C#- Klasser och objekt

Klasser vs Objekt

Klasser och objekt är grundstommen i objektorienterad programmering. Vad är då en klass och vad är ett objekt?

Klass

En klass kan ses som en byggritning för ett objekt. Det vill säga att klassen i sig har ingen effekt på programmet vi skapar förens vi skapar ett objekt av klassen. Vad ska då en klass innehålla? Det alla klasser måste innehålla är för det första en konstruktor som agerar som ingångspunkt när vi skapar vårt objekt. Denna konstruktor säger till vår klass att vi vill konstruera ett objekt av klassen. Om vi skapar en klass som kräver data att arbeta på utanför klassen så skriver man som argument i konstruktorn vilka indata den behöver. Efter att vi deklarerat vår konstruktor och dess indata(om behovet finns) så kan vi fylla på med den klasspecifika kod vi behöver det vill säga variabler och funktioner och så vidare.

Objekt

När vi har skapat vår klass kan vi nu skapa ett objekt från denna. Detta gör vi genom att skapa en variabel av klassen. Att skapa ett objekt ser ungefär likadant ut som att skapa en *array*, det vill säga att vi först definierar datatypen som nu blir vårt klassnamn, sedan variabelnamn följt av tilldelning som följs av nyckelordet *new* samt klassnamn med parentes.
Exempel: Klassnamn variabelNamn = new Klassnamn(indata vid behov skrivs här);

När vi skapat vårt objekt kan vi sedan använda oss av dess metoder/funktioner och utföra de uppgifter vi tänkt för denna. Det finns ingen gräns till hur många objekt vi kan skapa av samma klass, då klassen bara är en ritning för ett objekt.

Objektorienterad programmering

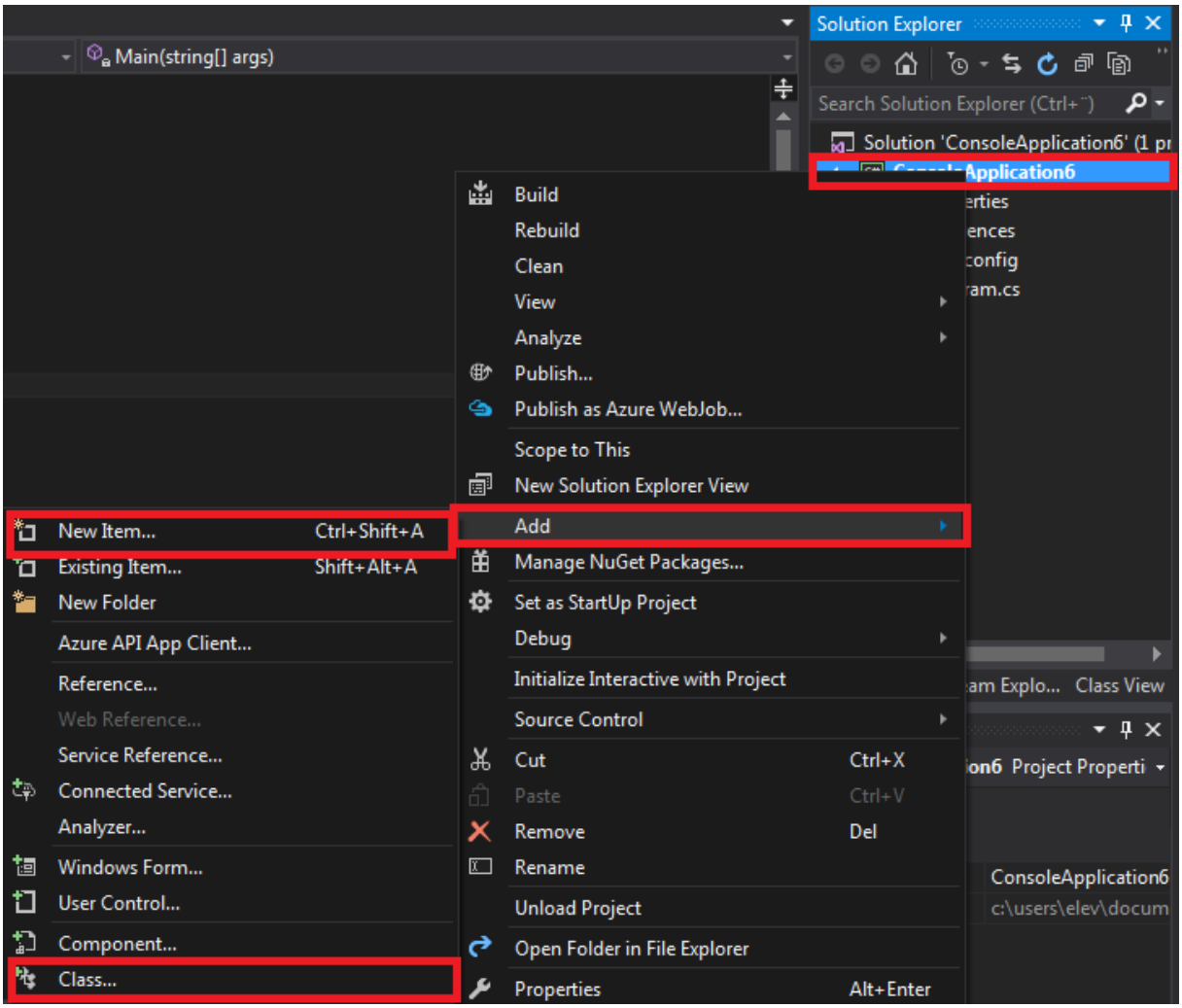
Objektorienterad programmering handlar om att skapa applikationer genom att använda objekt. Denna typ av programmering gör koden mer lättläst och enklare att orientera och är en standard inom programmering. Detta tar oss från att skriva all kod i samma klass till att skapa klasser med specifika uppgifter och funktioner som behövs för ett fungerande program.

