Översikt

- Implementera Structs och Enums.
- Organisera information i samlingar.

.eosec

Lektion 1: Implementera Structs och Enums

- Skapa och använda Enums.
- Skapa Structs.
- Access Modifiers.
- Använda Structs.
- Property.
- Skapa egenskap.
- Kalla på metod.
- Felsöka metoder.

.eosec

Skapa och använda Enums

- Enumeration type eller enums, ger möjlighet att skapa en variabel med ett antal fixerade värden.

```
enum Dag { Måndag, Tisdag, Onsdag, Torsdag, Fredag, Lördag, Söndag
);
```

- För att arbeta med enums skapar du instans för din variabel och specificerar vilken du vill använda.

```
Dag favoritDag = Dag.Fredag;
```

- Varje medlem i enum har ett namn och ett värde.

Skapa och använda Enums (forts.)

- Namn är en string, värdet är som standard integer.
- Sätts inget värde, kommer standardvärde att användas, börjar med 0 och ökas inkrementellt.
- Dag.Måndag är lika med 0 och Dag.Tisdag är lika med 1.

```
// Sätt enum variabel med namn
Dag favoritDag = Dag.Fredag;
// Sätt enum variabel med värde
Dag favoritDag = (dag)4;
```


Skapa och använda Structs

- Structs används för att skapa skräddarsydda typer.
- Structs representerar relaterade poster i en logisk enhet.
- De flesta inbyggda typerna i Visual C#, exempelvis int, bool och char, är definierade genom structs.

Skapa och använda Structs (forts.)

- Structs definieras med nyckelordet struct:

```
public struct Coffee
{
      public int Strength;
      public string Bean;
      public string CountryOfOrigin;
      // metoder etc
      ...
}
```

- Kan innehålla fält, egenskaper, metoder och händelser.

.easec

Access Modifiers

- Access Modifiers definierar var structs kan användas:

Access Modifiers	Beskrivning
public	Typ är tillgänglig för programkod i hela sammansättningen.
internal	Typ är tillgänglig bara i applikation.
private	Typ är bara tillgänglig inom struct

.easec

Använda structs

```
Coffee.coffee1 = new Coffee();
coffee1.Strength = 3;
coffee1.Bean = ”Arabica”;
coffee1.CountryOfOrigin = ”Kenya”;
```

- Syntax för att instansiera en struct, new Coffee(), liknar syntax för att kalla på metod.
- När du instansierar en struct, kallar du på en speciell typ av metod, en constructor.
- Constructor är en metod inom struct, med samma namn som struct.
- När instansiering görs utan argument, använder kompilator default constructor.

.easec

Använda structs (forts.)

- Om du vill kunna definiera standardvärden, när du instaniserar, får du lägga till constructor som accepterar dessa värden.
- För att tillfredställa olika behov, kan flera constructors användas.

.eosec

Egen constructor -exempel

```
public struct Coffee
{
        // Detta är skräddarsydd constructor
        public Coffee(int strength, string bean, string
countryOfOrigin)
                this.Strength = strength;
                this Bean = bean;
                this.countryOfOrigin = countryOfOrigin;
        }
        public int Strength;
        public string Bean;
        public string CountryOfOrigin;
        // metoder etc
        ...
}
```

- Kalla på constructor:

```
Coffee coffeel = new Coffee(4, ”Arabic”, ”Columbia”);
```

.eosec

Property

- Property är en construct som tillåter att hämta eller sätta värden för privata fält, inuti struct eller klass.
- För konsumerare av struct eller klass, kommer property ha beteendet som ett publikt fält.
- Inuti struct eller klass, implementeras property genom accessors, en speciell typ av metoder.
- Property kan inkludera en eller båda av följande:
  - get, för att få läsrättighet till ett fält.
  - set, för att få skrivrättighet till ett fält.

.eosec

Implementera property

```
public struct Coffee
{
        private int strength;
        public in Strength
        {
                get { return strength; }
                set { strength = value; }
        }
}
```

- get använder nyckelordet return för att returnera värdet av det privata fältet till anroparen.
- Set använder speciell lokal variabel, value, för att sätta värde för det privata fältet.

.eosec

Skapa indexer

- I vissa scenario vill du använda struct eller klass som behållare för area av värden.
- Skapa struct som inkluderar array:

```
public Struct Menu
{
        public string[] beverages;
        public Menu[string bev1, string bev2)
        {
                beverages = new string[] { ”Americano”, ”Café au Lait”, ”Café Macchiato”, ”Cappuccino”, ”Expresso” };
        }
}
```

.eosec

Skapa indexer (forts.)

