# Programowanie obiektowe

## Wykład 13

Marcin Młotkowski

25 maja 2017

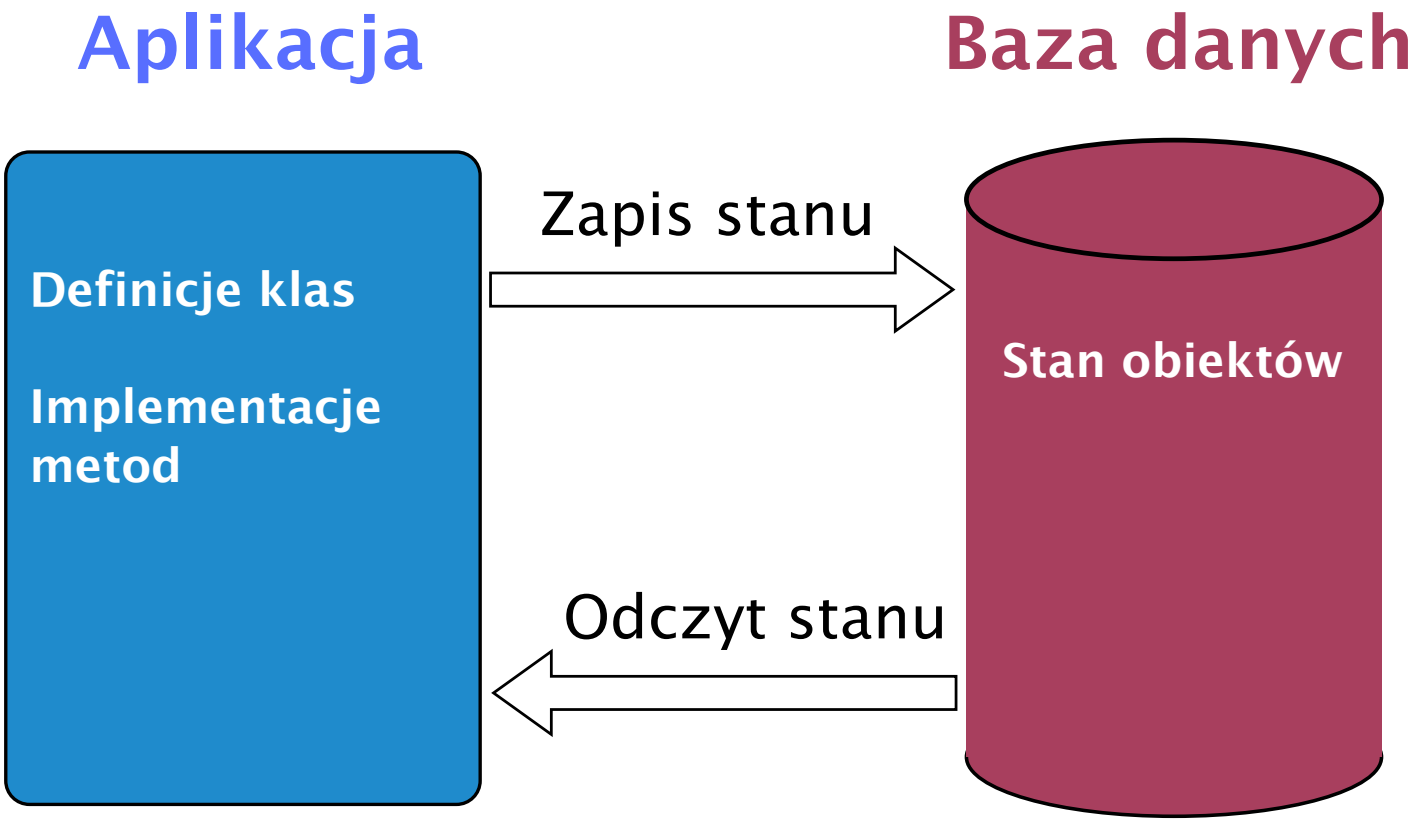
# Plan wykładu

# Trwałość (persistence)

**Definicja**

Cecha danej, zmiennej lub obiektu oznaczająca zachowanie jej wartości dłużej niż czas pojedynczego uruchomienia programu.

