04. Dziedziczenie

1 Dziedziczenie

Dziedziczenie (ang. *inheritance*) jest mechanizmem umożliwiającym współdzielenie funkcjonalności między klasami. Klasa może dziedziczyć po innej klasie — taką klasę nazywamy *pochodną* lub *potomną* (ang. *subclass, derived class, child class*) — co oznacza, że oprócz swoich własnych atrybutów oraz metod, uzyskuje także te pochodzące z klasy, z której dziedziczy — nazywanej klasą *bazową* (ang. *base class, superclass*).

```
1  class Animal {
2    public:
3      Animal() { age = 0; }
4      int getAge() { return age; }
5
6    protected:
7      int age;
8  };
9
10 class Cat : public Animal {
11   public:
12     Cat() { }
13 };
```

W klasie `Cat` dostępne są teraz atrybuty i metody z klasy `Animal`, dla których określono dostęp na poziomie `public` lub `protected`:

```
1  Cat cat;
2  std::cout << cat.getAge() << std::endl;
```

Zadanie 01 Zapoznaj się z kodem źródłowym zawartym w pliku *pob04-animals.cpp*. Dopisz klasy `Dog` oraz `Lion` dziedziczące z klasy `Animal`, które w konstruktorze wypiszą komunikat *„tworze obiekt klasy XYZ”*. Utwórz obiekty tych klas, co zaobserwowałeś?

2 Zmienna `this`

Zmienna `this` umożliwia odwołanie się w metodach niestatycznych (wszystkie, które do tej pory poznaliśmy) do obiektu, na rzecz którego dana metoda została wywołana. Dzięki temu możliwe jest np. uzyskanie dostępu do atrybutu przykrytego przez nazwę argumentu:

```
1  Animal(int age) {
2    this->age = age;
3  }
```

Zadanie 02 Do klas `Dog` oraz `Lion` dopisz jednoargumentowe konstruktory inicjalizujące wartość atrybutu `age`, przyjmujące argument w postaci `int age`.

Zadanie 03 Utwórz obiekty klasy `Dog` oraz `Lion` używając konstruktora jednoargumentowego. Wywołaj dziedziczone metody `getAge()` (odkomentuj odpowiednie linie).

Zadanie 04 Co się stanie jeżeli fragment `Dog : public Animal` zostanie zastąpiony przez `Dog : protected Animal` lub `Dog : private Animal`?

Zadanie 05 Dopisz metodę `getHumanAge()` w klasie `Dog`, która zwróci wiek zwierzęcia w ludzkich latach (wynikiem ma być wiek zwierzęcie pomnożony prez 4). Dla sprawdzenia odkomentuj odpowiednie linie.

3 Dziedziczenie wielokrotne

W języku C++ możliwe jest dziedziczenie po kilku klasach jednocześnie:

```
1 class Cat : public Animal, public Pet {
2   // ...
3 }
```

Zadanie 06 Jakie problemy mogą wystąpić podczas wielokrotnego dziedziczenia? Zwróć uwagę na mogące się pokrywać nazwy w klasach bazowych. W jaki sposób możliwe jest uzyskanie do nich dostępu?

Zadanie 07 Odkomentuj klasę `Pet`. Zmień definicję klasy `Dog`, aby dziedziczyła również z klasy `Pet`. Zwróć uwagę na to, że przy tworzeniu klasy pochodnej wywoływany jest konstruktor bezargumentowy klasy bazowej. W jakiej kolejności wywoływane są konstruktory gdy klasa dziedziczy po kilku klasach? Sprawdź co się stanie gdy zamienisz kolejność dziedziczenia.

Zadanie 08 Dodaj dwuargumentowy konstruktor do klasy `Dog(int age, std::string name)`, który poza wiekiem ustawi również imię zwierzęcia. Przetestuj odpowiednim fragmentem kodu.

Zadanie 09 Naszkicuj diagram dziedziczenia przedstawionych klas. Zwyczajowo zależność *B dziedziczy z A* oznacza się poprzez narysowanie strzałki od B do A.