Programowanie obiektowe (TZ1E2010)

Politechnika Białostocka - Wydział Elektryczny

Elektronika i telekomunikacja, semestr II
studia niestacjonarne I stopnia

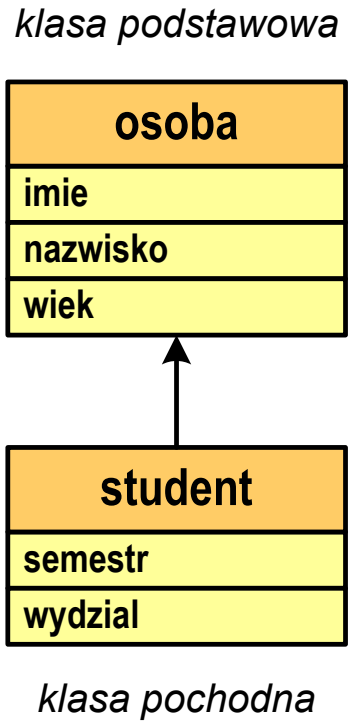
Rok akademicki 2020/2021

Pracownia nr 6 (23.04.2021)

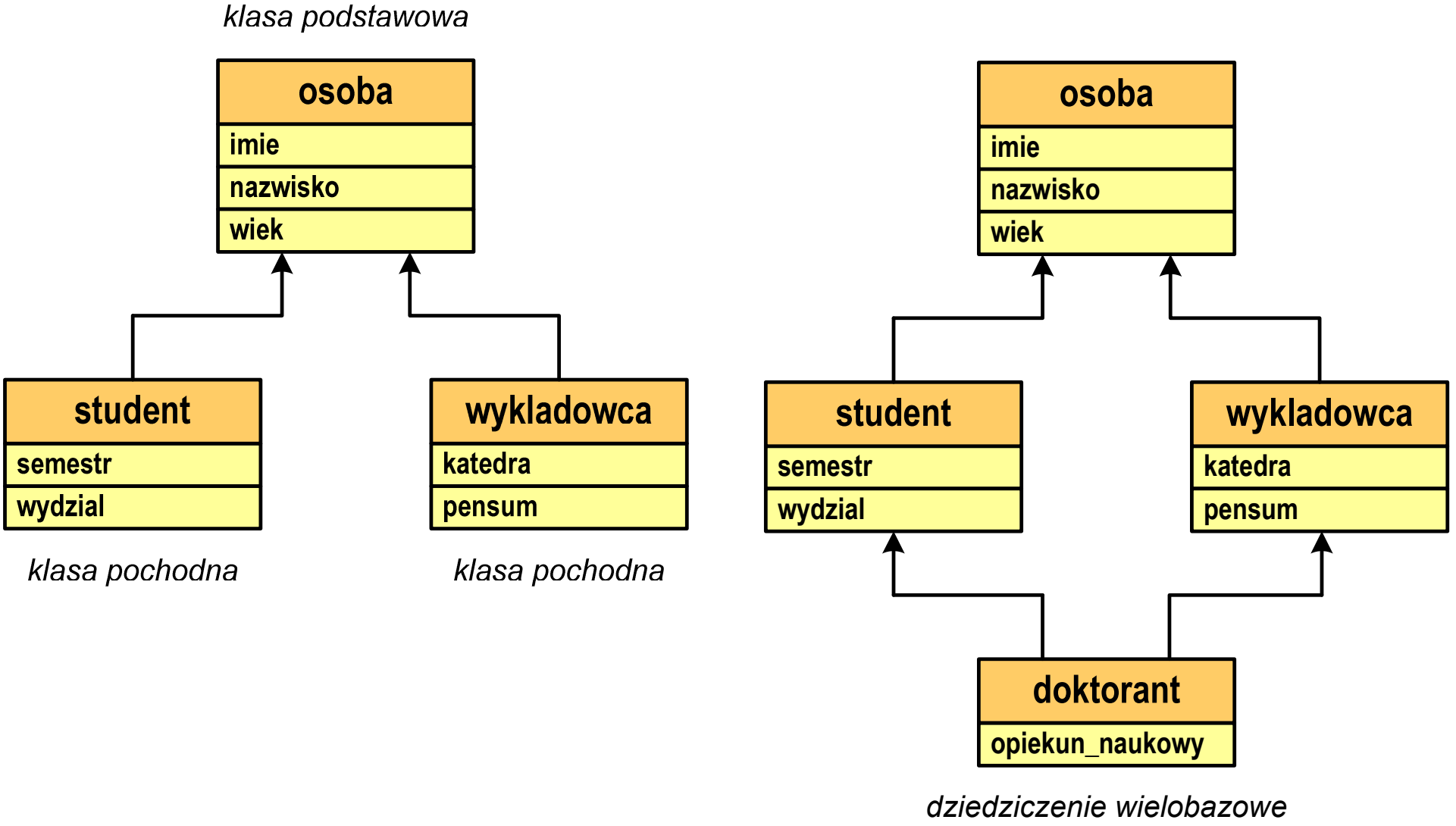
dr inż. Jarosław Forenc

Dziedziczenie

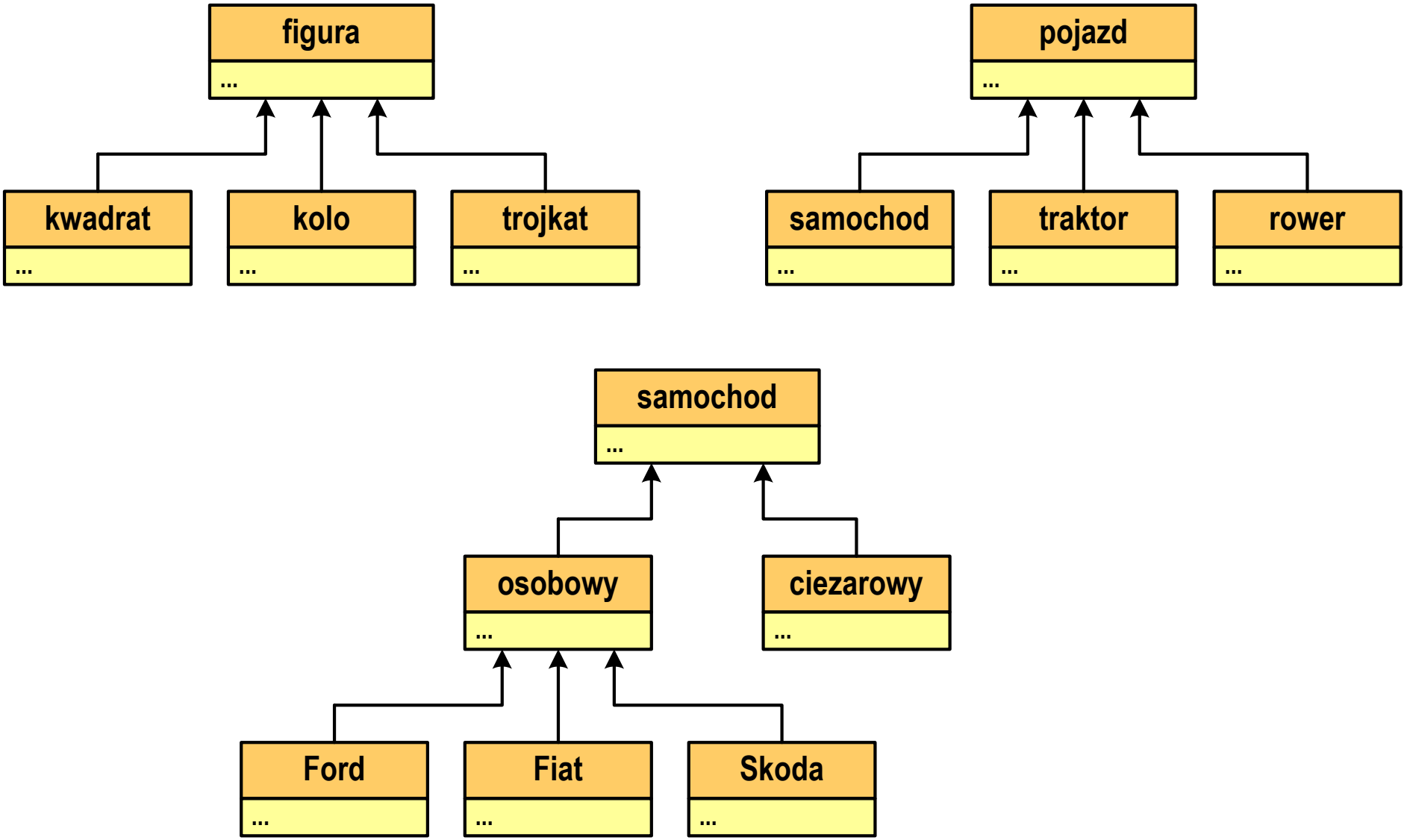
- dziedziczenie jest to technika pozwalającą na definiowanie nowej klasy przy wykorzystaniu klasy już istniejącej
- polega na przejmowaniu jednej klasy (bazowej, podstawowej) przez inną klasę (pochodną)
- przy dziedziczeniu, w skład obiektów klasy pochodnej automatycznie wchodzą pola klasy bazowej
- do obiektów klasy pochodnej możemy stosować operacje zdefiniowane przez funkcje składowe klasy bazowej



