Dziedziczenie I

Mechanizm dziedziczenia

Dziedziczenie to mechanizm umożliwiający definiowanie nowej klasy przy wykorzystaniu klasy wcześniej zdefiniowanej.

Jeśli mamy gotową klasę **klocek**, a potrzebna jest nam klasa **klocekObramowany**, mamy do wyboru kilka dróg:

1.
   - przerobić klasę **klocek**, na klasę **klocekObramowany**
   - mamy nową klasę, ale tracimy starą
2.
   - dodać klasę **klocekObramowany** na wzór klasy **klocek**
   - musimy pamiętać, by każdą poprawkę w klasie **klocek** nanosić również w klasie **klocekObramowany**
3.
   - zbudować klasę **klocekObramowany** wykorzystując klasę **klocek** i mechanizm dziedziczenia
   - zachowujemy starą klasę, a w nowej klasie piszemy tylko to co chcemy dodać lub zmienić.

Dziedziczenie jest trzecią cechą charakterystyczną programowania obiektowego.

Klasa pochodna

Nowa klasa zbudowana na podstawie starej klasy przy użyciu dziedziczenia nazywa się **klasą pochodną**.

Klasa pochodna **klocekObramowany** może być zdefiniowana na przykład tak:

```
class klocekObramowany : public klocek{
private:
   int kolorRamki;
public:
   klocekObramowany(int i, int j, int kol, int kolRamki){
     wiersz=i;
     kolumna=j;
     kolor=kol;
     kolorRamki=kolRamki;
```

```
    }
    void pokaz(){
      boxFill(wiersz-25,kolumna-25,wiersz+25,kolumna+25,kolor);
      box(wiersz-25,kolumna-
25,wiersz+25,kolumna+25,kolorRamki);
    }
};
```

Klasa **klocek** jest dla klasy **klocekObramowany** **klasą podstawową**, albo inaczej **klasą macierzystą**.

Mówimy też, że klasa pochodna jest **wywiedziona** z klasy macierzystej.

- Definicja klasy macierzystej musi być znana kompilatorowi w momencie kompilowania klasy pochodnej
- Klasa pochodna dziedziczy, t.zn. zawiera wszystkie składniki klasy macierzystej.

Czego się nie dziedziczy?

W C++ nie dziedziczy się konstruktorów, destruktorów i operatora przypisania

- Niedziedziczenie konstruktorów i destruktorów jest dość intuicyjne.
- Obiekt klasy pochodnej generalnie ma więcej cech, więc trudno oczekiwać, żeby konstruktor klasy podstawowej skonstruował poprawnie obiekt klasy pochodnej, a destruktor go zlikwidował
- Operator przypisania nie konstruuje obiektu od zera, ale likwiduje jego starą zawartość i wprowadza nową. W tym sensie wypełnia rolę zbliżoną do konstruktora, więc też się go nie dziedziczy

Co możemy w klasie pochodnej

Klasa pochodna zawiera składniki odziedziczone. Ponadto klasa ta może zawierać:

- dodatkowe dane składowe
- dodatkowe funkcje składowe
- nowe definicje funkcji składowych już zdefiniowanych w klasie macierzystej

Tak może wyglądać nowa wersja funkcji **jedz** w klasie **taksowka**

```
void taksowka::jedz(float ile_kilometrow){
  float ile_przejedziemy=min(ile_kilometrow,paliwo*10);
  przebieg+=ile_przejedziemy;
  paliwo-=ile_przejedziemy/10;
  taksometr+=ile_przejedziemy*3.5;
}
```

Nowe definicje funkcji w klasach pochodnych są przykładem zasłaniania, a nie przeładowania.

Do wersji funkcji z klasy macierzystej można się w klasie pochodnej odwołać wykorzystując kwalifikator zakresu:

```
taksowka wawel_taxi;
....
wawel_taxi.samochod::jedz(10.4);
....
```

Dziedziczenie, a kompozycja

- Dziedziczenie nie zastępuje kompozycji
- Proces kompozycji wykorzystujemy, jeśli między obiektami zachodzi relacja "ma": samochód ma silnik, komputer ma pamięć, okno ma ramę
- Proces dziedziczenia wykorzystujemy, jeśli między obiektami zachodzi relacja "jest": samochód jest pojazdem, klocek obramowany jest klockiem, wojskowy jest osobą.
- Tak w przypadku dziedziczenia jak i kompozycji mamy do czynienia z podobiektem: obiekt klasy bazowej bądź obiekt będący składnikiem zajmują część pamięci wydzielonej dla całego obiektu.