# Przechowywanie stanu w BD



# Scenariusz użycia trwałego obiektu

- Odczyt obiektu (stanu obiektu z bd)
- Modyfikacja stanu
- Wywołanie metod obiektu
- Zapis

# Zagadnienia

- Zwykle przechowywany jest stan obiektu a nie obiekt (tj. klasa, nadklasa, metody itp)
- Gorliwe czy leniwe aktualizacje (odczyt/zapis) stanu
- Mechanizm aktualizacji zmian (dodawanie nowych pól, zmiana typów pól w nadklasie etc)
- Pytanie: jaka część stanu powinna być przechowywana?

# Naturalne podejście

- Połączenie aplikacji obiektowej z istniejącym systemem baz danych
- Rozszerzenie środowiska obiektowego o możliwości przetwarzania dużych danych
- Zbudowanie całego środowiska od początku
- Rozszerzenie systemu BD o właściwości obiektowe

# Łączenie środowiska obiektowego z systemem baz danych

- Środowiska obiektowe: JAVA, C#, C++, Python ...
- Systemy baz danych: MySQL, Oracle, PostgreSQL, Sybase, Microsoft SQL Server ...

# Strategia implementacji

- Odwzorowanie cech obiektowych w relacyjnych bazach danych (object-relational mapping, ORM)
- Implementacja warstwy pośredniej

# Porównanie modelu relacyjnego z obiekowym

## Model obiektowy

- Ukrywanie danych i dostęp tylko przez wskazany interfejs
- Relacje między obiektami: asocjacje, dziedziczenie, kompozycja
- Tożsamość obiektów

## Model relacyjny

- Rekord
- Tabela: kolekcja rekordów tego samego typu
- Język zapytań

# Java Data Object

- definiuje specyfikację;
- obiekty mogą być pamiętane w plikach, relacyjnych i obiektowych bazach danych;
- szczegóły (gorliwość, leniwość, etc) są definowane w zewnętrznych plikach xml;
- przykładowa realizacja: Apache JDO, DataNucleus

# Java Persistence API

- specyfikacja;
- głównie do przechowywania stanu obiektów w relacyjnych BD;
- szczegóły są definiowane za pomocą annotacji lub w zewnętrznych plikach;
- Java Persistence Query Language: język zapytań przypominający SQL;
- implementacje: Hibernate, TopLink.

# C#: ADO.NET

- ADO - ActiveX Data Objects
- ADO.NET – środowisko dostępu do danych w BD

# Entity Framework

- Część środowiska ADO.NET
- Powstanie: 2008 rok
- Projekt danych w postaci konceptualnego modelu danych (Entity Data Model)
- Automatyczne odwzorowanie na RBD i model obiektowy
- Zapisywanie modelu danych w pliku XML

# LINQ: Language Integrated Query

- Projekt Microsoftu
- Dostęp na poziomie języka programowania do danych w BD za pomocą składni SQL-podobnej

# Przykład LINQ

```
Northwind db = new Northwind(connectionString);
var q = from o in db.Orders, c in db.Customers
    where o.Quality == "200"
    select new { o.DueDate,c.ItemID};
foreach (var t in q) { ... }
```

# Rozszerzenia RBD na przykładzie Oracle

- PL/SQL – rozszerzenie SQL o procedury
- Składnia zapożyczona z Ady
- Możliwość definiowania własnych programów wykonywanych na serwerze
- Możliwość deklarowania własnych typów danych (klas)
- Rozszerzenie SQL o odwołania do obiektów

# Serwer STONE

- Przechowuje obiekty
- Obiekty nie są kopiowane do aplikacji klienta, tylko zostają w serwerze
- Metody są wykonywane na serwerze

# Rozwiązanie w Smalltalku

Przechowywanie stanu aplikacji w jednym pliku (obrazie).

## Wady rozwiązania

- Kłopoty z przetwarzaniem dużych porcji danych
- Kłopoty z szybkim wyszukiwaniem danych
- Problem wielodostępu do danych

