C#- Metoder

Metoder/funktioner

Namnet för metoder/funktioner används omväxlande, men båda betyder samma sak. Metoder används när man behöver gruppera kod. Behovet uppstår när man vill skapa en metod som har en specifik uppgift eller när man ser att man har upprepande kod. Med upprepande kod menar jag att vi exempelvis har ett program där vi ofta behöver addera två tal. Istället för att skriva ”tal1 + tal2” överallt i vår klass kan vi därför välja att skapa en metod och skicka med två tal som argument.

Uppbyggnad

En metod består av en accesstyp, en identifierare om metoden är statisk eller icke-statisk(mer om detta när vi arbetar med objekt), returtyp och ett metodnamn följt av parenteser där vi kan passera argument om så behövs.

Exempel:
```
public static void myMethod(string text)
{
    Console.WriteLine(text);
{
```

I exemplet har vi skapat en metod med accesstyp *public* och returvärde *void* som tar in en sträng som argument. *Public* betyder att denna metod kan användas utanför klassen och detta går vi in på mer när vi kommer arbeta med objekt, så tills vidare behöver vi bara komma ihåg att använda *public* och *static* när vi skapar våra metoder. Vi använder returtypen ”*void*” här, detta betyder att metoden inte ska returnera något utan att koden inuti bara ska köras. Metoden fick namnet myMethod och som inparameter har vi en sträng som vi döpte till ”text”. Denna variabel blir en lokal variabel och kan endast användas inuti metoden som vi gör här när vi skriver ut variabeln ”text” till vår konsol.

Anropa metoder

För att anropa en metod behöver man skriva metodnamnet samt ge indata om det behövs.

Exempel:
```
myMethod(”Min första metod!”);
```

Här anropar vi vår metod och skickar med en sträng.

Returvärden

Som tidigare nämnt kan metoder ha flera olika returtyper. Dessa typer kan vara allt ifrån vanliga variabler till objekt vi skapat själva. När vi definierar en returtyp för en metod så säger vi till metoden ”kör denna kod, och returnera specificerad datatyp”. Viktigt att tänka på är att när vi sätter en returtyp så måste metoden returnera den datatypen, annars får vi syntaxerror. Låt oss säga att vi bygger en miniräknare och ska skapa en metod för addition. Vi väljer att jobba med heltal så vi vill använda en *int* som returtyp eftersom vi vill att metoden ska returnera summan av de två heltalen.

Exempel:
```
public static int adderaTal(int tal1, int tal2)
{
        return tal1 + tal2;
{
```

Här har vi skapat en metod för simpel addition. Här har vi specificerat att returtypen ska vara en integer och argumenten som två heltal vi utför vår addition på. Eftersom denna metod returnerar något så kan vi inte anropa den som vi gjorde med vår förra metod som hade returtypen *void*, utan måste antingen lagra värdet eller utföra någon form av operation på den direkt.

Exempel:
```
int summa = adderaTal(5, 2);
Console.WriteLine(adderaTal(4, 3));
```

Första raden sparar vi summan i en variabel, och på den andra raden skriver vi ut summan direkt i vår konsol.

Metoder – slutord

Exemplen vi gått igenom är väldigt simpla och alla metoder innehåller inte bara en rad kod. Vi använder metoder för att gruppera kod som används ofta och för att skapa funktioner. Det är väldigt viktigt att tänka på att en metod bara ska innehålla kod för den uppgift den ska utföra. Detta gör det enklare för dig som utvecklare att hålla reda på vad din kod gör och gör din kod med ”städad”.

Det är bättre att dela upp en metod till flera metoder om det krävs. Exempel för detta är om vi har en metod som ska ta in ett värde av en användare, använda en loop och skapa en multiplikationstabell och sedan skriva ut en formaterad text på tabellen. Då blir det bättre att dela upp detta i fler metoder. En metod som tar ett värde av användaren, en som använder värdet och skapar multiplikationstabellen och en metod som formaterar texten och gör en snygg tabell och returnerar detta.