Dziedziczenie II

Dostęp do składników w klasie pochodnej

- Składniki prywatne klasy macierzystej są zawsze dziedziczone jako prywatne, co oznacza, że w klasie pochodnej są one dostępne tylko poprzez ewentualne publiczne lub chronione funkcje składowe klasy macierzystej.
- Składniki chronione w klasie macierzystej są dla wszystkich klas pochodnych od tej klasy dostępne tak jak składniki publiczne, natomiast dla pozostałych funkcji te składniki nie są dostępne.
- Składniki publiczne klasy macierzystej są dostępne dla wszystkich, a więc w szczególności dla klas pochodnych

Tak więc klasa podstawowa przez odpowiednią specyfikację dostępu słowami **`public`**, **`protected`**, **`private`** może decydować, które składniki zamierza udostępnić wszystkim funkcjom, które zastrzega tylko dla swoich klas pochodnych, a które zachowuje tylko na swój prywatny użytek.

Klasa pochodna też decyduje

- Składniki chronione i publiczne są udostępniane przez klasę macierzystą klasie pochodnej.
- Klasa pochodna, jeśli chce, może zaostrzyć prawa dostępu do tych składników.
- Odbywa się to poprzez ustalenie **sposobu dziedziczenia** określanego słowem **`public`**, **`protected`** lub **`private`** poprzedzającym nazwę klasy macierzystej na liście dziedziczenia

Sposób dziedziczenia `public`:
składniki **`public`** z klasy podstawowej mają w klasie pochodnej również dostęp **`public`**, a odziedziczone składniki **`protected`** mają w klasie pochodnej dostęp **`protected`**.

Sposób dziedziczenia `protected` :
składniki **`public`** oraz **`protected`** w klasie pochodnej mają dostęp **`protected`**.

Sposób dziedziczenia `private` :
składniki **`public`**
oraz **`protected`** w klasie pochodnej mają dostęp **`private`**.

Tak więc klasa, która dziedziczy składniki jako **`private`** zamyka do nich dostęp wszystkim ewentualnym klasom pochodnym od tej klasy.

Domyślnym sposobem dziedziczenia składników jest **`private`** w przypadku klasy pochodnej utworzonej za pomocą słowa **`class`**, a **`public`**, w przypadku klasy pochodnej utworzonej za pomocą słowa **`struct`**.

Konstruktory klasy pochodnej

- Konstruktor klasy pochodnej wygląda tak jak zwykły konstruktor, z tym, że na jego liście inicjalizacyjnej można umieścić wywołanie konstruktora klasy macierzystej.
- Konstruktor klasy pochodnej
  - najpierw wywołuje konstruktor klasy macierzystej (z listy inicjalizacyjnej lub domniemany)
  - potem wywołuje konstruktory klas składowych klasy pochodnej (jeśli takie istnieją)
  - na końcu wykonuje się sam

- W przypadku dziedziczenia kilkupokoleniowego na liście inicjalizacyjnej konstruktora można umieścić tylko wywołanie konstruktora klasy macierzystej, a więc bezpośredniego przodka.
- Ewentualne wywołania konstruktorów klas wcześniejszych w hierarchii dziedziczenia odbywają się pośrednio, bo konstruktor klasy macierzystej z kolei woła konstruktor swojej klasy macierzystej jeśli ją ma itd.

Destruktor i operator przypisania generowane automatycznie w klasie pochodnej

Jeśli klasa pochodna nie definiuje swojego destruktora, to kompilator próbuje go wygenerować automatycznie metodą "składnik po składniku". Wykorzystuje przy tym destruktor klasy macierzystej do zlikwidowania składników pochodzących z klasy macierzystej. Podobnie dzieje się z konstruktorem kopiującym i z operatorem przypisania.

Jeśli destruktor, konstruktor kopiujący lub operator przypisania w klasie macierzystej są prywatne to kompilator nie wygeneruje automatycznie odpowiedniej funkcji w klasie pochodnej.