Programowanie Obiektowe
Wprowadzenie

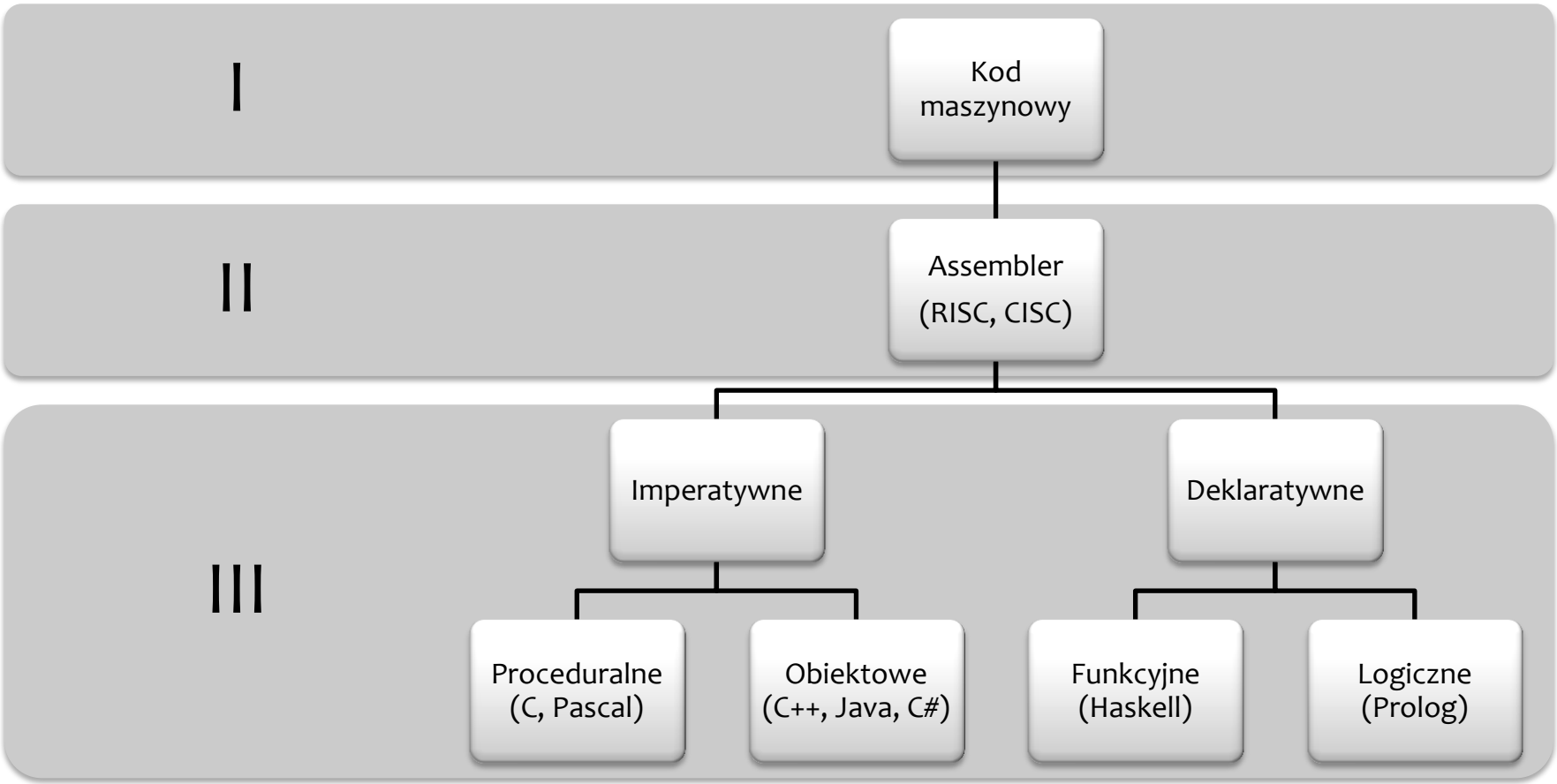
Dariusz Brzeziński

Politechnika Poznańska, Instytut Informatyki

Programowanie obiektowe

- Object-oriented programming
- Najpopularniejszy obecnie styl (paradygmat) programowania
- Rozwinięcie koncepcji programowania strukturalnego
  - Podejście tradycyjne (proceduralne): program jako kolekcja funkcji (a wcześniej lista instrukcji)
  - Podejście obiektowe: program jako kolekcja współpracujących ze sobą obiektów
- Obiekty łączą dane i operacje na nich
- Zalety programowania obiektowego:
  - Ułatwia współpracę i podział zadań między programistów
  - Ułatwia pielęgnację i rozbudowę aplikacji
  - Ułatwia ponowne wykorzystywanie wcześniej napisanego kodu
  - Często umożliwia naturalne modelowanie rzeczywistości
  - Odpowiednie dla dużych projektów, popularne w inżynierii oprogramowania

Rozwój paradygmatów programowania



Programowanie strukturalne a programowanie obiektowe

Podejście strukturalne (C++)

```
struct Punkt
{
  int x, y;
};

void narysuj (struct Punkt P)
{
  // ciało funkcji
}
```

Podejście obiektowe (C++)

```
class Punkt
{
  int x, y;
public:
  void narysuj ()
  {
    // ciało funkcji
  }
};
```

Podstawowe pojęcia (1/5)

- **Klasa** – definiuje charakterystykę „czegoś”; wyznacza modularną strukturę programu zorientowanego obiektowo
  - Charakterystyka obejmuje atrybuty (pola, właściwości) i zachowanie (metody)
  - Pola i metody klasy są określane jako **składowe klasy**
  - Przykład: klasa Punkt opisująca cechy i zachowanie wspólne dla wszystkich punktów np. współrzędne
- **Obiekt** – instancja (wystąpienie) klasy
  - Np. konkretny Punkt – punkt o współrzędnych (3,4)
  - Zbiór wartości atrybutów obiektu w danej chwili jest określany mianem **stanu obiektu**