Dziedziczenie - przykłady



Dziedziczenie - przykłady



Dziedziczenie

- przykładowa klasa podstawowa i klasa pochodna

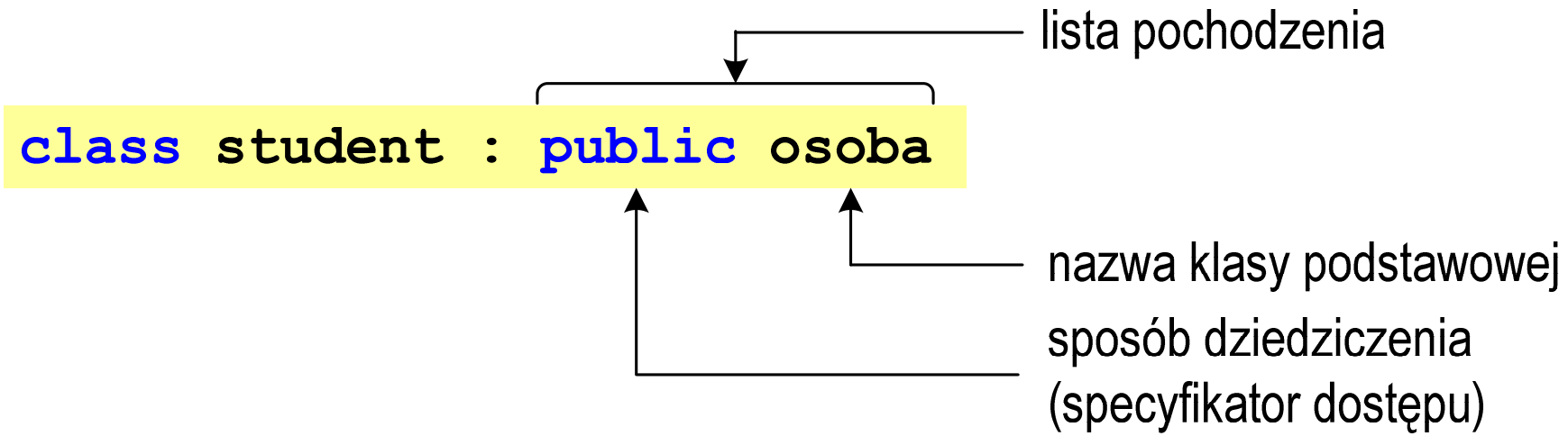
```
/* klasa podstawowa */

class osoba
{
   char *imie;
   char *nazwisko;
   int   wiek;
 public:
   osoba(char *i,char *n,int w);
   ~osoba()
   void drukuj();
};
```

```
/* klasa pochodna */

class student : public osoba
{
   char *wydzial;
   int   semestr;
 public:
   student(char *i, char *n,
        int w, char *wy, int s);
   ~student()
   void drukuj();
   void promocja();
};
```

Dziedziczenie



- w klasie pochodnej można zdefiniować:
  - dodatkowe dane składowe
  - dodatkowe funkcje składowe
  - dane i funkcje o takich samych nazwach jak w klasie podstawowej (dane i funkcje z klasy podstawowej są zasłaniane)
- jeśli nie podamy sposobu dziedziczenia, to domyślnie będzie to private

Dziedziczenie - sposoby dziedziczenia

klasa podstawowa \ sposób dziedziczenia	private	protected	public
private	-	-	-
protected	private	protected	protected
public	private	protected	public

