Języki i Paradygmaty Programowania

dr inż. Michał Ciesielczyk
michal.ciesielczyk@put.poznan.pl

Konsultacje:
środa, 08:30-09:30, pokój BM-319
środa, 11:15-11:45, sala M-216

Dane do logowania

- login:

******@student.put.poznan.pl

- hasło:

******

Tematyka

- C++
  - Podstawowe paradygmaty programowania obiektowego
    - Hermetyzacja
    - Dziedziczenie
    - Polimorfizm
    - Abstrakcja
  - Wyjątki
  - Biblioteka standardowa STL
- Java
  - Podstawowe paradygmaty programowania obiektowego
    - Hermetyzacja
    - Dziedziczenie
    - Polimorfizm
    - Abstrakcja
  - Wyjątki
  - Programowanie wielowątkowe
  - GUI

Sposób oceny

- Obecność (obowiązkowa).
  - Maks. 1/3 nieobecności, czyli 4-5 zajęć.
- Zadania:
  - praca na zajęciach/w domu.
- Kolokwia:
  - Kolokwium I – C++
  - Kolokwium II – Java
- Zaliczenie kolokwium wymaga uzyskania powyżej 50% punktów.
- Projekt dodatkowy.

Ocena końcowa

Ocena	Wymagana liczba punktów
3,0	> 40%
3,5	> 55%
4,0	> 70%
4,5	> 80%
5,0	> 90%

50% × **kolokwium I** +
30% × **kolokwium II** +
20% × **projekt**

Otrzymanie oceny pozytywnej wymaga uzyskania powyżej 50% punktów z każdego kolokwium.

Zadania

- Zadania na laboratorium wykonywane są **samodzielnie**.
- Zadania których nie udało się zdążyć zrobić na laboratoriach automatycznie stają się **zadaniem domowym**.
- Zadania domowe należy obowiązkowo zrobić w ciągu tygodnia do dnia poprzedzającego zajęcia (włącznie).
- Rozwiązania wszystkich zadań należy umieścić na repozytorium wskazanym przez prowadzącego.

Projekt dodatkowy

- Temat uzgodniony z Prowadzącym
  - C++ lub Java
  - projekt i implementacja aplikacji użytkowej
- Dla studentów o **zaawansowanych** umiejętnościach
  - do zaliczenia przedmiotu wymagane pozytywne zaliczenie obu kolokwiów!
- Termin: 30.05.2017
  - prezentacja + dokumentacja (sprawozdanie) + kod źródłowy

Repozytorium SVN

formularz rejestracyjny:

goo.gl/forms/G5RjrWrEj7

zarządzanie projektami:

studentrepos.cie.put.poznan.pl

Środowisko

- C++
  - CLion
  - Visual Studio 2017
- Java
  - Eclipse (4.4 lub nowsze)
  - Netbeans (8.x lub nowsze)

