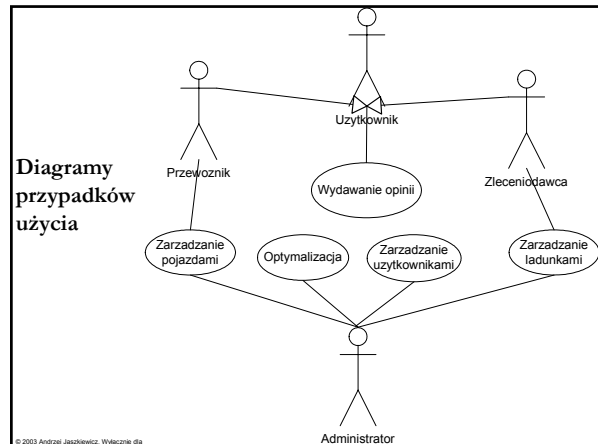


UML a kod w C++ i Javie

- Projektowanie oprogramowania
- Dokumentowanie oprogramowania

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka



© 2003 Andrzej Jaszkiewicz. Wyłączone dla

Klasy użytkowników i wykorzystywane funkcje

- Mogą sugerować podział systemu na odrębne aplikacje, np.
 - aplikacja dla użytkowników systemu
 - aplikacja dla administratora
- Elementy interfejsu użytkownika, np.
 - różne tryby pracy, np. różne systemy menu
 - oddzielne fragmenty w strukturze menu
 - blokowanie dostępu do pewnych funkcji

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Przypadki użycia

- Funkcje systemu
- Elementy interfejsu użytkownika
 - menu, podmenu, polecenia w menu
 - dialogi

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Związki pomiędzy przypadkami użycia

- Struktura menu
- Polecenia dostępne w dialogach, np. wywoływanie innych dialogów
- Kreatory (creators, wizards)

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Diagramy klas

- Bezpośrednie przełożenie na kod
- Wiele dodatkowych elementów wykorzystywanych na etapie projektowania

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Klasa

Pojazd

```
class Pojazd {
    ...
}
```

Nazwy - prefixy, suffixy,
zamiana spacji i
nie dozwolonych znaków

Student dzienny

```
class CStudentDzienny
{
    ...
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Pola C++

```
class CPojazd {
    ...
    ... Nazwa;
    ... CenaKilometra;
    ... CenaGodziny;
}
```

Pojazd
Nazwa
Cena kilometra
Cena godziny

Na przykład:

```
class CPojazd {
protected:
    char* Nazwa;
    double CenaKilometra;
    double CenaGodziny;
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Pola Java

```
class Pojazd {
    ... nazwa;
    ... cenaKilometra;
    ... cenaGodziny;
}
```

Pojazd
Nazwa
Cena kilometra
Cena godziny

Na przykład:

```
class CPojazd {
    protected String nazwa;
    protected double cenaKilometra;
    protected double cenaGodziny;
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Symbole widoczności pól i operacji

+ public – publiczne
protected – zabezpieczone/chronione
- private – prywatne

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Pojazd

+Nazwa
#Cena kilometra
-Cena godziny

```
class CPojazd {
public:
    ... Nazwa;
protected:
    ... CenaKilometra;
private:
    ... CenaGodziny;
}
```

```
class Pojazd {
    public ... nazwa;
    protected ... cenaKilometra;
    private ... cenaGodziny;
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Typy pól

Pojazd

+Nazwa : char*
#Cena kilometra : double
-Cena godziny : double

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Operacje i metody C++

Pojazd
+Koszt()

```
class CPojazd {
...
public:
... Koszt (...);
};
...
... CPojazd::Koszt (...) {
...
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Operacje i metody Java

Pojazd
+Koszt()

```
class Pojazd {
...
public ... koszt (...) {
...
}
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Nagłówki operacji C++

Pojazd
+Koszt(in Czas : double, in Droga : double): double

```
class CPojazd {
...
public:
double Koszt (double Czas, double Droga);
};
...
double CPojazd::Koszt (double Czas, double
Droga) {
...
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

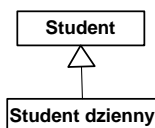
Nagłówki operacji Java

Pojazd
+Koszt(in Czas : double, in Droga : double): double

```
class Pojazd {
...
public double koszt (double czas, double droga) {
...
}
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Generalizacja-specjalizacja - dziedziczenie