- Att få åtkomst till area:

```
public Struct Menu
{
        public string[] beverages;
        public Menu[string bev1, string bev2)
        {
                beverages = new string[] { ”Americano”, ”Café au Lait”, ”Café Macchiato”, ”Cappuccino”, ”Expresso” };
        }
}
```

- När area exponeras som publika fält använder du följande syntax för att hämta:

```
Menu myMenu = new Menu();
String firstDrink = myMeny.beverages[0];
```

.eosec

Skapa indexer (forts.)

- Om indexer skapas, kan du använda myMenu[0] för att få första post från menu.
- Indexer liknar property, genom att den använder get och set för att kontroller tillgång till fält.
- Viktigaste är att indexer ger möjlighet till direkt tillgång till medlemmar i en samling, med namn för hållare, antingen struct eller klass genom ett integer värde.
- För att deklarera indexer, används nyckelordet this.

.easec

Skapa indexer -exempel

```
public Struct Menu
{
        private string[] beverages;
        // Detta är indexer
        public string this[int index]
        {
                get { return.this.beverages[index]; }
                set { this.beverages[index] = value; }
        }
        // Ger möjlighet för klient att räkna ut storlek på samling
        public int Length
        {
                get { return beverages.Length;
        }
}
```

.easec

Skapa indexer –exempel (forts.)

```
public Struct Menu
{
        private string[] beverages;
        // Detta är indexer
        public string this[int index]
        {
                get { return.this.beverages[index]; }
                set { this.beverages[index] = value; }
        }
        // Ger möjlighet för klient att räkna ut storlek på samling
        public int Length
        {
                get { return beverages.Length;
        }
}
```

.easec

Skapa indexer –exempel (forts.)

- Att få tillgång till poster i area genom att använda indexer:

```
Menu myMenu = new Menu();
string firstDrink = myMenu[0];
int numberOfChoises = myMenu.Lenght;
```

.easec

Övning Skapa och använda struct och enum

.easec

Lektion 2: Organisera information i samlingar

- Varför samlingar?
- Typer av samlingar.
- Välja typ av samling.
- Standardklasser för samlingar.
- Specialklasser för samlingar.
- List samlingar.
- Dictionary klass.
- Ställa fråga till samling.

.easec

Varför samlingar?

- När du skapar flera enheter av samma typ, behöver du ett sätt att hantera dessa tillsammans, samling används för detta.
- Samlingar är viktiga för grafiska applikationer. Kontroller som drop-down meny:er och andra typer aav menyer använder sig av samlingar.

.easec

Typer av samlingar

- Alla typer av samlingar delar gemensamma karaktärsdrag.
- För att hantera samling av poster, måste du kunna:
  - Lägga till poster i samling.
  - Ta bort poster från samling.
  - Hämta specifik post från samling.
  - Räkna antalet poster i samling.
  - Läsa igenom samling efter post.
- Alla klasser för samlingar i C# har metoder för att hantera dessa saker.

.easec

Välja typ av samling

- *List* klass, lagrar linjär samling av poster.
- *Dictionary* klass, lagrar samling med nyckel/värde par.
- *Queue* klass, lagrar poster i en först in, först ut samling.
- *Stock* klass, lagrar poster i en sist in, sist ut samling.
- Standardklasser – System.Collections.
- Specialklasser – System.Collections.Spezialized.

.easec

Standardklasser för samlingar

Klass	Beskrivning
ArrayList	- Samling för generellt syfte - Linjär samling av objekt
BitArray	- Samling av Booleanvärde - Funktioner för AND, NOT eller XOR
Hashtable*	- Samling för generellt syfte - Inkluderar metoder och egenskaper som gör det möjligt att hämta post genom nyckel, ta bort post och för kontroll om specifik post finns i samling
Queue	- Först in, först ut samling
SortedList*	- Samling sorterad på nyckel - Hämta post genom nyckel eller index
Stock	- Sist in, sist ut

* Av typen dictionary

.eosec

Standardklasser för samlingar (forts.)