# Plan wykładu

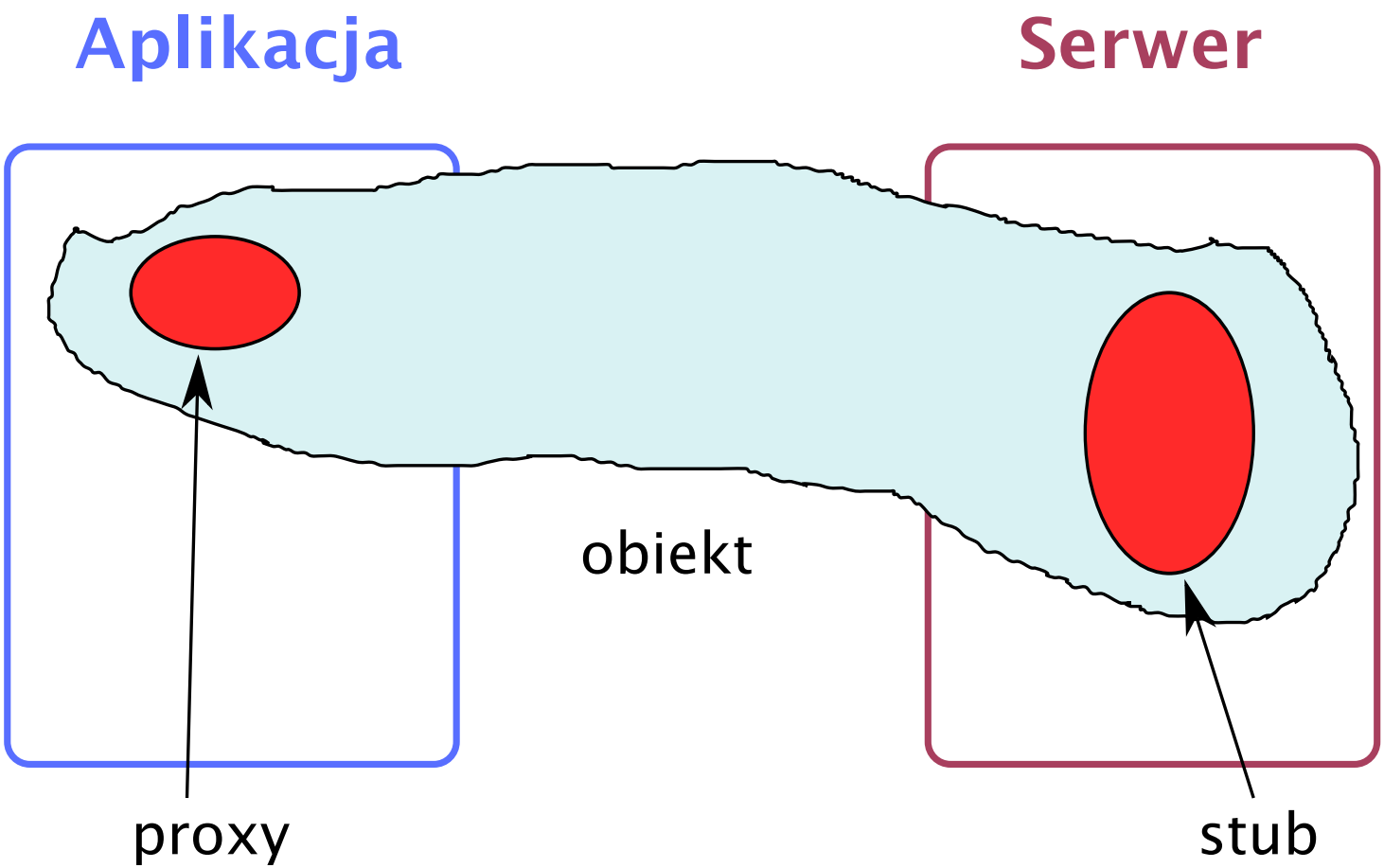
# Obiekty trwałe: inna koncepcja

- Obiekt (jego stan) oraz implementacja metod jest pamiętana na serwerze
- Wykonanie metody powoduje odwołanie się do zdalnego obiektu, wykonanie metody na serwerze i zwrócenie klientowi wyniku
- Serwer dba o spójność danych, wielodostęp, transakcyjność etc