- podczas dziedziczenia nie są dziedziczone: konstruktor, destruktor i operator przypisania "**=**"

Przykład:

```
student st1("Jan","Kos",20,"WE",2);     - deklaracja obiektu
st1.drukuj();                           - wywołanie funkcji z klasy student
st1.osoba::drukuj();                    - wywołanie funkcji z klasy osoba
```

- możliwe jest dziedziczenie wielokrotne, tzn. klasa pochodna może być klasą podstawową dla innej klasy

Przykład: dziedziczenie (1/4)

```
#include <iostream>
#include <cstring>
using namespace std;

class osoba
{
  private:
    char *imie;
    char *nazwisko;
    int   wiek;
  public:
    osoba(char*,char*,int);
    ~osoba();
    void drukuj();
};
```

```
class student : public osoba
{
   private:
     char *wydzial;
     int   semestr;
   public:
     student(char*,char*,int,
             char*,int);
     ~student();
     void drukuj();
     void promocja();
};
```

klasa podstawowa (bazowa)

klasa pochodna

Przykład: dziedziczenie (2/4)

```
osoba::osoba(char *i, char *n, int w)
{
    imie = new char[strlen(i)+1];
    nazwisko = new char[strlen(n)+1];
    strcpy(imie,i);
    strcpy(nazwisko,n);
    wiek = w;
}

osoba::~osoba()
{
    delete [] imie;
    delete [] nazwisko;
}

void osoba::drukuj()
{
    cout << imie << " " << nazwisko;
    cout << " " << wiek << endl;
}
```

konstruktor klasy osoba

destruktor klasy osoba

Przykład: dziedziczenie (3/4)

```
student::student(char *i,char *n,int w,char *wy,int s) : osoba(i,n,w)
{
    wydzial = new char[strlen(wy)+1];
    strcpy(wydzial,wy);
    semestr = s;
}

student::~student()
{
    delete [] wydzial;
}

void student::drukuj()
{
    osoba::drukuj();
    cout << "Wydzial: " << wydzial;
    cout << " Semestr: " << semestr << endl;
}
```

konstruktor klasy student

lista inicjalizacyjna konstruktora klasy student zawierająca wywołanie konstruktora klasy podstawowej (osoba)

destruktor klasy student

Przykład: dziedziczenie (4/4)

```
void student::promocja()
{
     semestr++;
}

int main(void)
{
     student st1("Jan","Kowalski",20,"WE",2);

     st1.drukuj();
     st1.promocja();
     st1.drukuj();
     st1.osoba::drukuj();
}
```

jako pierwszy zostanie wywołany konstruktor klasy podstawowej (osoba) a po nim konstruktor klasy pochodnej (student)

- kolejność wywołania konstruktorów:
  - konstruktor klasy podstawowej
  - konstruktor obiektów składowych „goszczących” w klasie pochodnej
  - konstruktor klasy pochodnej

Przykład: dziedziczenie (4/4)

```
void student::promocja()
{
     semestr++;
}

int main(void)
{
     student st1("Jan","Kowalski",20,"WE",2);

     st1.drukuj();
     st1.promocja();
     st1.drukuj();
     st1.osoba::drukuj();
}
```

jako pierwszy zostanie wywołany konstruktor klasy podstawowej (osoba) a po nim konstruktor klasy pochodnej (student)

- kolejność wywołania destruktorów jest odwrotna w stosunku do konstruktorów - jako pierwszy jest wywoływany destruktor klasy student, a po nim destruktor klasy osoba

Przykład: dziedziczenie (4/4)

```
void student::promocja()
{
     semestr++;
}

int main(void)
{
     student st1("Jan","Kowalski",20,"WE",2);

     st1.drukuj();
     st1.promocja();
     st1.drukuj();
     st1.osoba::drukuj();
}
```

jako pi
konstr
(osoba)
l

```
Jan Kowalski 20
Wydzial: WE Semestr: 2
Jan Kowalski 20
Wydzial: WE Semestr: 3
Jan Kowalski 20
```

- kolejność wywołania destruktorów jest odwrotna w stosunku do konstruktorów - jako pierwszy jest wywoływany destruktor klasy student, a po nim destruktor klasy osoba