# Język C++ zajęcia nr 5

## Hermetyzacja

Komponenty klasy (dane składowe i funkcje składowe) mogą być związane z różnym rodzajem dostępu:

- Dostęp **prywatny** (etykieta **`private`**) - składowe prywatne są dostępne jedynie z wnętrza klasy. Oznacza to, że dane prywatne są dostępne tylko dla funkcji składowych klasy, a funkcje prywatne mogą być wywoływane tylko przez funkcje składowe tej klasy. Prawo dostępu do składowych prywatnych klasy mają również funkcje *zaprzyjaźnione* z tą klasą.
- Dostęp **publiczny** (etykieta **`public`**) - składowe publiczne są dostępne bez ograniczeń, również na zewnątrz klasy.
- Dostęp **zabezpieczony** (etykieta **`protected`**) - składowe zabezpieczone są dostępne na zasadach dostępu prywatnego oraz dodatkowo są dostępne w klasach powstałych z danej klasy przez dziedziczenie.

**Program „Ukrywanie składowych”.**

**Zinterpretuj poszczególne elementy kodu źródłowego!!!!**

**Ukrywanie składowych:**

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

//-------------------- Definicja klasy
class tramwaj
{
  //-------------------- Składowe prywatne
  private:
    int max_liczba_miejsc;              // Pojemność tramwaju
    int liczba_pasazerow;               // Liczba pasażerów wewnątrz
    int wolne_miejsca()                 // Prywatna funkcja (inline)
    {
      return max_liczba_miejsc - liczba_pasazerow;
    }
```

```
 //-------------------- Składowe publiczne
  public:
    void ustal_stan(int max);                  // Funkcje publiczne
    void wsiada(int liczba_osob);
    void wysiada(int liczba_osob);
    void podaj_stan();
};
//-------------------- Definicje funkcji składowych
void tramwaj::ustal_stan(int max)
{
   max_liczba_miejsc = max;
   liczba_pasazerow = 0;
   return;
}
void tramwaj::wsiada(int liczba_osob)
{
   if(wolne_miejsca()==0)
   {
     cout << "\n ** Brak miejsc: Nikt nie wsiada";
     return;
   }
   if(liczba_osob <= wolne_miejsca())
   {
     liczba_pasazerow = liczba_pasazerow + liczba_osob;
     cout << "\n<-- Wsiadaja wszyscy oczekujacy = " << liczba_osob
          << " osob";
   }
   else
   {
     cout << "\n<-- Wsiada jedynie " << wolne_miejsca()
          << " osob z " << liczba_osob << " oczekujacych";
    liczba_pasazerow = max_liczba_miejsc;
   }
   return;
}
void tramwaj::wysiada(int liczba_osob)
{
   if(liczba_osob <= liczba_pasazerow)
   {
     liczba_pasazerow = liczba_pasazerow - liczba_osob;
     cout << "\n    ---> Wysiada " << liczba_osob << " pasazerow";
   }
   else
   {
     liczba_pasazerow = 0;
     cout << "\n    ===> Wysiadaja wszyscy pasazerowie";
   }
   return;
}
```

```
void tramwaj::podaj_stan()
{
   cout << "\nLiczba wolnych miejsc: " << wolne_miejsca();
   return;
}

int main()
{
   tramwaj a;
   tramwaj b;
   //------------------- Scenariusz -------------------
   a.ustal_stan(36);
   b.ustal_stan(48);
   a.podaj_stan();
   b.podaj_stan();
   a.wsiada(14);
   b.wsiada(5);
   a.podaj_stan();
   b.podaj_stan();
   a.wsiada(30);
   a.wsiada(4);
   a.podaj_stan();
   a.wysiada(5);
   a.wysiada(45);
   b.wysiada(5);
   a.podaj_stan();
   b.podaj_stan();
}
```

**Oczekiwane wyniki:**

