Доклад

По Увод в обектно- ориентураното програмиране (ООП)

Андрей-11а клас

1.ООП-определение

А) **Обектно ориентираното програмиране (*ООП*)** е парадигма в компютърното програмиране, при която една програмна система се моделира като набор от обекти, които взаимодействат помежду си, за разлика от традиционното виждане, в което една програма е списък от инструкции, които компютърът изпълнява. Всеки обект е способен да получава съобщения, обработва данни и праща съобщения на други обекти.

2.Клас.Дефиниране на класове

А) **Класът** е една от характеристиките на обектно ориентираното програмиране. Това е съвкупност от променливи и функции, които са обвързани в логическа структура и работят заедно. Класът служи като модел за представяне на реални обекти и софтуерни обекти, описвайки атрибути (свойства) и методи (поведение) на обектите.

3.Полета и свойства

А) Полета се дефинират в тялото на класа. Те са места, които са хранилища за данни. Там се пазят данните (стойностите), които отразяват състоянието на всяка конкретна инстанция на класа (обект) и са нужни за работата на други видове членове на класа – методи, конструктори, индексатори, свойства – моделиращи поведението на коя да е инстанция (т.е. на всички инстанции) на този клас. Съществуват и такива полета, наречени статични, които са общи за всички инстанции (обекти).

Б)Деклариране на полета

<modifiers> - незадължителна част от декларацията на едно поле. Означава както модификатори за достъп (нива на видимост), така и други модификатори.

<Type> -тип на полето. Той се указва задължително. Това може да бъде както примитивен тип (byte, short, char и т.н.) или масив, така и някакъв клас, т.е. може да бъде произволен .NET тип от CTS или тип, дефиниран от потребителя.

<Field\_name> - име на полето (според правилата за именуване на идентификатори в C#). представлява идентификатор, т. е. последователност от unicode символи – главни и малки букви, цифри, - (тире) и \_(подчертаващо тире), незапочваща с цифра или тире.

4.Конструктор

А) В обектно ориентираното програмиране **конструкторът** е блок от инструкции, който се изпълнява за инициализация на един обект при неговото създаване. Практически конструкторът може да извършва всякакъв вид действия, но ако не друго, то е поне признак за много лош стил на програмиране да се използва за действия, които не засягат пряко създавания обект, тъй като това може да доведе до неочаквани грешки, особено в големи програми. В езика за програмиране C++ конструкторът се представя като член-функция на един обект, носеща същото име като него.

Пример за дефиниран от програмиста конструктор в C#:

**class** **MyClass**

{

int x;

double y;

bool Initialized;

*// Когато бъде създаден нов обект от класа MyClass конструктура ще бъде извикан и ще инициализира обекта.*

**public** MyClass()

{

x = 5;

y = 0.5;

Initialized = **true**;

}

}

5.Методи

А) **Метод** в програмирането е съставна част от една програма, която решава конкретен проблем. Може както да приема параметри така и да връща стойност. Целта на писането на дадена програма е решаването на дадена задача. За да бъде ефективно решена дадена задача в програмирането, тя се разделя на подзадачи, разработват се алгоритми за решение на тези подзадачи и накрая тези подзадачи се сглобяват в цялостна програма. Обособените парчета код, решаващи дадената подзадача се наричат подпрограми (subroutines). В различните езици подпрограмите се срещат под други имена като функции (functions) или процедури (procedures). В C#, те се наричат методи (methods).

Най-простият пример за метод е "**Main(…)**", който винаги го декларираме между отварящата и затварящата скоба на нашия клас:

**class** **HelloCSharp**

{

*// Декларираме нашият метод между скобите на класа*

**static** **void** Main(string[] args)

{

Console.WriteLine("Hello C#!");

}

}