OOP – Object Oriented Programming

OOP – ObjektOrienterad Programmering

- OOP Handlar om att dela upp ett program efter funktion och syfte
- Att dela upp kod underlättar återbruk av funktionalitet
- Återbruk funktionalitet minskar risken för buggar
- ”DRY” – Don’t Repeat Yourself
- Genom att skriva kod objektorienterat ger en renare struktur och underlättar utvecklingen

Klasser och Objekt

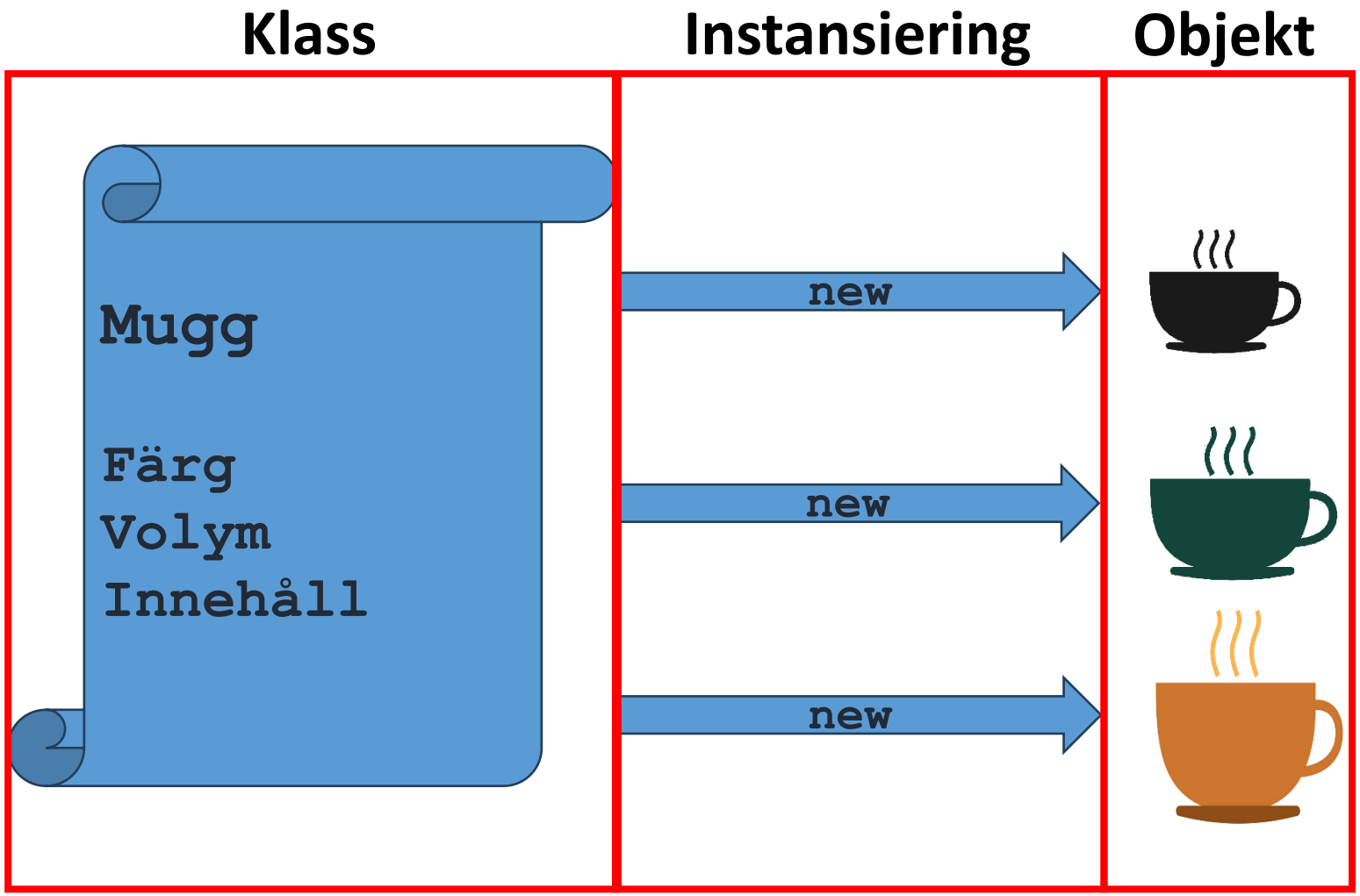
- En Klass kan liknas vid en ritning för ett objekt
- Ett objekt är en instans utav en Klass
- Ett objekt har en Typ – vilken klass den är en instans av
- Man kan skapa flera instanser av samma klass
- Men inte flera klasser för samma objekt
- Exempel: Niklas är av Typen människa. Alltså Niklas är en instans av klassen Människa.

Exempel:

```
namespace OOPDemo
{
    4 references
    public class Player
    {
        public string _name;
        public int _level;
        public string _class;
    }
}
```

```
namespace OOPDemo
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Player niklas = new Player();
            niklas._name = "Niklas";
            niklas._level = 12;
            niklas._class = "Joker";

            Player evaBritt = new Player();
            niklas._name = "Eva-Britt";
            niklas._level = 99;
            niklas._class = "Champion";
        }
    }
}
```



Klassen mugg instansieras tre gånger med olika värden på Färg, Volym och Innehåll

`new` – Nyckelord för att instansiera klasser

- Nyckelordet `new` används för att skapa objekt.
- En klass används för att beskriva hur ett objekt ska se ut.
- Objekt kan användas för att utföra uppgifter, skapa små delar av en applikation eller för att strukturera data.