Podstawowe pojęcia (2/5)

- **Metoda** – operacja, która może być wykonana na obiekcie, reprezentująca jego funkcjonalność
  - Np. metoda narysuj() klasy Punkt, która może być wywołana na rzecz konkretnego obiektu klasy Punkt

- **Przekazywanie komunikatów** – proces polegający na przekazaniu danych z obiektu do obiektu lub zleceniu wywołania metody na rzecz obiektu

Podstawowe pojęcia (3/5)

- **Hermetyzacja** – ukrycie szczegółów implementacji klasy przed kodem korzystającym z klasy
  - Składowe klasy dostępne z zewnątrz stanowią interfejs klasy
  - Składowe niedostępne z zewnątrz mogą być zmieniane bez wpływu na pozostały kod aplikacji
  - Realizowane poprzez kwalifikatory dostępu do składowych klasy
  - Podstawowe kwalifikatory dostępu do składowych:
    - **public** – dostęp publiczny
    - **protected** – dostępne w klasie definiowanej i klasach pochodnych
    - **private** – dostępne tylko w definiowanej klasie

Podstawowe pojęcia (4/5)

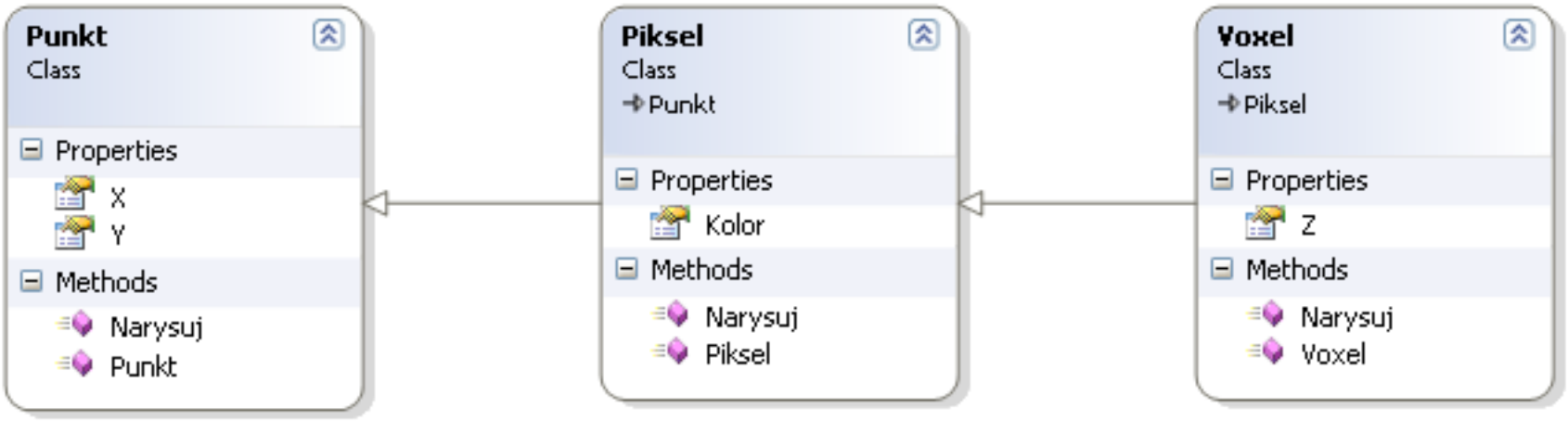
- **Dziedziczenie** – definiowanie **podklasy** (**klasy pochodnej**) jako **specjalizacji** klasy istniejącej (**klasy bazowej**, **nadklasy**)
  - Np. klasa Piksel jako specjalizacja klasy Punkt
  - Podklasa dziedziczy z klasy bazowej atrybuty i metody
  - Podklasa może posiadać dodatkowe atrybuty i metody oraz redefiniować metody odziedziczone

- **Dziedziczenie wielobazowe** – polega na definiowaniu klasy jako dziedziczącej bezpośrednio z więcej niż jednej klasy
  - Dostępne nie we wszystkich językach
  - Prowadzi do skomplikowanych programów

Podstawowe pojęcia (5/5)

- **Abstrakcja** – praca z obiektami na poziomie ogólności (względem hierarchii dziedziczenia) odpowiednim dla rozwiązywanego problemu
  - Np. traktowanie w danym kontekście instancji klasy Piksel jako instancji klasy Punkt, jeśli nie są w nim wykorzystywane atrybuty i metody niewystępujące w klasie Punkt
  - Umożliwia np. traktowanie kolekcji wystąpień konkretnych podklas klasy Punkt ogólnie – jako punktów
- **Polimorfizm** – różne zachowanie w odpowiedzi na takie samo wywołanie metody w zależności od konkretnej klasy obiektu
  - W połączeniu z dziedziczeniem i abstrakcją
  - **Późne wiązanie** – decyzja o tym, z której klasy metodę wywołać podejmowana w trakcie działania programu, a nie na etapie kompilacji

Diagramy klas



C++

- zaprojektowany jako rozszerzenie języka C o obiektowe mechanizmy abstrakcji danych
- jest to język pozwalający na programowanie zarówno proceduralne jak i obiektowe
- C++ posiada bardzo rozbudowaną składnię
- żaden popularny kompilator nie jest w pełni zgodny z obowiązującym standardem języka, choć to się poprawia
- posiada rzadkie w innych językach obiektowych: dziedziczenie wielobazowe, unie, bezpośrednie zarządzanie wolną pamięcią, operacje arytmetyczne na wskaźnikach