Från och med nu kommer vi skapa klasser och objekt för att utföra alla uppgifter för våra program istället för att skriva all kod i ingångsklassen. I ingångsklassen kommer vi bara skriva den kod som behövs för att anropa våra objekt.

I kommande exempel skapar vi en klass vi kallar för Recept. Denna klass kommer alltså att vara en byggritning för hur vi skapar ett simpelt recept.

Hur vi skapar en klass

En klass skapar vi genom att högerklicka på projektnamnet, gå ner till add och sedan trycker på new item och letar efter class, eller trycker på class direkt i menyn.
Mitt tips är att använda sig av Ctrl + Shift + A och välja class i listan.



Klass Recept

Ett recept består av vilka ingredienser som behövs och instruktioner man bör följa för att nå resultat. Vi börjar med att skapa en sträng för receptets namn, en sträng-array för ingredienser och en sträng-array för instruktioner. I klassen ska dessa vara tomma eftersom när vi skapar objekt av klassen så ska dessa definieras av skaparen. Eftersom klassen kommer behöva namn, ingredienser och instruktioner för att bli ett objekt bör vi därför inkludera dessa som indata i konstruktorn.

```
class Recept
{
    // Klasspecifika variabler
    private string name;
    private string[] ingredients;
    private string[] instructions;

    // Konstruktor med indata
    public Recept(string name, string[] ingredients, string[] instructions)
    {
        // Nyckelordet "this" används för att referera till klassens variabel
        // då klassens och indatans namn är likadan
        this.name = name;
        this.ingredients = ingredients;
        this.instructions = instructions;
    }
}
```

Nu har vi skapat vår klass och inkluderat privata klasspecifika variabler, en konstruktor med indata och inuti konstruktorn har vi tilldelat värdena till våra klasspecifika variabler och kan nu användas i klassen.

Det vi behöver göra nu är att skapa funktioner till klassen så att den blir användbar. Det vi vill skapa är ett snyggt recept och skicka tillbaka som ett strängvärde till den klass som använder ett recept-objekt. Vi börjar med att skapa tre metoder.

Den första heter **printRecipe()**, den har accesstyp public då vi vill att den ska kunna användas utanför klassen, returtyp är sträng och den ansvarar för att skapa ett formaterat recept och returnera receptet.

Den andra heter **listIngredients()**, accesstyp är private då endast recept-klassen behöver använda den och den returnerar en sträng. Denna används för att lista upp ingredienserna.

Den tredje och sista heter **makeInstructions()**, accesstyp private av samma anledning som **listIngredients()** och den returnerar en sträng. Denna ansvarar för att skapa instruktionerna.

I dessa lägger vi in en lokal strängvariabel och returnerar den strängen.

Koden bör nu se ut så här:

```
        // Metod med accesstyp public och returvärde string
        // ansvarar för skapandet av recept samt returneringen av
        // receptet till den som anropar metoden.
        public string printRecipe()
        {
            string recipe = "";

            return recipe;
        }

        // Metod med accesstyp private och returvärde string
        // ansvarar för att skapa en lista av ingredientserna i arrayen.
        private string listIngredients()
        {
            string listedIngredients = "";

            return listedIngredients;
        }

        // Metod med accesstyp private och returvärde string
        // ansvarar för att formatera instruktionerna till ett
        // snyggare format.
        private string makeInstructions()
        {
            string formattedInstructions = "";

            return formattedInstructions;
        }

    } // class
} // namespace
```

Metoden **printRecipe()** kommer använda sig av de andra två metoderna så vi väntar med att skapa ny kod till den och går direkt till **listIngredients()**.

listIngredients()

I **listIngredients()** vill vi använda oss av en *foreach* för att iterera genom vår *ingrediensarray* och lägga in värdet i vår lokala variabel *listedIngredients*. Innan vi implementerar vår loop så adderar vi till vår sträng med texten ”För detta recept behöver du:\n” där ”\n” betyder ny rad. Nu implementerar vi vår foreach-loop och för varje ingrediens lägger vi till det element i *listedIngredients*. Efter vår loop lägger vi till en ny rad bara för att separera innehållet i strängen och sedan returnerar vi listedIngredients.

