Изпитна тема № 4: Обектно-ориентирано програмиране

Шаблонни класове и методи. Наследяване, абстракция и интерфейси. Полиморфизъм.

A picture containing text, screenshot, font

Description automatically generatedШаблонните типове (generics) се използват за прилагане на една функционалност към множество обекти от различни типове. Шаблонните типове могат да бъдат класове, структури, интерфейси и методи. При инстанцирането на класове и структури или извикването на методи ще бъдат използвани конкретни типове, които се предават като параметри. Създаването на шаблонен клас, е чрез добавянето на параметър към декларацията, който съответства на неизвестен тип. Едва при инстанциране този параметър ще бъде заместен с конкретен тип. Ако класът се нуждае от няколко различни неизвестни типа, те трябва да се изброят със запетая между знаците ‚<‘ и ‚>‘ в декларацията на класа, като всеки един от тях трябва да има различен идентификатор. При създаване на обект от типизиран клас е необходимо да посочим типа, с който той ще работи. В следния пример е деклариран шаблонен клас и съответно 2 обекта, които използват функционалностите на класа, но с различен тип.

A screenshot of a computer program

Description automatically generated with medium confidence

Наследяването, заедно с капсулацията, абстракцията и полиморфизма, е един от четирите основни принципа на обектно-ориентирано програмиране. Наследяването позволява да бъдат създавани класове, които преизползват и разширяват функционалностите на други класове. Класът, чиито елементи са наследявани се нарича базов (base) клас, а класът, който ги наследява се нарича производен или дъщерен (derived) клас. В C# производните класове могат да имат само един пряк базов клас. При декларация на клас-наследник, той наследява всички елементи на базовия клас с изключение на тези, които имат поле на видимост private, конструкторите и деструктора За да може един клас да наследи елементи, се добавя „:“, последвано от името на родителския клас.

Абстракция се нарича процеса по скриване на детайлите, относно имплементацията и това по какъв начин работи даден обект. Основната цел е показването само на необходимите функционалности, с които потребителят трябва да работи. Чрез абстракцията се намалява комплексността и се увеличават възможностите за надграждане. В C# тя се постига, чрез използването на абстрактни класове или интерфейси.

Абстрактните класове се декларират като се използва модификатора abstract пред ключовата дума class. Те не могат да бъдат инстанцирани, т.е. не може да бъде създаден обект от абстрактен клас. Тяхната цел е да предоставят обща дефиниция (характеристика) на базов клас, която да се използвана от множество производни класове. Освен абстрактни класове могат да бъдат декларирани и абстрактни методи. За да бъде един метод абстрактен трябва да се добави модификатора abstract преди типа на връщаната стойност. Абстрактните методи нямат имплементация, следователно тяхната дефиниция не завършва с тялото на метода, а с ‚;‘. Абстрактните класове могат да съдържат както абстрактни, така и обикновени методи, чиято имплементация също ще бъде наследена. Абстрактните методи трябва да бъдат имплементирани от класовете наследници на базовия клас. Когато един клас няма нито един имплементиран метод и нито една член променлива, той се нарича чист абстрактен клас (pure abstract class) В следния пример е създаден абстрактен клас и абстрактен метод, който е имплементиран в клас-наследник.

A screen shot of a computer code

Description automatically generated with low confidence

Подобно на абстрактните класове, интерфейсите се използват, за да се опише какво поведение (behavior) трябва да имат класовете, които го имплементират. Те се използват, когато много класове имат общо поведение, но не споделят общ базов клас. Интерфейсът не може да бъде инстанциран, а неговата функционалност трябва да бъде реализирана, чрез класовете, от които се имплементира. За разлика от абстрактните класове, интерфейсите не могат да съдържат елементи, които не са абстрактни. Един клас може да имплементира повече от един интерфейс, за разлика от наследяването при абстрактните класове. Това позволява гъвкавост при дефиниция на поведението на класовете и се определя като едно от основните предимства на работата с интерфейси. За деклариране се използва ключовата дума interface, последвана от идентификатор (име). Прието първата буква от името на интерфейсите да бъде „I“ (главно и). Синтаксиса за имплементиране на интерфейс е същия както при наследяването на класове – „:“ последвани от името на интерфейса.