ArrayList

```
C:\Windows\system32\cmd.exe
Bean type: Arabica
Country of origin: Columbia
Strength (1-5): 4
Bean type: Arabica
Country of origin: Vietnam
Strength (1-5): 3
Bean type: Robusta
Country of origin: Indonesia
Strength (1-5): 4
Press any key to continue . . .
```

```
static void Main(string[] args)
{

    // Skapa nya ArrayList collection.
    ArrayList beverages = new ArrayList();
    // Skapa pekare för att lägga till collection.
    Coffee coffee1 = new Coffee(4, "Arabica", "Columbia");
    Coffee coffee2 = new Coffee(3, "Arabica", "Vietnam");
    Coffee coffee3 = new Coffee(4, "Robusta", "Indonesia");
    // Lägg till objekt
    // Pekar konverteras till Object type när de läggs till dessa.
    beverages.Add(coffee1);
    beverages.Add(coffee2);
    beverages.Add(coffee3);
    // Hämta konverteras (cast) tillbaks till original typ.
    Coffee firstCoffee = (Coffee)beverages[0];
    Coffee secondCoffee = (Coffee)beverages[1];

    foreach (Coffee coffee in beverages)
    {
        Console.WriteLine("Bean type: {0}", coffee.Bean);
        Console.WriteLine("Country of origin: {0}", coffee.CountryOfOrigin);
        Console.WriteLine("Strength (1-5): {0}", coffee.Strength);
    }
}
```

Hashtable

```
C:\Windows\system32\cmd.exe
The ingredients of a Irish Coffee are: Coffee, Whiskey, Cream, Sugar
Press any key to continue . . .
```

```
static void Main(string[] args)
{
    // Skapa ny Hashtable collection.
    Hashtable ingredients = new Hashtable();
    // Lägg till par bestående av nyckel/värde till samling.
    ingredients.Add("Irish Coffee", "Coffee, Whiskey, Cream, Sugar");
    ingredients.Add("Café Mocha", "Coffee, Milk, Chocolate");
    ingredients.Add("Cappucino", "Coffee, Milk, Foam");
    ingredients.Add("Irish Coffee", "Coffee, Whiskey, Cream, Sugar");
    ingredients.Add("Macchiato", "Coffee, Milk, Foam");
    // Kontrollera om nyckel finns.
    if (ingredients.ContainsKey("Irish Coffee"))
    {
        // Hämta värde som är kopplat till nyckel.
        Console.WriteLine("The ingredients of a Irish Coffee are: {0}", ingredients["Irish Coffee"]);
    }
}
```

.eosec

Specialklasser för samlingar

Klass	Beskrivning
ListDictionary	- Av typen dictionary - Optimerad för små samlingar (<10)
HybridDictionary	- Av typen dictionary - Implementeras som ListDictionary för små samlingar (<10), som Hashtable om samling blir större
OrderedDictionary	- Osorterad samling av typen dictionary - Hämtar post med index eller nyckel
NameValueCollection	- Av typen dictionary - Lista där både nyckel och värde är string - Hämtar post med index eller nyckel
StringDictionary	- Samling där alla poster är string
BitVector32	- Struktur fixerad med storlek - Representerar värden som Boolean eller integer

.eosec

List samlingar

- Vanligaste typen är ArrayList.
- Lagrar poster linjärt.
- Objekt av vilken typ som helst kan läggas till, men ArrayList representerar varje post som System.Object instanser.
- När du lägger till post, kommer ArrayList konvertera dessa till Objekt.
- När poster hämtas, måste du konvertera dessa till originaltyp.

.eosec

List samlingar (forts.)

- Lägga till objekt:

```
Coffee coffee1 = new Coffee(4, ”Arabica”, ”Columbia”);
ArrayList beverages = new ArrayList();
Beverages.Add(coffee1);
```

- Hämta med index:

```
Coffee firstCoffee = (Coffee)beverages[0];
```

- Foreach loop för att läsa igenom:

```
foreach(Coffee c in beverages)
{
        Console.WriteLine(c.CountryOfOrigin);
}
```

.eosec

Dictionary klass

- Dictionary klass lagrar samlingar som ett pat av nyckel/värde.
- Vanligaste typen är Hashtable.
- När värde lagras i Hashtable, måste nyckel och ett värde definieras.
- Innehåller två samlingar, en för nyckel och en för värde.

.eosec

Ställa fråga till samling

- LINQ är en frågeteknik som är inbyggt i .NET.
- Frågetekniken bygger syntax som är standardiserad.
- Ett antal källor kan användas:
  - .NET samlingar.
  - SQL Server.
  - ADO.NET datasets.
  - XML-dokument.

.easec

Ställa fråga till samling (forts.)

- Syntax:

```
from <namn_på_variabel> in <datakälla>
where <kriteria>
orderby <kriteria_för_att_sortera_resultatet>
select <namn_på_variabel>
```

.easec

Övning Arbeta med samlingar

.easec

Repetitionsfrågor