```
Liczba wolnych miejsc: 36
Liczba wolnych miejsc: 48
<-- Wsiadaja wszyscy oczekujacy = 14 osob
<-- Wsiadaja wszyscy oczekujacy = 5 osob
Liczba wolnych miejsc: 22
Liczba wolnych miejsc: 43
<-- Wsiada jedynie 22 osob z 30 oczekujacych
** Brak miejsc: Nikt nie wsiada
Liczba wolnych miejsc: 0
    ---> Wysiada 5 pasazerow
    ===> Wysiadaja wszyscy pasazerowie
    ---> Wysiada 5 pasazerow
Liczba wolnych miejsc: 36
Liczba wolnych miejsc: 48
```

# Obiektowe argumenty funkcji

Aktualne argumenty wywołania funkcji są często obiektami zdefiniowanych klas, a ich przekazywanie może odbywać się bezpośrednio poprzez **wartość obiektu, wskaźnik na obiekt lub referencję do obiektu.** Przekazywanie argumentu przez wskaźnik lub referencję jest polecane ze względu na mniejsze wymagania pamięciowe (stos procesora) i mniejszą czasochłonność operacji wywołania funkcji.

**Wprowadź program „Obiektowe argumenty funkcji i ich przekazywanie”.**

**Zinterpretuj poszczególne elementy kodu źródłowego!!!!**

**Skompiluj i uruchom program.**

```
#include <iostream>
using namespace std;
class pociag
{
  public:
   char odjazd[6];
   char rodzaj[20];
   char dokad[20];
   void ustal_rozklad(char *gd, char *rd, char *dd);
};

//-------------------- Definicja funkcji składowej

void pociag::ustal_rozklad(char *gd, char *rd, char *dd)
{
   int i;
   i=0;
   while(odjazd[i]=gd[i])i++;        // Kopiowanie znaków
   i=0;
   while(rodzaj[i]=rd[i])i++;
   i=0;
   while(dokad[i]=dd[i])i++;
}

//-------------------- Definicje funkcji globalnych

void komunikat_war(pociag pc)        // Przekazanie przez wartość
{
   cout << "\nPociag " << pc.rodzaj << " do " << pc.dokad
        << " odjedzie o godzinie " << pc.odjazd;
}
```

```
void komunikat_ref(pociag &pc)            // Przekazanie przez referencję
{
   cout << "\nPociag " << pc.rodzaj << " do " << pc.dokad
        << " odjedzie o godzinie " << pc.odjazd;
}
void komunikat_wsk(pociag *pc)            // Przekazanie przez wskaźnik
{
   cout << "\nPociag " << pc->rodzaj << " do " << pc->dokad
        << " odjedzie o godzinie " << pc->odjazd;
}

int main()
{
   pociag a;
   pociag *b=&a;
   a.ustal_rozklad("16:24", "osobowy", "Kielc");
   komunikat_war(a);
   komunikat_ref(a);
   komunikat_wsk(b);
}
```

**Oczekiwane wyniki:**

```
Pociag osobowy do Kielc odjedzie o godzinie 16:24

Pociag osobowy do Kielc odjedzie o godzinie 16:24

Pociag osobowy do Kielc odjedzie o godzinie 16:24
```

Zadanie CPP05 (do wysłania przez Moodle):

Zdefiniuj klasę **punkt** zawierającą:

- dane składowe: współrzędne x, y punktu na płaszczyźnie,

- funkcje składowe:

- **ustal_polozenie (x, y)**, która ustala współrzędne punktu w przestrzeni,
- **przesun_o_wektor (xx, yy)**, która przesuwa dany punkt o wektor posiadający współrzędne [xx, yy]
- **wyznacz_odleglosc()**, która oblicza i zwraca odległość euklidesową danego punktu od początku układu współrzędnych
- **podaj_polozenie( ... ),** która zwraca **przez referencje** aktualne współrzędne punktu.

Napisz funkcję **main** definiującą dwa punkty (obiekty klasy **punkt**) i zawierającą przykładowy scenariusz określający pewne położenia początkowe punktów, realizujący pewne ich przesunięcia na płaszczyźnie, podający ich nowe położenia oraz wyznaczający końcowe odległości punktów od początku układu współrzędnych.