Dziedziczenie kilkupokoleniowe

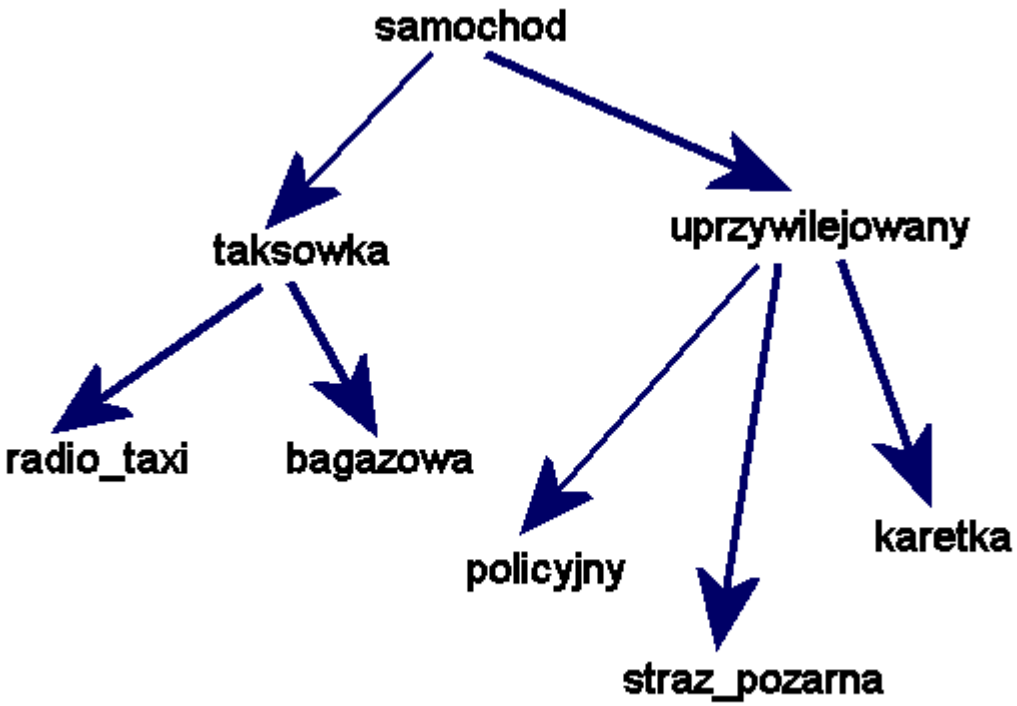
Klasa pochodna od jakiejś klasy sama może być klasą macierzystą dla innej klasy.

```
class pojazd {
....
}

class samochod: public pojazd{
....
}

class taksowka: public samochod{
```

```
....
}

class radio_taxi: public taksowka{
....
}
```

Hierarchia klas

- Jedna klasa macierzysta może mieć wiele klas pochodnych.
- Każda z tych klas pochodnych może być klasą macierzystą kolejnych klas pochodnych.
- W ten sposób powstaje tak zwana **hierarchia klas** lub **hierarchia dziedziczenia**.



- Zwyczajowo klasę najbardziej pierwotną wizualizuje się najbardziej w górze, a kolejne klasy pochodne rozrastają się w dół.
- Stąd częściowy porządek poniędzy klasami jaki wprowadza hierarchia dziedziczenia opisuje się przy pomocy zwrotu: **A** jest wyżej(niżej) od klasy **B** w hierarchii dziedziczenia
- W zwrocie tym "wyżej" oznacza, że **A** jest przodkiem **B**, a "niżej", że **B** jest przodkiem **A**.

Ustawienie klas w hierarchię dziedziczenia, to nie tylko oszczędność czasu

przy definiowaniu klas znajdujących się na dole hierarchii. Ważniejsze jest to, że obiekty, które znajdują się na dole hierarchii, mogą być czasem traktowane tak, jakby były obiektami z góry hierarchii.

Pożytki z dziedziczenia

- Oszczędzamy czas: definiujemy tylko różnice, a nie wszystko od nowa
- Aby użyć klasy do tworzenia klas pochodnych, nie musimy znać kodu klasy podstawowej, wystarczy gdy wiemy jak z niej korzystać
- przybliża to sztukę programowania do życia codziennego
- jest wielkim ułatwieniem w programowaniu zespołowym
- jest nieocenione przy dużych projektach, bo pozwala pracować lokalnie: nie musimy przez cały czas pamiętać wszystkich szczegółów i o nie się martwić