Изпитна тема № 3: Обектно-ориентирано програмиране

Клас, конструктор, полета, свойства, създаване на обекти от клас. Функции/методи в класовете, ключова дума this. Енкапсулация на данни в класовете, методи за достъп и промяна на полета (getters/setters). Статични полета и методи в класовете.

Клас (class) в ООП наричаме описание (спецификация) на съставен тип от данни и действия (behavior/functions), обвързани в логическа структура и работещи заедно. Класът представлява шаблон, който описва видовете състояния и поведението на конкретните обекти (екземплярите), които биват създавани от този клас. В класът се посочват данните (attributes) и кодът (behavior), с помощта на който се оперира с данните.

Конструкторът е специален метод на класа, който се извиква автоматично при създаване на обект от този клас и обикновено извършва инициализация на полетата му. Съществуват няколко типа конструктори. Всеки клас в C# има конструктор по подразбиране, който може да бъде предефиниран. По дефиниция той никога не приема аргументи. Конструкторът по подразбиране създава полетата на класа и ги инициализира със стойностите по подразбиране за съответния тип. Освен конструктор по подразбиране, който няма параметри има и така наречените параметризирани конструктори, които приемат аргументи, при създаването на нов обект.

Полета (fields) представляват променливи, декларирани в класа. В тях се пазят данни, които отразяват състоянието на обекта и са нужни за работата на методите на класа. Стойността, която се пази в полетата, отразява конкретното състояние на дадения обект, но съществуват и такива полета, наречени статични, които са общи за всички обекти.

Свойства (properties) – Основната цел на свойствата е да осигуряват капсулация на състоянието на класа, в който са декларирани, тоест да го защитят от попадане в невалидни данни. Свойствата позволяват да бъдат задавани стойности, но без да бъде разкривана имплементацията.

Създаването на обекти от предварително дефинирани класове става чрез оператора new. Новосъздаденият обект се присвоява на променлива от тип, съвпадащ с класа на обекта. При присвояване същинският обект не се копира, а в променливата се записва само референция към новосъздадения обект, неговият адрес в паметта.

Методите представляват именувани блокове програмен код. Те извършват действия и чрез тях реализира поведението на обектите от този клас. В методите се изпълняват алгоритмите и се обработват данните на обекта. Те дават възможност един код, да бъде изпълняван множество пъти, заради което помагат да се избегне повторение на код. За да се декларира един е метод е необходимо да бъдат описани: тип на връщаната стойност, име, списък с параметри и тяло. При необходимост могат да бъдат добавени и модификатори на достъп като public, private, protected или internal.

Ключовата дума this в C# дава достъп до референцията към текущия обект, когато се използва от метод в даден клас. Чрез нея можем да достъпваме елементите на съответния клас.

В следния пример е създаден клас, в който са включени полета, конструктори, свойства и методи. След това се демонстрира по какъв начин се декларира обект от създадения клас и как се извикват методите му. A picture containing text, screenshot, font

Description automatically generated

A screenshot of a computer program

Description automatically generated with low confidence

Капсулацията е един от основните принципи в обектно-ориентираното програмиране. Тя се използва за скриване на детайлите относно имплементация и даване на достъп само до необходимите елементи. Капсулацията позволява да определим кои методи и атрибути са достъпни за потребителите на класа. Има 4 основни нива на достъп: public, private, protected, internal. Използвайки модификатора public, ние указваме, че съответния елемент ще бъде достъпен за всеки друг клас, независимо дали е деклариран в същия проект. Модификатора private налага най-много ограничения. Това ниво на достъп позволява елементът да бъде достъпен единствено от класа, в който е дефиниран. Това е и видимостта по подразбиране на класовете. Нивото на достъп protected позволява елементите да бъдат достъпвани от класовете наследници на базовия клас. Чрез модификатора internal указваме, че елементите ще бъдат достъпни единствено за класовете в същия проект (assembly).

Методите за достъп и промяна на полета (getters/setters) се използват, за да бъдат зададени ограничения на това какви данни могат да попаднат в класа. Декларацията на метода за достъп (getter) се прави в тялото на свойството, а в тялото на метода се задават действия, които ще се извършат преди да върне резултат. Ако липсва тяло, се извършват присвояване и връщане на стойност по подразбиране. Методът трябва да завършва задължително с операция return или throw. Подобно на метода за четене, методът за промяна също се декларира в тялото на свойството. Той има тип на връщана стойност void, а подадената стойност се достъпва, чрез скрития параметър value. В примера е декларирано свойство, което съдържа методите A picture containing text, screenshot, font, line

Description automatically generatedза достъп и промяна.

Полетата или методите, които имат модификатор static в декларацията си се наричат статични. Това означава, че тези елементи принадлежат на самия клас, а не на конкретна инстанция. Когато поле или метод бъдат маркирани като статични, могат да се използват, без да се създава обект от съответния клас, ако полето им на видимост го позволява. При инициализиране на статични полета по време на декларация, инициализация се случва само веднъж, при създаването на първия обект от съответния клас. При работа със статични методи не може да се използва ключовата дума this, съответно статичните методи нямат достъп до останалите нестатични елементи в класа, понеже те не принадлежат към нито една инстанция. За да достъпим статичните атрибути на класа е необходимо да напишем името на класа, операторът . и името на статичния елемент.

Обобщен план (Опорни думи)

1. Клас, конструктор, полета, свойства, създаване на обекти от клас
   * Клас – описание (спецификация) на съставен тип от данни и действия. Шаблон, описва състояния и поведение. Данни (attributes), код (behavior).
   * Конструктор – Изпълнява се веднъж, при създаване на обект. По подразбиране, без параметри, параметризиран.
   * Полета – променливи в класа, пазят конкретно състояние на обекта.
   * Свойства – капсулация на състоянието на класа. Валидации.
   * Създаване на обекти от клас – ключова дума new, променлива от същия тип, запазва се референция.
2. Функции/методи в класовете, ключова дума this
   * Методи – наименуван блок от код, извършват действия в класовете, ограничават повторенията. Тип на връщана стойност, име, параметри, тяло, модификатор
   * This – референция към текущия обект
3. Пример – клас Person, полета \_name, \_age, конструктор с параметри, ключова дума this, метод Greet – отпечатва \_name и \_age.
4. Енкапсулация на данни в класовете, методи за достъп и промяна на полета (getters/setters).
   * Капсулация – скрива имплементация, нива на достъп, public, private, protected, internal.
   * Getter – в тялото на свойството, ако липсва тяло на getter, default стойност. Завършва с return или throw.
   * Setter - в тялото на свойството, ако липсва тяло на getter, default стойност. Скрит параметър value.
   * Пример – клас NightClubClient, полета \_name, \_age. Свойство Name default get/set. Свойство Age, проверка за пълнолетие.
5. Статични полета и методи в класовете
   * Всичко, което има static модификатор
   * Не принадлежи на инстанция
   * Не може да се използва this
   * Полетата се инициализират веднъж

Казус

Реализирайте програма на C#, в която е приложен клас, който трябва да има private полета:

* \_num1 и \_num2 (double тип)

Методи:

* Add(), метод, който връща сумата от две числа.
* Substract(), метод, който връща разликата от \_num1 и \_num2 (num2- num1).
* Multiply(), метод, който връща резултата от умножението на числа
* Divide(), метод, който връща резултата от делене на \_num2 на num1

Дефинирайте параметризиран конструктор, който приема два параметъра num1 и num2 и присвоява тези параметри на съответните полета в класа.

A screenshot of a computer program

Description automatically generated with medium confidence