Полиморфизмът е концепция в обектно-ориентираното програмиране, която се изразява в това една променлива, функция или обект да приемат множество форми, в зависимост от това по какъв начин се използват. В C# има 2 основни вида полиморфизъм. По време не компилация и по време на изпълнение. Полиморфизмът по време на компилация се реализира, чрез презареждане (overload) на методи или оператори. В класовете могат да бъдат създавани методи, които имат едни и същи имена, но са с различна сигнатура (signature), т.е. имат различен брой и тип на техните параметри. По този начин един метод, може да изпълнява различни действия, спрямо това какви аргументи са му подадени. Полиморфизмът по време на изпълнение се реализира, чрез презаписване (override) на методи. За целта на даден метод в базовия клас се добавя модификатора virtual, който указва, че класовете наследници, могат да добавят собствена имплементация на съответния метод. За да може метод от производния клас да презапише метод от базовия клас, той трябва да бъде добавен модификатора override преди типа на връщана стойност и да има същото име, същия брой и тип параметри и същия тип на връщана стойност. Благодарение на презаписването на методи един метод може да има различно поведение (behavior), спрямо това в кой клас се намира. В следния пример е деклариран базов клас Singer, който е наследен от 2 класа Galena и Preslava. Методът Sing от класа Singer е презаписан в класовете наследници, което демонстрира полиморфизъм по време не изпълнение на програмата.

A screenshot of a computer program

Description automatically generated with low confidence

A picture containing text, screenshot, font, number

Description automatically generated

Обобщен план (Опорни думи)

1. **Шаблонни класове и методи**
   * Една функционалност, множество обекти от различни типове.
   * Класове, структури, интерфейси, методи
   * Предаване на тип/типове като параметри
   * Пример- клас MailBox, \_senderName – string, \_content -T. Конструктор – set \_senderName, Методи, PrintMail – typeOf(T), \_content. UploadMail(T mail): \_content = mail;
2. **Наследяване, абстракция и интерфейси**
   * Наследяване - Класове, преизползват и разширяват функ. на други класове. Базов клас, Производен клас. Само един базов клас. Наследява се всичко без private, конструктор, деструктор.
   * Абстракция – скриване на детайли за имплементация, показване само на нужното. Намалява комплексността и възможности за надграждане.
   * Абстрактни класове- предоставят обща характеристика на поведението (behavior) на даден клас. Ключова дума abstract. Не могат да се инстанцират. Могат да имат неабстрактни елементи. Pure abstract class. Абстрактни методи – модификатор abstract, без тяло, трябва да се имплементират от класовете наследници.
   * Интерфейси – описват поведение, използват се с класове с общо поведение, които нямат общ базов клас. Не могат да се инстанцират. Не могат да имат неабстрактни елементи. Един клас може да имплементира много интерфейси. “I” за първа буква
3. **Полиморфизъм**
   * Един обект, много форми, в зависимост от начина на ползване
   * По време на компилация – презареждане (overload) на методи и оператори. Еднакви имена, различна сигнатура (брой и тип параметри).
   * По време на изпълнение – презаписване (override) на методи. Ключови думи virtual и override. Класа наследник може да създаде своя имплементация
   * Пример – клас Singer, виртуален метод Sing, презаписан от класове наследници Galena и Preslava.

Казус

Реализирайте програма на C#, в която, вече сме имплементирали клас Book, който има абстрактен метод GetDetails(), параметризиран конструктор и три частни полета със съответните им свойства с дефинирани get аксесори. Напишете клас MyBook, който наследява класа Book и има параметризиран конструктор приемаш параметри string titile, string author и string price. Приложете метода GetDetails на класа Book в класа MyBook, така че да връща подробностите за MyBook.

A picture containing text, screenshot, font, number

Description automatically generated

A screenshot of a computer program

Description automatically generated with low confidence