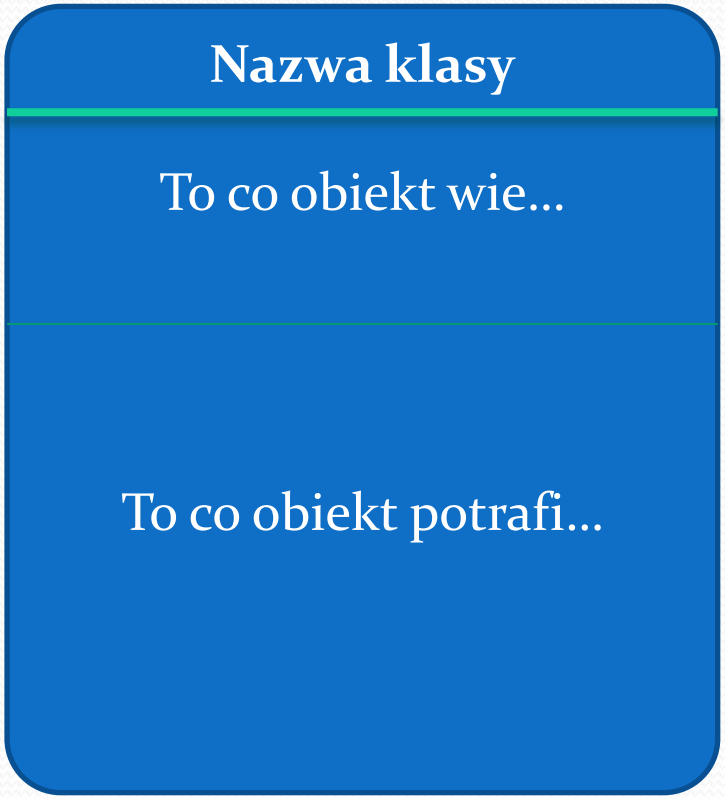
Klasy i obiekty

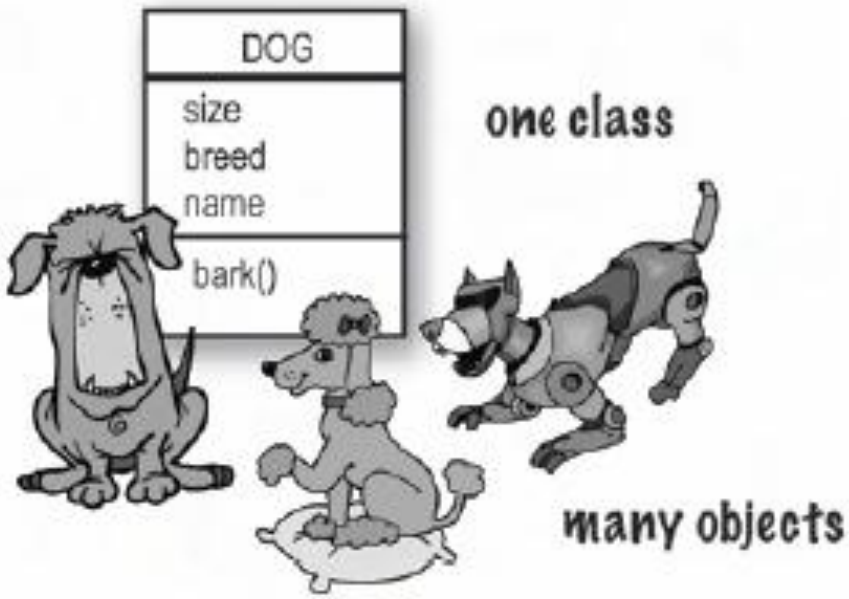
Wyższy wymiar magii.

Ogólna zasada pracy z obiektami i klasami



- Klasa będzie swoistym „projektem” obiektów tworzonych na jej podstawie.
- To co obiekt wie = atrybuty
- To co obiekt potrafi = metody

Klasa vs. obiekt



Kawałek kodu

```
1  class NazwaKlasy {
2          public: //pola i metody są publicznie dostępne
3
4           //definiowanie pól
5           int poleInt;
6           float poleFloat;
7
8           //deklarowanie metod
9           int Metoda1();
10          void Metoda2();
11         private:
12          int polePrywatne;
13          double polePrywatne2;
14
15          int metodaPrywatna(float parametr);
16     }; //pamiętaj o średniku!
```

Użycie obiektów

```
18  NazwaKlasy *obiekt = new NazwaKlasy();
19
20  obiekt->poleInt = 3;
21  obiekt->Metoda1();
22  obiekt->metodaPrywatna();
23
24  delete obiekt;
```

