Семинар 9 по ООП

Наследяване

За пореден път

- 4 те основни принципа на ООП са:
  - Абстракция
  - Енкапсулация
  - Наследяване
  - Полиморфизъм

Що е наследяване

- Наследяването е един от четирите основни принципа на ООП. Чрез него дадени класове получават същите член данни и член функции като други класове. Чрез него можем да създаваме сложни йерархии от класове, които споделят сходно поведение. Това ни улеснява защото не трябва да реализираме от нулата даден клас и също така дадени класове могат да споделят еднакъв интерфейс.

Модификатори за достъп при наследяване

- В C++ имаме 3 вида наследяване: public, private и protected.
- Най-често се използва public наследяване
- Когато се окаже такъв модификатор на наследяването, в класа наследник дадена член данна получава по-рестриктивния от двата (този в класа родител и този на наследяването)

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

Пример

Малко за преобразуването на типове

- Всеки обект направен от някакъв клас наследник може да се преобразува към обект от типа на родителския клас, но обратното не е позволено.

Предефиниране на методи

- Освен, че можем да добавяме нови полета и методи, ние можем и да променяме съществуващи такива. Както в примера аз промених метода sound.
- Трябва просто да направим метод със същата сигнатура в класа наследник.
- Ако искаме да извикаме метод на родителя, трябва да го укажем експлицитно.

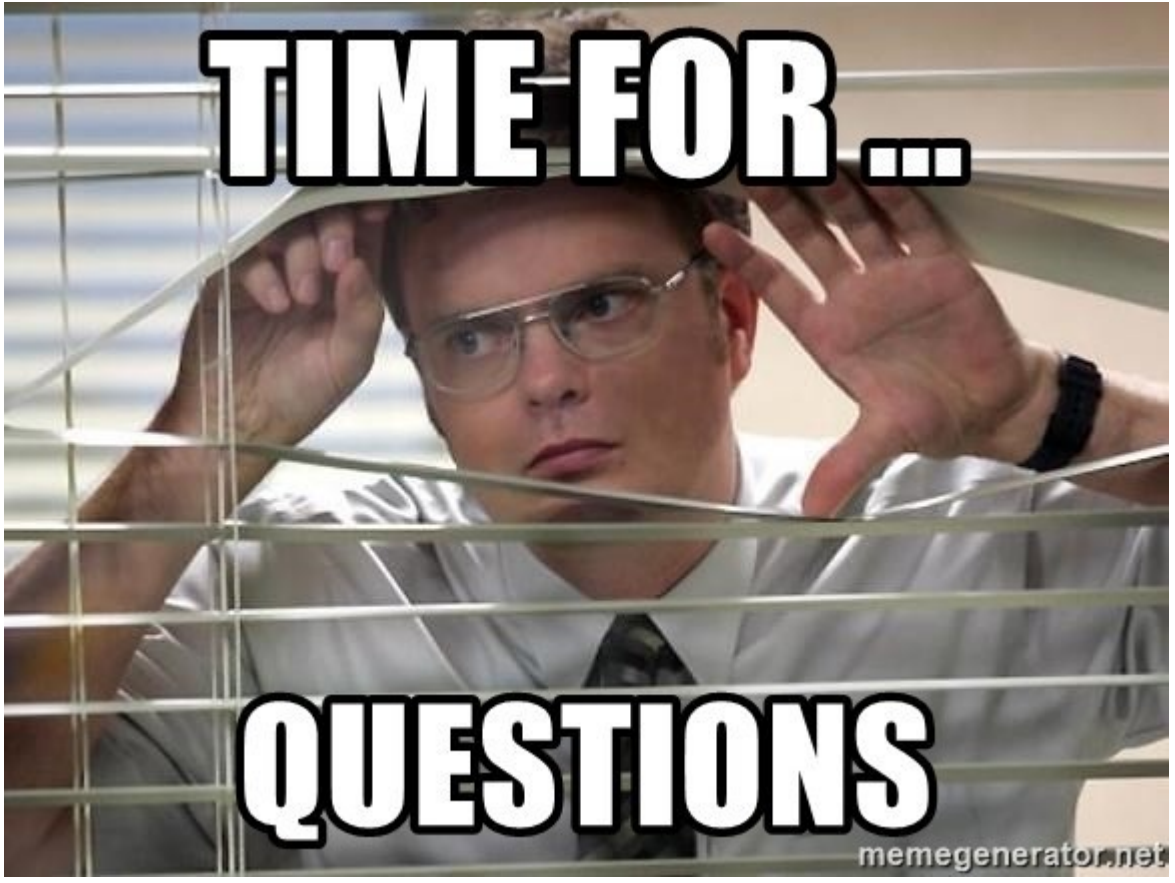
Пример (Г. Атанасов)

```
class Parent {
public:
  void print(){
    std::cout << "hi";
  }
};

class Child : public Parent {
public:
  void print() {
    Parent::print();
    std::cout << " how are you";
  }
};

int main() {
  Child child;
  child.print(); // -> hi how are you
  static_cast<Parent>(child).print(); // -> hi
  return 0;
}
```

