# *Programowanie obiektowe*

## **Zajęcia 5.** Polimorfizm, klasy abstrakcyjne, rodzaje rzutowań

**Zadanie 1.** Polimorfizm w C++

1. Utwórz w wybranym miejscu na dysku folder `PolimorfizmCpp`.
2. W VSC otwórz utworzony w poprzednim punkcie folder.
3. Do projektu dodaj plik `main.cpp`.
4. Wykonaj implementację poniższych klas (zadbaj by stan obiektów był zabezpieczony zgodnie z założeniami enkapsulacji i można było się do nich dostać tylko i wyłącznie za pomocą dedykowanych metod; do projektu dodaj odpowiednie pliki `cpp` oraz `h` dla każdej z definiowanych klas.):
   (a) `Osoba`, którą cechują następujące własności:
      - imię i nazwisko
      - ustawienie imienia i nazwiska
      - odczytanie imienia i nazwiska
      - metoda o nazwie przedstaw, której zadaniem jest wyświetlenie pełnego opisu osoby

   (b) `Student` (dziedziczy po klasie `Osoba`), którego cechują następujące własności:

- nazwa uczelni
- srednia ocen
- odczytanie nazwy uczelni
- zmiana uczelni
- odczytanie sredniej ocen
- zmiana sredniej
- wykonaj redefinicję metody przedstaw z klasy `Osoba`, tak, aby wyświetlała pełną informację na temat danego studenta.

(c) `Pracownik` (dziedziczy po klasie `Osoba`), którego cechują następujące własności:

- stanowisko
- pensja
- zmiana stanowiska
- odczytanie stanowiska
- zmiana pensji
- odczytanie pensji
- wykonaj redefinicję metody przedstaw z klasy `Osoba`, tak, aby wyświetlała pełną informację na temat danego pracownika.

5. W pliku `main.cpp` dodaj przykładową implementację funkcji main testującą podstawową funkcjonalność zaimplementowanych klas.

6. Skompiluj i uruchom plik `main.cpp` w trybie debug np. poprzez wciśnięcie przycisku F5 a następnie wybierz odpowiednią dla Twojego systemu i kompilatora konfigurację.

7. Zmodyfikuj użytą konfigurację (zapisaną w pliku `.vscode/tasks.json`) w taki sposób żeby zamiast pojedynczego pliku kompilacji uległy wszystkie pliki cpp zawarte w katalogu, w którym znajduje się aktualnie przez nas modyfikowany plik. Możesz tego dokonać poprzez zamianę użytego w pliku `${file}` na `${fileDirname}/*.cpp` lub nieco ogólniej `${fileDirname}/*${fileExtname}`.

8. Utwórz funkcję przyjmującą obiekt klasy `Osoba` przez referencję oraz przez wskaźnik. Np.:

```
void foo(Osoba &os) {
  os.przedstaw();
}
void bar(Osoba *os) {
  os->przedstaw();
}
```

9. Utwórz obiekt klasy `Osoba`, wywołaj metody ustawiające wartości poszczególnych pól, a następnie przekaż obiekt do utworzonej w punkcie 8 funkcji. Skompiluj program i uruchom go w trybie debug (np. poprzez wciśnięcie F5).

10. Utwórz obiekt klasy `Pracownik`, wywołaj metody ustawiające wartości poszczególnych pól, a następnie przekaż obiekt do utworzonej w punkcie 8 funkcji. Skompiluj program i uruchom go w trybie debug (np. poprzez wciśnięcie F5).

11. Porównaj efekt wywołań w punktach od 9 do 10, zapisz swoje obserwacje i spróbuj je zinterpretować.

12. Zmodyfikuj deklarację funkcji `przedstaw` w klasie `Osoba` poprzez dodanie specyfikatora `virtual`.

13. Ponownie wykonaj kroki od 9 do 10, skompiluj program i uruchom go w trybie debug (np. poprzez wciśnięcie F5), zapisz swoje obserwacje i spróbuj je zinterpretować.

14. Zastąp definicję funkcji `przedstaw` w klasie `Osoba` następującą:

```
virtual void przedstaw() = 0;
```

15. Czy teraz możesz skompilować projekt? Postaraj się ustalić przyczynę problemu, opisz ją a następnie postaraj się zmodyfikować kod tak, żeby projekt znowu się kompilował.

16. Zmodyfikuj deklarację destruktora w klasie `Osoba` poprzez dodanie specyfikatora `virtual`. Ponownie skompiluj program i uruchom go w trybie debug (np. poprzez wciśnięcie F5). Co uległo zmianie? Zapisz swoje obserwacje i spróbuj je zinterpretować.

17. Zaimplementuj możliwość tworzenia nowych osób: studentów lub pracowników (w pętli), które będą przechowywane w tablicy. Do tego celu należy stworzyć proste menu, które będzie obsługiwało podawane z wejścia komendy. Użytkownik może podawać następujące komendy słowne: "pracownik" (utwórz obiekt klasy `Pracownik` i dodaj go do tablicy), "student" (utwórz obiekt klasy `Student` i dodaj go do tablicy), "wyswietl" (wyświetlenie zawartości tablicy, gdzie dla każdego obiektu zostanie wywołana jego metoda `przedstaw`), "wyjscie" (zakończenie działania programu). Aby to zrealizować wykonaj następujące kroki:

18. W celu przechowania utworzonych obiektów klas `Pracownik`, `Student` wykorzystaj np. kontener `vector`:

```
std::vector<Osoba*> tablicaOsob;
```

19. Skompiluj program i uruchom go w trybie debug (np. poprzez wciśnięcie F5), zapisz swoje obserwacje, wnioski i spróbuj je zinterpretować.