Metodyka i Techniki Programowania II

C++: Dziedziczenie 1

Autorzy instrukcji: dr inż. Zbigniew Hulicki, mgr inż. Artur Kos

Kiedy zaczynamy analizować klasy, które mają jedną lub więcej cech wspólnych, to natychmiast nasuwa się wniosek, że warto byłoby je w jakiś sposób uporządkować. Oczywistym sposobem uporządkowania może być uogólnienie, tzn. należy zdefiniować taką klasę, która będzie zawierać tylko te atrybuty i operacje, które są wspólne dla pewnej grupy klas.
W językach obiektowych taką uogólnioną klasę nazywa się *superklasą* lub *klasą rodzicielską*, a każda z klas związanej z nią grupy nazywa się *podklasą* lub *klasą potomną*
W języku C++ odpowiednikami tych terminów jest *klasa bazowa* (ang. base class) i *klasa pochodna* (ang. derived class).
Klasa bazowa może zawierać tylko te elementy składowe, które są wspólne dla wyprowadzanych z niej klas pochodnych. Zwykle tę własność wyraża się nieco inaczej, tzn. mówimy, że klasa pochodna dziedziczy wszystkie cechy swojej klasy bazowej.

Gdyby dziedziczenie ograniczyć jedynie do przekazywania klasie pochodnej cech klasy bazowej, to byłoby to mało przydatne (kopiowanie, klonowanie). Dlatego w języku C++ **mechanizm dziedziczenia** wzbogacono o następujące możliwości:
1. W klasie pochodnej można dodawać nowe zmienne i funkcje składowe.
2. W klasie pochodnej można redefiniować funkcje składowe klasy bazowej.
Tak określony mechanizm pozwala zbliżyć się do tego, co dziedziczenie oznacza w języku potocznym, tzn. sensownie określone klasy pochodne zawierają cechy klasy (lub kilku klas) bazowej oraz nowe cechy, które wyraźnie odróżniają je od klasy bazowej.

Zadanie 1

1. Spróbuj uruchomić poniższy kod:

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

class student
{
private:
    void printOpis();
public:
    student();
    string Opis_ = "student grupy";
};

class starosta
{
public:
    string Opis_ = "starosta grupy";
};

student::student()
{
    cout << "Tworzenie obiektu klasy student o nazwie: " << Opis_ << endl;
}

void student::printOpis()
{
    cout << "Opis: " << Opis_ << endl;
}

int main()
{
    student stud;
    stud.printOpis();
}
```

   Dlaczego program się nie kompiluje? Usuń błąd i uruchom program. Zwróć uwagę na treść wyświetlaną na ekranie.
2. Uzupełnij program tak, aby klasa starosta dziedziczyła po klasie student.
3. Utwórz nowy obiekt klasy starosta i uruchom program. Jaki jest rezultat ? Jaki z tego płynie wniosek ?
4. Wywołaj dla niego metodę printOpis(). Jaki jest rezultat ? Jaki z tego płynie wniosek ?

5. Następnie kod umieść **W SWOJEJ GAŁĘZI** w repozytorium zdalnym.

Zadanie 2

Korzystając ze zmodyfikowanego w zadaniu 1 kodu zadania:

1. Uzupełnij klasę starosta tak, aby posiadała własną metodę printOpis(), różniącą się jedynie treścią wyświetlanego komunikatu.
2. Wywołaj dla niego metodę printOpis(). Jaki jest rezultat ? Jaki z tego płynie wniosek ?
3. Następnie kod umieść **W SWOJEJ GAŁĘZI** w repozytorium zdalnym

Zadanie 3

1. Uruchom poniższy kod:

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

class student
{
public:
    string imie_nazwisko_ = "NO_NAME";
        unsigned int nr_indeksu_ = 0;
    student(string imie_nazwisko, unsigned int
    nr_indeksu); string Opis_ = "student grupy"; void
    printOpis();
    void printDane()
    {
        cout << " Metoda printDane klasy bazowej" << endl;
                cout << " imie nazwisko " << imie_nazwisko_ << endl;
                cout << " nr indeksu " << nr_indeksu_  << endl;
    }
};

class starosta : public student
{
public:
    string email_ = "no@noemail";
    starosta(string imie_nazwisko, unsigned int nr_indeksu, string
    email); string Opis_ = "starosta grupy";
};

starosta::starosta(string imie_nazwisko, unsigned int nr_indeksu, string email) :
  student(imie_nazwisko,  nr_indeksu), email_(email)
{
    cout << "Tworzenie obiektu klasy starosta o nazwie: " << Opis_ << endl;
}

student::student(string imie_nazwisko, unsigned int nr_indeksu) :
  imie_nazwisko_(imie_nazwisko)
{
    nr_indeksu_ = nr_indeksu;
    cout << "Tworzenie obiektu klasy student o nazwie: " << Opis_ << endl;
}

void student::printOpis()
{
    cout << "Opis: " << Opis_ << endl;
}

int main()
{
    student stud("Jan Kowalski",7);
    stud.printOpis();
    cout << "Dane:" << stud.imie_nazwisko_ << " " << stud.nr_indeksu_ << endl;
    starosta star("Aleksandra Nowak",999,"mail@nomail.dot"); star.printOpis();

    cout << "Dane:" << star.imie_nazwisko_ << " " << star.nr_indeksu_ << endl;
}
```

1. Przeanalizuj jego działanie. Zwróć uwagę jak tworzony jest nowy obiekt klasy pochodnej.
2. Zmodyfikuj definicję klasy podstawowej tak, aby składowe imie_nazwisko i nr_indeksu były prywatne.
3. Uruchom program. Czy jego działanie zmieniło się ? Jaki z tego płynie wniosek ?
4. Wykorzystaj metodę printDane().

5. Dodaj metodę printDane() do klasy starosta, która uwzględni także email_. Czy dostęp do składowych imie_nazwisko i nr_indeksu jest możliwy ? Popraw program, aby wyświetlał poprawnie wszystkie dane klasy starosta.
6. Następnie kod umieść **W SWOJEJ GAŁĘZI** w repozytorium zdalnym