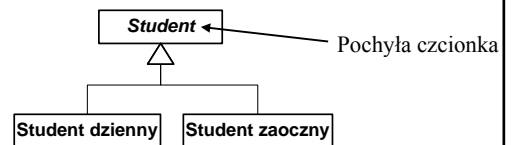


```
class CStudentDzienny :
public CStudent {
...
}
```

```
class StudentDzienny
extends Student {
...
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Klasy abstrakcyjne



Klasy nie posiadające obiektów (bezpośrednio tej klasy)

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Klasy abstrakcyjne C++

- Brak tworzenia obiektów tej klasy w kodzie
- Operacje abstrakcyjne – muszą być zdefiniowane w każdej ze specjalizacji, której obiekty będą tworzone

```
class CStudent {
    ...
    virtual CGrupa* PodajGrupe () = 0;
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Klasy abstrakcyjne Java

```
abstract class Student {
    ...
}
albo
abstract class Student {
    ...
    abstract CGrupa podajGrupe ();
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Interfejsy (interfaces)

- Zbiór operacji (deklaracji metod)
 - *Przypomina klasę zawierającą wyłącznie operacje abstrakcyjne*
 - *Może zawierać stale*

```
«interface»
Obiekt graficzny
+Rysuj()
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Interfejsy w C++

- Nie wspierane?

```
class CObiektGraficzny {
public:
    virtual void Rysuj () = 0;
}
```

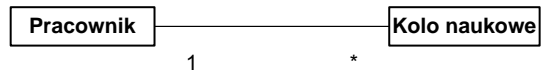
© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Interfejsy w Javie

```
interface IObiektGraficzny {
    void rysuj ();
}
Implementacja interfejsu
class Rysunek implements IObiektGraficzny
{
    ...
    public void rysuj ();
}
```

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Związki klas



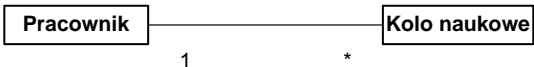
- Ogólnie dowolny sposób pozwalający na przechowanie informacji o powiązanych obiektach

- Np. tablica zawierająca pary powiązanych obiektów

Kolo naukowe	Pracownik
K. Informatyki	Nowak
K. Fizyki	Kamiński
K. Chemii	Zieliński

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunku Informatyka

Związki klas



- Najczęściej dodatkowe pola przechowujące informacje o powiązanych obiektach
 - Każdy obiekt klasy Pracownik będzie przechowywał informacje o powiązanym obiekcie (dokładnie jednym) klasy Kolo naukowe
 - Każdy obiekt klasy Kolo naukowe będzie przechowywał informacje o powiązanych obiektach (dowolnej liczbie) klasy Pracownik

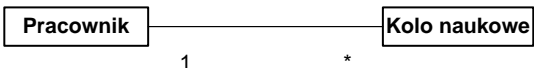
© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Sposób przechowywania informacji o powiązanych obiektach

- Identyfikatory (np. nazwy)
- Wskaźniki/referencje

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Związki w C++



- Najczęściej wskaźniki

```

class CPracownik {
...
protected:
    vector <CKoloNaukowe*> rKoloNaukowe;
}

class CKoloNaukowe {
...
protected:
    CPracownik* rPracownik;
}
    
```

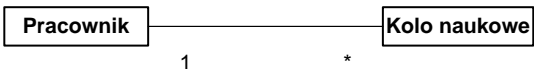
© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Związki w C++

- Krotność 1
 - Wskaźnik, który musi wskazywać na powiązany obiekt
- Krotność 0..1
 - Wskaźnik, który może mieć wartość NULL
- Krotność *, 1..*
 - Klasa vector (biblioteka STL) – dla 1..* nie może być pusty
 - Tablica wskaźników
 - Inna struktura danych

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Związki w Javie



- Najczęściej referencje i ich kolekcje

```

class Pracownik {
...
protected Vector rKoloNaukowe;
// lub
protected KoloNaukowe[] rKoloNaukowe;
}

class KoloNaukowe {
...
protected Pracownik pracownik;
}
    
```

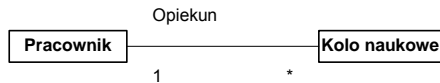
© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Związki w Javie

- Krotność 1
 - Referencja, która musi wskazywać na powiązany obiekt
- Krotność 0..1
 - Referencja, która może mieć wartość NULL
- Krotność *, 1..*
 - Obiekt klasy z biblioteki standardowych struktur danych Javy – dla 1..* nie może być pusty
 - Tablica referencji
 - Inna struktura danych

© 2003 Andrzej Jaszkiewicz. Wyłączone dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Wykorzystanie nazw ról w związkach



```

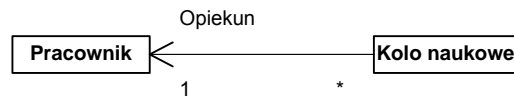
class CKoloNaukowe {
...
protected:
    CPracownik* rOpiekun;
}
    
```

```

class KoloNaukowe {
...
protected Pracownik opiekun;
}
    
```

© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Związki skierowane



```

class CPracownik {
...
}
    
```

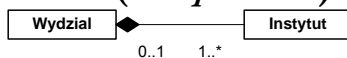
```

class CKoloNaukowe {
...
protected:
    CPracownik* rPracownik;
}
    
```

Brak informacji o powiązanych obiektach klasy Kolo naukowe

© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

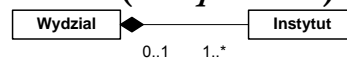
Związek kompozycji (*composition*)



- W zasadzie na poziomie implementacji nierozróżnialne od związków zwykłych
- Często obiekt będący całością jest odpowiedzialny za przechowywanie swoich składowych (dodawanie, usuwanie)

© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Związek kompozycji (*composition*)



- W C++ czasami wykorzystanie obiektów zamiast wskaźników

```

class CWydzial {
...
protected:
    vector <CInstytut> rInstytut;
}
    
```

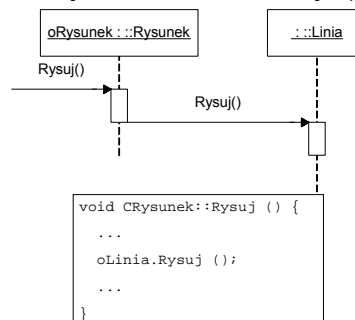
© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Diagramy sekwencji

- Wywoływanie metod w programie
- Podstawa implementacji metod

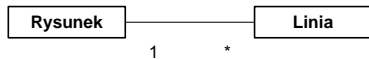
© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Wywołanie metody (*call*)



© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Dla powiązanych obiektów w C++



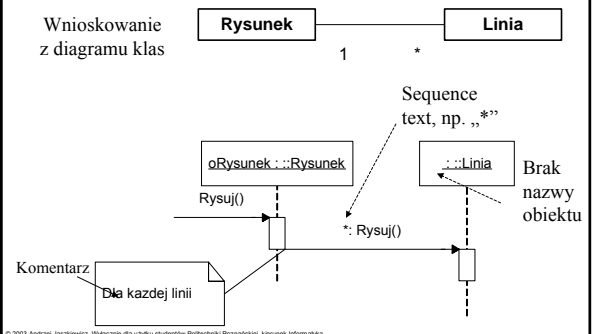
```

void CRysunek::Rysuj () {
    ...
    rLinia->Rysuj ();
    ...
}
  
```

© 2003 Andrzej Jasiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

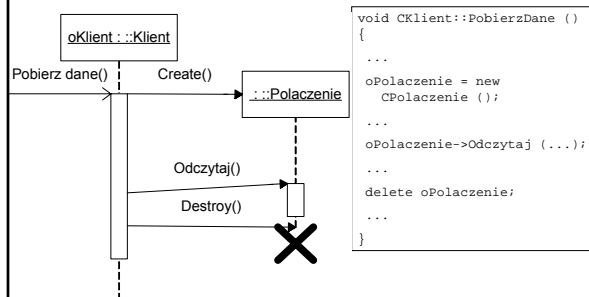
Czy wywołanie w pętli?

Wnioskowanie z diagramu klas

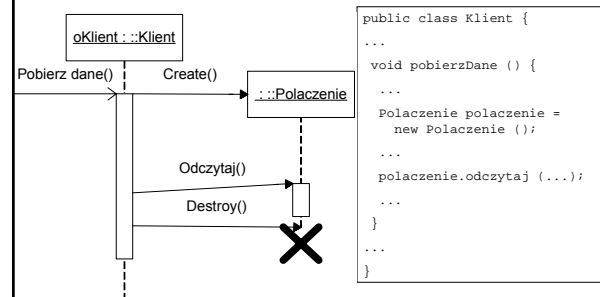


© 2003 Andrzej Jasiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Tworzenie i usuwanie obiektów w C++



Tworzenie i usuwanie obiektów w Javie



Dostęp do pól

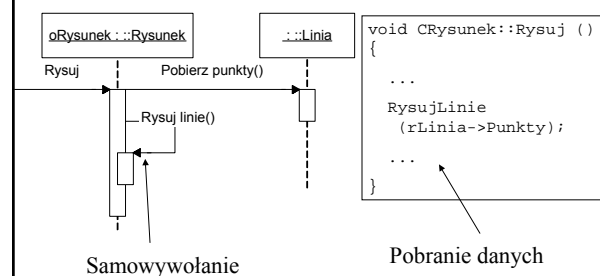
Operacje:

- Pobierz dane / Get data
- Ustaw dane / Set data
- Pobierz pole / Get field
- Ustaw pole / Set field

Mogą być implementowane jako odczyt/zapis pól

© 2003 Andrzej Jasiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

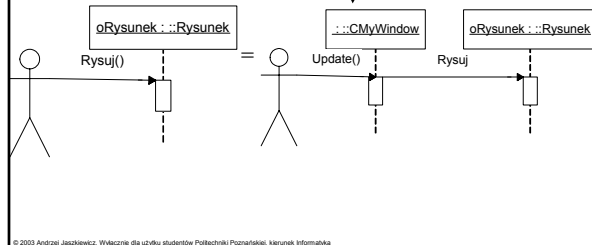
Dostęp do pól i samowywołanie



© 2003 Andrzej Jasiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

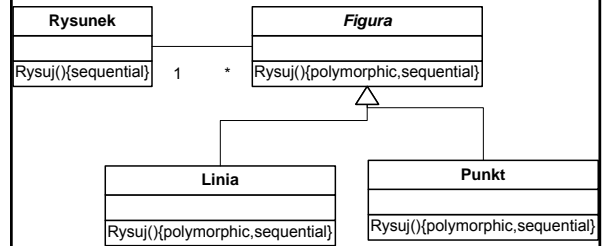
Wywołania pochodzące z zewnątrz

Klasa interfejsowa



© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Operacje wirtualne (polimorficzne)



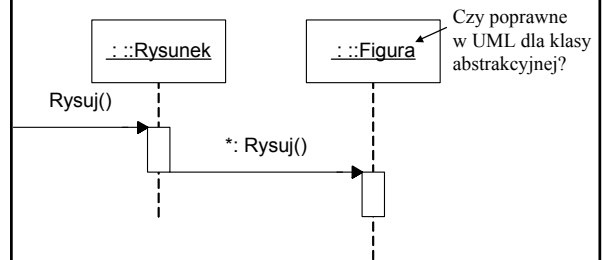
© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Operacje wirtualne

- W Javie domyślne
- W C++
 - `virtual void Rysuj ();`

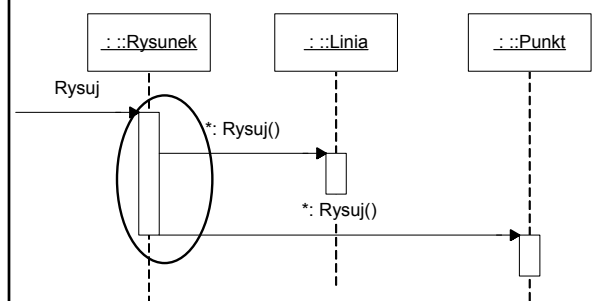
© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Punkt widzenia klasy Rysunek



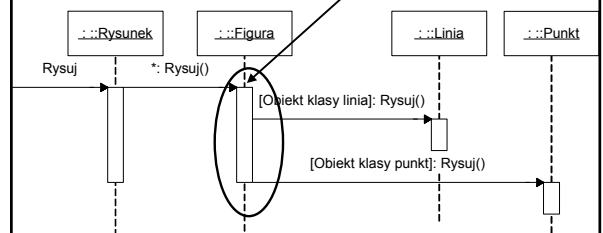
© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Rzeczywiste wywołania metod dla obiektów