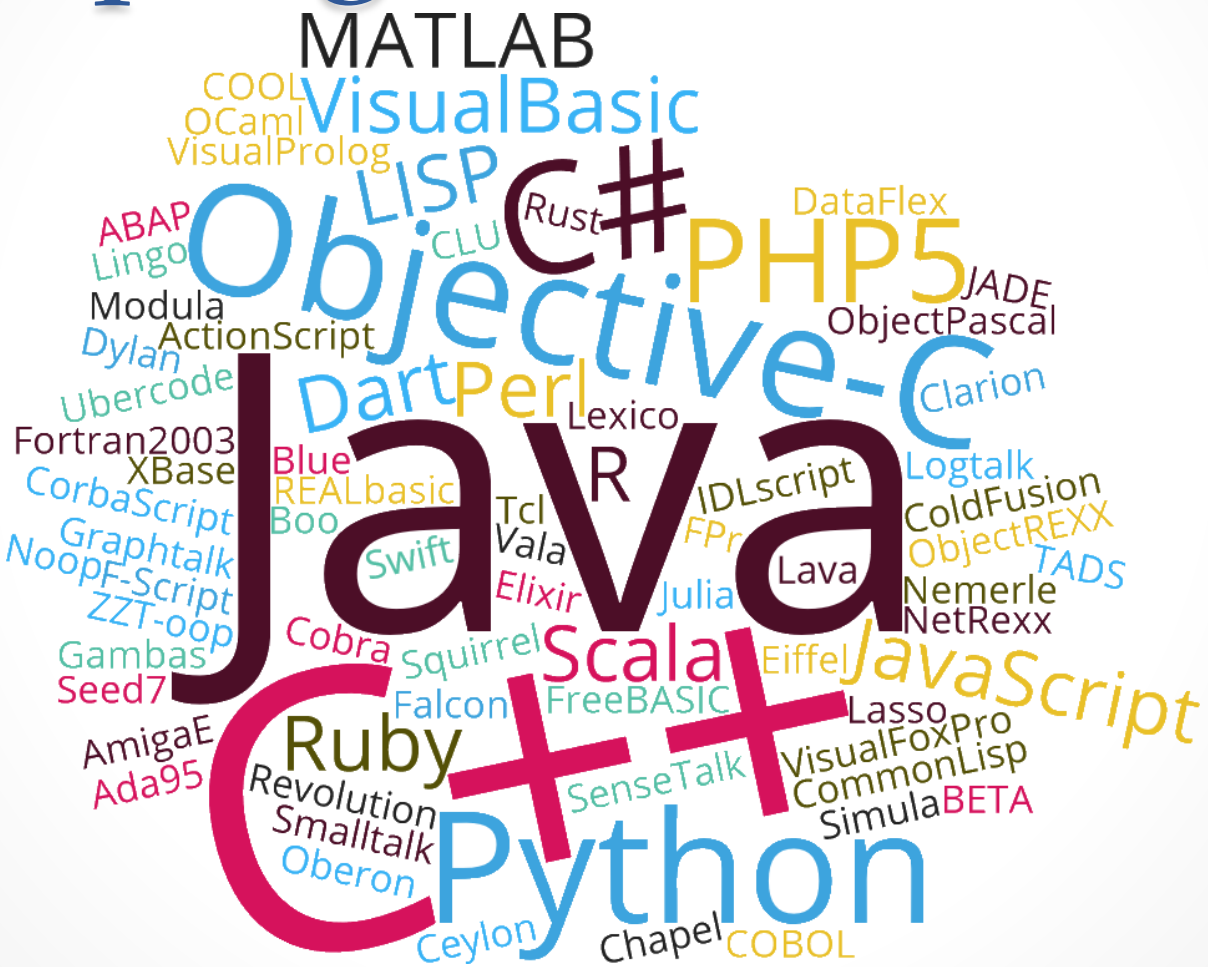
Literatura

- http://en.cppreference.com/w/
- http://www.cplusplus.com/
- https://isocpp.org/faq
- https://msdn.microsoft.com/en-us/library/3bstk3k5.aspx
- Programming – Principles and Practice Using C++. Bjarne Stroustrup.

Wprowadzenie do programowania obiektowego

Laboratorium 01

Obiektowe języki programowania



Popularność języków programowania

1	Java	21.145%
2	C	15.594%
3	C++	6.907%
4	C#	4.400%
5	Python	4.180%

http://www.tiobe.com/index.php/tiobe_index

Obiekty

- Pozwalają utworzyć model świata rzeczywistego (lub abstrakcyjnego) na potrzeby programu

- Są podstawowym elementem we wszystkich językach obiektowych
  - w tym C++ oraz Java

Obiekt - przykład

```
struct C {      // struktura o nazwie C
     int e;     // element o nazwie e
     int f();   // funkcja o nazwie f
};


C c;            // zmienna typu C
c.e = 7;        // dostęp do elementu e
int x = c.f(); // wywołanie funkcji f
```

Demo

• • •

przykładowa struktura ***Date***

Obiekty – podstawowe operacje

- Domyślny konstruktor (domyślnie: pusty)
  - Brak domyślnego konstruktora jeśli inny został zadeklarowany
- Konstruktor kopiowania (copy constructor)
  - (domyślnie: kopiuje wszystkie elementy klasy)
- Przypisania (copy assignment)
  - (domyślnie: kopiuje wszystkie elementy klasy)
- Destruktor (domyślnie: pusty)

- Na przykład

```
X x;        // konstruktor bezargumentowy
X x2 = x;   // konstruktor kopiowania
x = x2;     // przypisanie
```

Demo

Konstruktory

Instancje obiektów

- Statyczna alokacja pamięci:

```
Point p1(1, 2);
```

- Smart pointer:

```
std::unique_ptr<Point> p2 =
     std::make_unique<Point>(3,4);
```

- Dynamiczna alokacja pamięci:

```
Point* p3 = new Point(5, 6);
```

Demo

Tworzenie obiektów

Const

```
int x = 1;                  // zmienna
const int y = 2;            // stała
//…
x = y;                  // OK
y = x;                  // ERROR: y jest stałą
```

const

```
struct Date {
      // …
      int day() const;        // zwraca (kopię) dnia
      // …
      void add_day(int n);    // przesuwa datę o n dni do przodu
      // …
};
```

Różnica pomiędzy funkcjami mogących zmieniać obiekty („mutate”) a tymi, które nie mogą („const member functions”)

```
const Date dx(…);// deklaracja stałej dx
int d = dx.day();// OK
dx.add_day(4);   // ERROR: nie można zmienić stałej (immutable)
```

Demo

• • •

modyfikator const

Zadania…