Въпроси?



Задача 1 (Г. Атанасов)

Да се реализира клас `Person`, който представлява човек с име, възраст и имейл. Да се реализират следните методи:

- print - извежда подробна информация за човека;
- send_mail - по подадено съобщение (стринг) добавя съобщението в текстов файл с името на имейла на човека, като то трябва да започва със: `"To {name}: "`;
- view_inbox - изкарва на екрана съобщенията пратени на имейла на човека;

Да се реализира клас `Student`, който представлява студент в университет. Освен характеристиките на човека, студента да има още и факултетен номер. Да се предефинират методите `print` и `send_mail`, които съответно да извеждат подробна информация за студента и да пращат съобщение, което да започва със: `"To {name}, {faculty_number}:"`.

Да се реализира клас `Lecturer`, който представлява лектор в университет. Освен характеристиките на човека, лекторът да има още и име на катедра, от която е част. Да се предефинират методите `print` и `send_mail`, които съответно да извеждат подробна информация за лектора и да пращат съобщение, което да започва със: `"To {name}, part of the {department_name} department:"`.

Да се реализира клас `University`, който представлява университет, който има име, списък от студенти и списък от лектори. Да се реализират следните методи:

- add_student - добавя студент към университета;
- add_lecturer - добавя лектор към университета;
- remove_student - по име на студент го премахва от университета;
- remove_lecturer - по име на лектор го премахва от университета;
- print - изкарва подробна информация за университета и хората в него;
- send_mail_to_student - по подадено съобщение, изпраща имейл с това съобщение до всички студенти;
- send_mail_to_lecturers - по подадено съобщение, изпраща имейл с това съобщение до всички лектори;
- send_mail_to_all - по подадено съобщение, изпраща имейл с това съобщение до всички хора, които са част от университета;

Задача 2

- МОН иска да направи следната платформа за сравняване на университети. Има два типа студенти FMIStudent и UNSStudent Програмата трябва да смята средния успех на студентите във ФМИ и УНСС и да казва кои са с по-голям успех. Трябва да може да добавяте и премахвате студент и да променяте оценките им.

Задача 3

Реализирайте подходяща ООП йерархия за ферма. Фермата съдържа в себе си следните типове животни.

- Кон - различното на коня е, че може да се язди с функцията dqq(). Когато яздиш коня енергията му намалява
- Коза - нея можем да я доим с функцията milk() при доене енергията и спада
- Кокошка - нищо специално
- Куче - нищо специално
- Котка - нищо специално

Всяко животно може да издава звук с функцията sound() като за всяко тя ще изведе различно нещо на стандартния изход. Всяко животно може да се храни с функцията feed(string foodType) като за всяко животно ще са позволени само определени храни.