Ilustracja efektu operacji wirtualnej

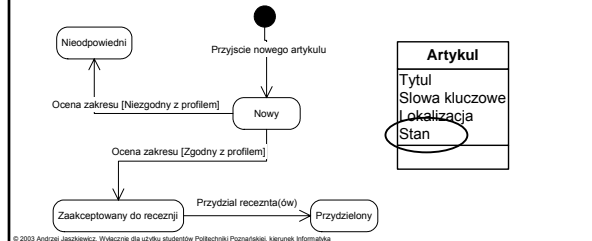
W rzeczywistości ta metoda nie istnieje



© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Diagramy stanów

- Realizacja klasyczna – pole przechowujące stan i odpowiednia reakcja metod



Zmiany stanów w metodach

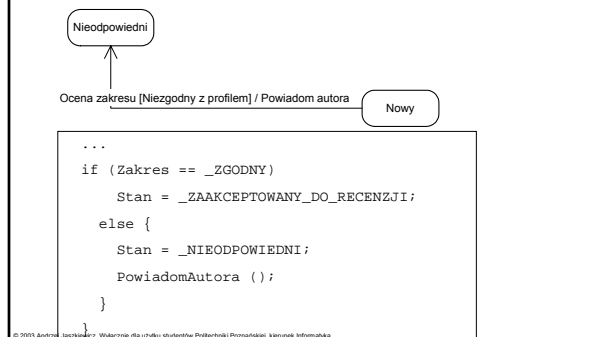
```

void CArtykul::OcenaZakresu (TZakres
Zakres) {
    if (Stan == _NOWY) {
        if (Zakres == _ZGODNY)
            Stan = _ZAAKCEPTOWANY_DO_RECENZJI;
        else
            Stan = _NIEODPOWIEDNI;
    }
    else
        // Niepoprawne
        ...
}

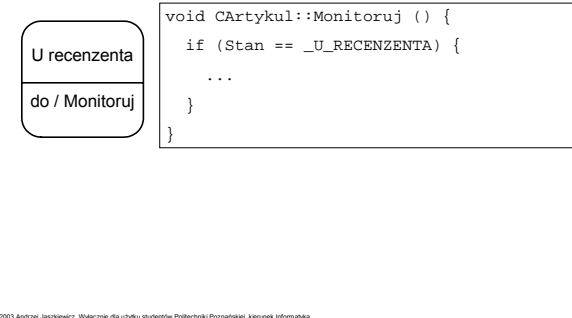
```

© 2003 Andrzej Jaszkiewicz. Wyłączenie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Akcje/operacje -> metody (fragmenty metod)

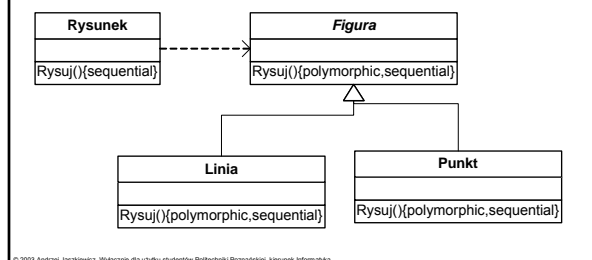


Akcje/operacje -> metody (fragmenty metod)



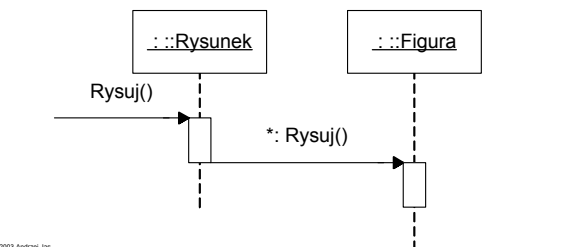
Związek zależności (dependency)

- Zmiana jednego elementu może powodować zmianę drugiego



Związek zależności (dependency)

- Często oznacza uogólnienie (być może wielu) wywołań metod i dostępów do pól



Pakiety

```

classDiagram
    class Klient
    class Typ_pojazdu["Typ pojazdu"]
    class Pojazd
    class Wolny_pojazdu["Wolny pojazd"]
    class Rozwiazanie
    class Trasa
    class Populacja

    Klient "1" -- "*" Typ_pojazdu
    Typ_pojazdu "1" -- "*" Pojazd
    Pojazd "1" -- "*" Wolny_pojazdu
    Rozwiazanie "1" -- "*" Trasa
    Rozwiazanie "1..*" -- "1" Populacja
    Typ_pojazdu ..> Rozwiazanie
  
```

© 2003 Andrzej Jacekiewicz. Wyłączenie dla studentów Politechniki Poznańskiej, kierunek Informatyka

© 2003 Andrzej Jaszkiewicz. Wyłącznie dla użytku studentów Politechniki Poznańskiej, kierunek Informatyka

Diagramy wdrożenia – deployment diagrams

The diagram illustrates the deployment architecture of a system. It features several nodes (machines) and components (modules) distributed across them. The nodes are: Broker server, Client machine, Modelling tool server, Solver server, and Solver server (Distributed solver). The components are: Broker, Interactive analysis module, Modelling tool, Solver, and Distributed solver. The diagram shows the following relationships: Broker server contains Broker component. Client machine contains Interactive analysis module component. Modelling tool server contains Modelling tool component. Solver server contains Solver component. Solver server (Distributed solver) contains Distributed solver component. The Broker component is connected to the Interactive analysis module component via a dashed line labeled 'OC transport interface'. The Modelling tool component is connected to the Solver component via a dashed line labeled 'Modelling tool interface'. The Solver component is connected to the Distributed solver component via a dashed line labeled 'Solver interface'. The Solver component is also connected to the Distributed solver component via a solid line labeled '1'.

Wzrost (klasa węzłów)

Component (klasa komponentów)

Interfejs