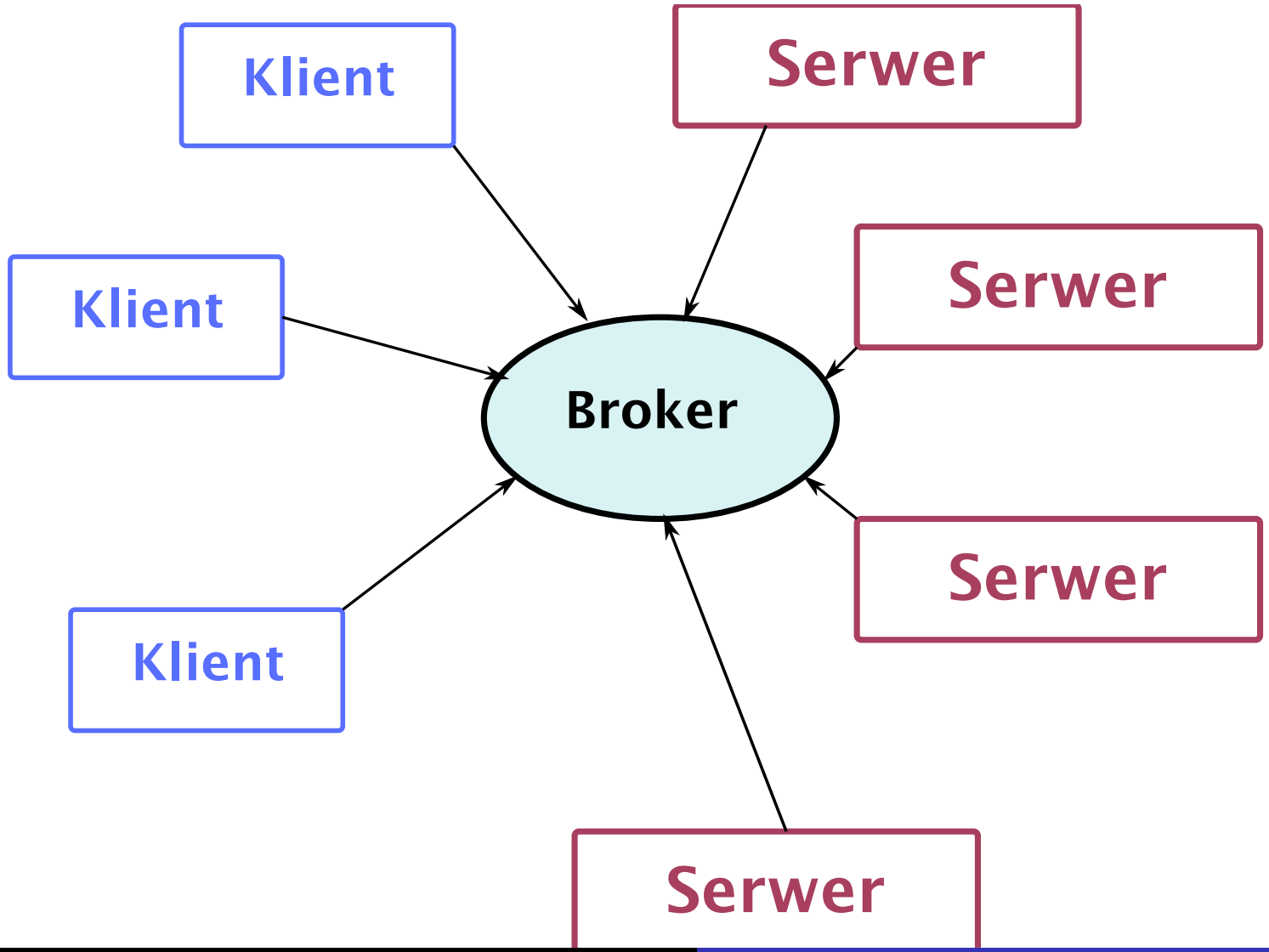
# Postulaty dotyczące takiego środowiska

- Jednolity dostęp do obiektów lokalnych i odległych
- Łatwość tworzenia odwołań do odległych obiektów
- Niezależność standardów od platformy

# Architektura



# Broker



# Środowiska obiektów rozproszonych

- Java RMI
- CORBA
- SOAP
- DCOM

# Java RMI

- RMI – Remote Method Invocation
- Współpraca tylko między aplikacjami napisanymi w Javie
- java.rmi.*
- Całe środowisko jest częścią SDK

# CORBA

- Zbiór standardów opracowanych przez OMG (Object Management Group)
- Niezależność od platformy/sprzętu/języka
- Implementacje: VisiBroker, ORBit
- Bardzo obszerna specyfikacja

# SOAP

- Simple Object Access Protocol
- Oparty na XML
- Standard W3C
- Implementacje: .NET Remoting, Apache SOAP
- .NET Remoting: również realizacja binarna

# DCOM

- Distributed Component Object Model
- Implementacja tylko na systemy Microsoftu
- Uznany za przestarzały na rzecz .NET Remoting