Nu har vi implementerat metoden för att lista ingredienser och koden bör se ut så här:

```
// Metod med accesstyp private och returvärde string
// ansvarar för att skapa en lista av ingredientserna i arrayen.
private string listIngredients()
{
    string listedIngredients = "";

    listedIngredients = "För detta recept behöver du:\n";

    // Itererar igenom varje sträng variabel i ingredientsarrayen
    // och lägger till varje element i listedIngredients
    // följt av en ny rad. "\n" skapar ny rad.
    foreach (string ingredient in ingredients)
    {
        listedIngredients += ingredient + "\n";
    }

    listedIngredients += "\n";

    return listedIngredients;
}
```

makeInstructions()

Vår funktion **makeInstructions()** kommer se nästan identisk ut som **listIngredients()** och använder sig av en *foreach* för att tilldela till sträng.
Koden bör därför se ut såhär:

```
// Metod med accesstyp private och returvärde string
// ansvarar för att formatera instruktionerna till ett
// snyggare format.
private string makeInstructions()
{
    string formattedInstructions = "";

    // Itererar igenom varje instruktion(element) i instructions
    // och lagrar den följt av ny rad("\n") i formattedInstructions
    foreach (string instruction in instructions)
    {
        formattedInstructions += instruction + "\n";
    }

    return formattedInstructions;
}
```

printRecipe()

Sista metoden **printRecipe()** är den enklaste att implementera av dem alla. Den använder sig av de två metoderna vi nyss skapat och metodens uppgift är endast att lägga ihop dessa strängar och returnera värdet. I metoden börjar vi med att skapa en titel genom att lägga till strängen ”Hur man bakar: ” och därefter inkluderar vi vår namn-variabel för receptet samt två nya rader. Sedan anropar vi våra metoder **listIngredients()** och **makeInstructions()** och lägger till dem i vår sträng. Därefter returnerar vi strängen.

```
// Metod med accesstyp public och returvärde string
// ansvarar för skapandet av recept samt returneringen av
// receptet till den som anropar metoden.
public string printRecipe()
{
    string recipe = "";

    // Lägg till en simpel titel
    recipe = "Hur man bakar: " + name + "\n\n";
    // Aropa och lägg till returneringsvärden
    // från våra metoder in i recipe.
    recipe += listIngredients();
    recipe += makeInstructions();

    return recipe;
}
```

Då är vår recept klass klar och nu kan vi skapa objekt av denna!

Skapa objekt utav Recept.cs

Först börjar vi med att skapa tre variabler för namn, ingredienser och instruktioner. Vi döper receptet till Chokladkaka och skapar sedan två sträng *arrayer* och fyller dessa med strängar. Nu behöver vi skapa objektet, detta gör vi genom att specificera vilken datatyp objektet är, och i detta fall blir det *Recept*, följt av namn samt tilldelningen med nyckelordet *new* och Recept(våra argument).
Kod: **Recept** kaka = new **Recept**(namn, kakIngredienser, kakInstruktioner);

Nu kan vi fritt anropa vår metod **printRecipe()** genom att skriva namnet på vårt objekt följt med ”.” och metodnamn. Eftersom metoden returnerar en sträng och vi vill skriva ut receptet till konsolen så skriver skickar vi med metoden som argument i en *WriteLine*.
Kod: Console.WriteLine(kaka.printRecipe());

```
class Program
{
    static void Main(string[] args)
    {
        // Namn, ingredienser och instruktioner
        string kakNamn = "Chokladkaka";
        string[] kakIngredienser = new string[]
        { "1dl mjöl", "1msk kakao", "2 ägg", "1dl socker", "50g smält smör"};

        string[] kakInstruktioner = new string[]
        { "Vispa ihop socker och ägg.", "Blanda ner mjölet, kakaon och det smälta smöret",
        "Häll ner i en form och in i ugnen på 225 grader i 30 minuter", "Njut"};

        // Skapa recept objekt med namnet kaka och
        // skicka med argument
        Recept kaka = new Recept(kakNamn, kakIngredienser, kakInstruktioner);

        // Skriv ut receptet genom att skicka med metoden
        // printRecipe() från objekt kaka som argument
        Console.WriteLine(kaka.printRecipe());

        Console.ReadLine();
    } // main
} // class
```

Output:

```
Hur man bakar: Chokladkaka

För detta recept behöver du:
1dl mjöl
1msk kakao
2 ägg
1dl socker
50g smält smör

Vispa ihop socker och ägg.
Blanda ner mjölet, kakaon och det smälta smöret
Häll ner i en form och in i ugnen på 225 grader i 30 minuter
Njut
```

Då har vi skapat vår första klass och skapat ett objekt av denna! Testa att skapa ett nytt objekt av klassen Recept och skriv ut till konsol.

Notering: Jag har ingen aning om hur man bakar.