Konstruktory

```
class Punkt() {
    public:
    int x,y;
    string color;
    Punkt() {
        //to jest konstruktor domyślny
        this->x = 0;
        this->y = 0;
        this->color = "Brak";
    }
    Punkt(int iX, int iY, string iColor) {
        //to jest konstruktor parametrowy
        this->x = iX;
        this->y = iY;
        this->color = iColor;
    }
    Punkt(const Punkt &punkt) {
        //konstruktor kopiujący
        this->x = punkt.x;
        this->y = punkt.y;
        this->color = punkt.color;
    }
};
Punkt *p1 = new Punkt(); //wartosci "wyzerowane"
Punkt *p2 = new Punkt(5,5,"Biały"); //wartości ustawione
Punkt *p3 = new Punkt(p2); //wartosci skopiowane
```

Ćwiczenia

1. Skoro znamy konstruktory, to doczytaj o destruktorach.
2. Stwórz klasę „Liczba zespolona” i oprogramuj jej działanie zgodnie z regułami matematyki. Powinna posiadać pary metod get-set, trzy konstruktory i być rozdzielona do odpowiednich plików .h i .cpp

Ćwiczenia

- Napisz klasę realizującą rozwiązywanie równania kwadratowego.
- Klasa powinna posiadać pola prywatne A, B, C, reprezentujące współczynniki równania, wraz z odpowiednimi akcesorami i modyfikatorami.
- Dodatkowo należy stworzyć funkcję składową delta(), obliczającą wyróżnik trójmianu, oraz trzy funkcje obliczające miejsca zerowe.
- Klasa powinna być wyposażona w konstruktor domyślny oraz trójparametrowy konstruktor ogólny.
- Należy zastanowić się nad przypadkiem, gdy równanie staje się liniowym tzn. A = 0 i zaproponować rozwiązanie tego problemu.

Ciąg dalszy

- Obiekt stworzonej klasy może być wykorzystany następująco:
  - RownanieKwadratowe r;
  - double num;
  - cout << ’Podaj A:’; cin >> num; r.ustawA(num);
  - cout << ’Podaj B:’; cin >> num; r.ustawB(num);
  - cout << ’Podaj C:’; cin >> num; r.ustawC(num);
  - if(r.delta() > 0)
    - cout << "Pierwiastki rownania x1=" << r.obliczX1() << " x2=" << r.obliczX2() << endl;
  - else
  - if(r.delta() = 0)
    - cout << "Pierwiastek podwójny x12=" << r.obliczX12() << endl;
  - else
    - cout << "Brak pierwiastkow rzeczywistych" << endl;

Obiekty mogą być inicjowane następująco:

- RownanieKwadratowe r1; // Konstruktor inicjuje A=B=C=0;
- RownanieKwadratowe r2(3,2,1); // Konstruktor inicjuje A=3,B=2,C=1;

Ćwiczenia

Załóżmy, że gromadzimy wodę mineralną. Są trzy rodzaje butelek – o pojemności 2l (duże), średnie o pojemności 1l oraz małe o pojemności 0.5 litra. Stworzyć klasę MyWater z metodami:

- void addLarge(int) – dodaje do zapasu wody podaną jako argument liczbę dużych butelek.
- void addMedium(int) – dodaje do zapasu wody podaną jako argument liczbę średnich butelek.
- void addSmall(int) – dodaje do zapasu wody podaną jako argument liczbę małych butelek.

Dodatkowo należy zaimplementować metody umożliwiające uzyskanie informacji o tym ile jest każdego rodzaju butelek, oraz jaka jest łączna pojemność zgromadzonej wody.

Pojemności butelek (dużych, małych, średnich) przedstawić jako pola statyczne. Dostarczyć metod pozwalających uzyskiwać informacje o tych pojemnościach oraz je zmieniać.

Klasę przetestować wyprowadzając dane na ekran:

- Mam teraz 6.5 litrów wody.
- Dużych butelek: 2
- Małych butelek: 3
- Średnich butelek: 1

Live demo

- Dziedziczenie
- Metody wirtualne
- Redefinicja metod
- Składniki statyczne
- Przeciążanie operatorów

Ćwiczenia

1. Napisz program, w którym zdefiniujesz jedną z wymienionych hierarchii:
    1. Pojazdów
    2. Książek
    3. Co Ci do głowy wpadnie ☺

Zadbaj o poprawność schematu dziedziczenia, odpowiednie zadeklarowanie pól i metod w zależności od poziomu szczegółowości, itp.
Niezależnie od wszystkiego – prowadź ewidencję ile obiektów danej klasy aktualnie istnieje.

Ćwiczenia

- Wykonaj prostą grę polegającą na symulacji zachowań drogowych
- Zdefiniuj klasy użytkowników drogi (pieszych, samochodów, rowerów).
- Zadbaj o ich dziedziczenie z abstrakcyjnej klasy bazowej
- Każdy z obiektów danej klasy potrafi się poruszać po planszy (konsoli) z różną prędkością. Każdy z obiektów ma również swój odpowiedni znaczek
- Obiekty poruszają się po planszy w sposób losowy, gra kończy się w momencie „kraksy”.
- Zadbaj o krokową pracę (aby można było podglądać każde stadium).
- Przeprowadź symulacje dla różnej wielkości planszy oraz dla różnej liczby użytkowników drogi