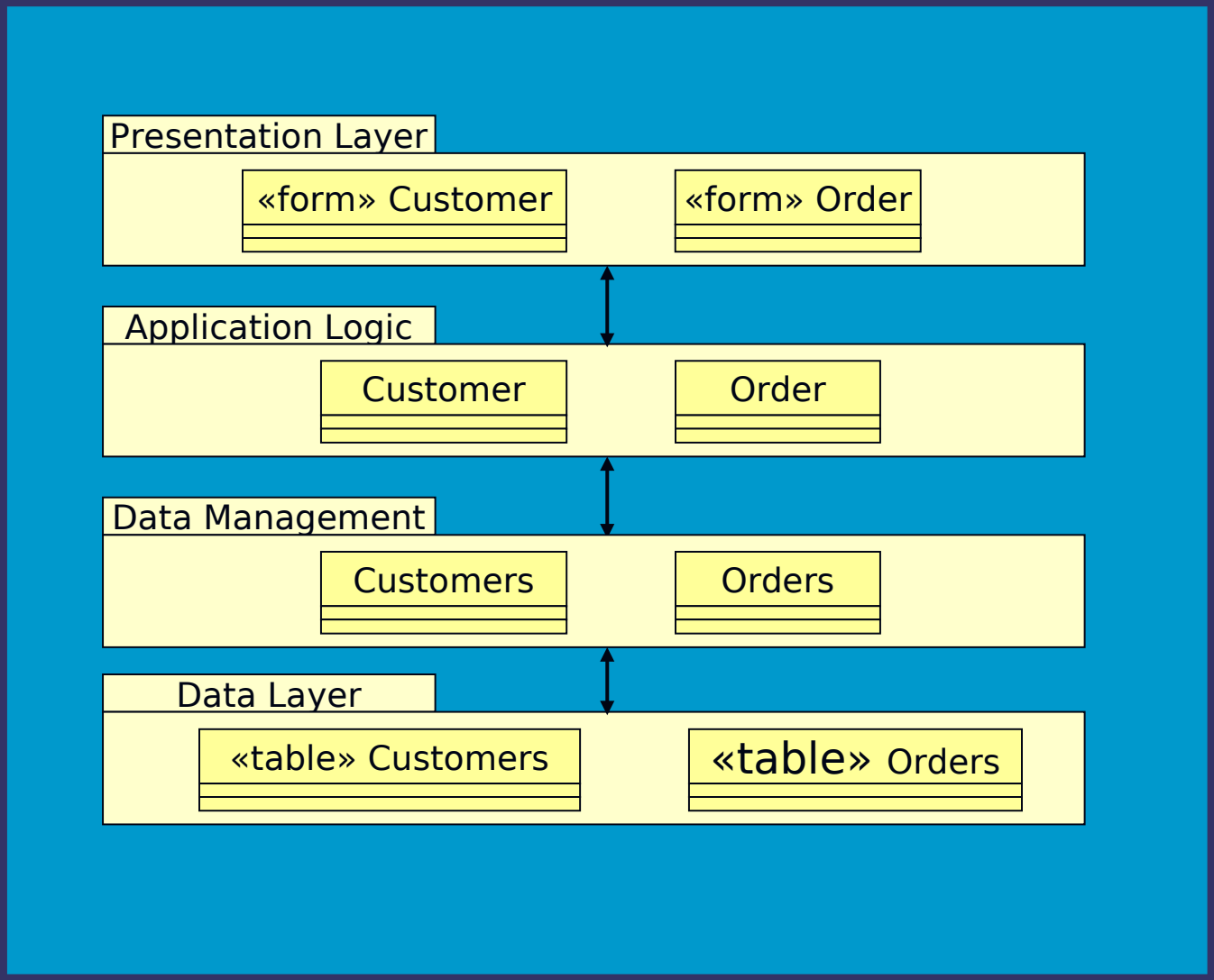
Warstwa logiki aplikacji

- Warstwa logiki aplikacji (*Application Logic*) – zwana również warstwą biznesową (*Business Domain*) – zawiera klasy realizujące operacje wymagane w konkretnej operacji wynikające z analizy wymagań.

Warstwa prezentacji

- Warstwa prezentacji (*Presentation Layer*) zawiera komponenty interfejsu użytkownika (okna, strony etc.):
  - umożliwia użytkownikowi wydawanie poleceń dla systemu i wprowadzanie danych
  - prezentuje dla użytkownika dane przetwarzane z warstwy logiki aplikacji

Przykładowa struktura klas z podziałem na warstwy



Literatura

- Dennis A., Wixom B.H., Tegarden D., *Systems Analysis & Design. An Object-Oriented Approach with UML*, John Wiley and Sons, USA, 2002

Pakiety w Java jako przestrzenie nazw

Definicja pakietu Java

- Pakiety służą organizacji modelu przez łączenie elementów specyfikacji w grupy.
- Pliki są grupowane w pakiety, które tworzą swoją integralną przestrzeń nazw.
- Klasy mogą mieć te same nazwy wewnątrz innych pakietów, a do pełnej identyfikacji używa się złożenia nazwy pakietu i nazwy własnej klasy
- Pakiet - zbiornik na klasy:
  - dzieli przestrzeń nazw na rozłączne zbiory
  - kontroluje widoczność klas i ich składowych

Deklaracja pakietu Java

- Instrukcję pakietu umieszczamy jako pierwszą komendę pliku źródłowego:
  package MojPakiet;
- Wszystkie klasy w pliku należą do pakietu *MojPakiet*.
- Inne pliki mogą posiadać tą samą instrukcję: pakiet rozkłada się na wiele plików.
- Wszystkie pliki *.class* w pakiecie *MojPakiet* są zapisywane w katalogu *MojPakiet*.
- Pakiet wielo-poziomowy:
  package pakiet1.pakiet11.pakiet111;

Importowanie pakietów

Dodatkowe klasy/pakiety, które będą potrzebne w kodzie programu importujemy słowem kluczowym ***import***:

import java.util.;*

- *java.lang* - pakiet automatycznie importowany do każdego pliku Java.
  - Pakiet java.lang zawiera klasę System
  - Klasa System zawiera pole: static PrintStream out;
  - Klasa PrintStream zawiera metodę: void println(String x)

Konflikty nazw

- Jeśli klasa o tej samej nazwie występuje w dwóch różnych pakietach importowanych do programu:

  import pakiet1.;*

  import pakiet2.;*
- kompilator wyświetli komunikat błędu gdy spróbujemy użyć jednej z klas.
- Należy wówczas użyć pełnej nazwy:
  - pakiet1.Klasa
  - pakiet2.Klasa