Övning

- Skapa en konsolapplikation med en klass för en pokémon (kalla den Pokemon)
- Pokemon ska ha en variabel för namn (_name) och en för typ (_type)
- I Program.cs, skapa 2 instanser av Pokemon och ge båda var sitt namn och var sin typ.
- Skriv sedan ut namnet och typen för vardera genom att skriva ut värdet på variablerna i vardera objekt.

Class Members

Field – Fält

- Ett fält är en variabel som är tillgänglig i hela klassen
- Ett fält kan ha access modifiers (public/private etc)
- När man namnger ett fält är det bra att kunna särskilja dem från lokala variabler (ett sätt är att börja namnet med ’_’)
- Exempel på fältnamn: _name, _age, _characterTraits

```
string _name;
int _level;
string _class;
```

Konstruktorer

- En Klass har alltid en konstruktor
- En konstruktor är en sorts metod som returnerar en instans av typen som den finns i
- Om det inte finns någon konstruktor definierad så används defaultkonstruktorn
- Defaultkonstruktorn har inga parametrar och gör inget annat än att returnera en instans utav klassen
- Om en klass har en konstruktor definierad så skriver den över defaultkonstruktorn
- En klass kan ha flera konstruktorer så länge de har olika parametrar

```
public class Player
{
    private string _name;
    private int _level;
    private string _class;

    2 references
    public Player(string name, int level, string charClass)
    {
        _name = name;
        _level = level;
        _class = charClass;
    }
}
```

```
2 references
public Player(string name, int level, string charClass)
{
    _name = name;
    _level = level;
    _class = charClass;
}

0 references
public Player()
{
}
```

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Player niklas = new Player("Niklas", 12, "Joker");

        Player evaBritt = new Player("Eva-Britt", 99, "Champion");
    }
}
```

Properties – Säkrare variabler

- En klass kan definiera flera properties
- En Property är ett sätt att exponera värdet av ett fält
- En property består av en get- och en set-metod
- En get-metod låter oss kontrollera hur värdet skall presenteras
- En set-metod låter oss bestämma hur ett värde ska få sättas

Static or not?

- Static är ett nyckelord som bestämmer huruvida en class member är synlig på Typen eller enbart på instanser av Klassen
- En static metod körs på typen ej på en instans utav Klassen
- En static metod har alltså inte tillgång till värden på specifika värden eller icke-static metoder

Object

Basklassen till alla klasser

Object – Vad innehåller den?

- Object innehåller några metoder som är viktiga att känna till
- Equals(object) – Används för att jämföra två objekt med varandra
  - Basimplementationen av Equals jämför referenserna. Alltså inte innehållet.
  - Basimplementationen returnerar alltså bara true om man jämför en referens till samma objekt.
- ToString() – Hanterar hur ett objekt representeras som sträng
  - Basimplementationen av denna representerar ett objekt som dess namn, exempel: **ConsolApp.Pokemon**
- Andra som är värda att känna till: Finalize, GetHashCode, GetType, ReferenceEquals och MemberwiseClone

Override – Ändrat beteende av metod

- ”Method overriding” – ett sätt att implementera ett nytt beteende hos en klass som ärver av en annan.
- Equals och ToString är två metoder som ganska ofta skrivs över med override.
- Equals kan skrivas över för att beskriva hur objekt av en viss typ ska jämföras.
- Man kan alltså istället för att jämföra referenser välja att jämföra värden på properties.

Struct

Struct – Sammansatt värdetyp

- Till skillnad från klasser som är referenstyper så är struktar värdetyper
- En struct kan innehålla fält, properties, metoder och kan ha kosntruktorer
- Struct används vanligtvis endast i specialfall men är värda att känna till

Record

Record – En speciell referenstyp

- Records är referenstyper men med egen implementation av bland annat Equals och ToString
- En record kan innehålla fält, properties, metoder och kan ha konstruktorer
- Record är det senaste tillskottet till Typ-familjerna
- Records används ofta för att definiera databehållare, tex Kund, Produkt osv.
- Records är per default klasser (record class), men man kan också definiera dem som struktar (record struct)

Livekod och Exempel

Collections

Collection – En samling objekt

- En Collection är en klass som kan hålla i en grupp av objekt
- Till skillnad från en Array så kan antalet element i en Collection variera och behöver inte ha en definierad längd när den skapas
- Collections är till för att på ett enklare sätt hantera sökning, hantering och lagring av objekt.
- Exempel på Collections i C#:
  - List, Queue, Stack, Dictionary, SortedList

List

- List<T> Är en starkt typad lista där man använder index för att hämta element.
- Några av de metoder som List innehåller
  - Add – Lägg till element i listan
  - Clear – Rensar hela listan
  - Contains – Letar efter ett specifikt element i listan
  - IndexOf – Returnerar index av ett element
  - Sort – Sorterar listan
  - Reverse – Vänder ordningen på element i listan

Queue

- Queue<T> Är en Collection som agerar som en kö
- Några av de metoder som Queue innehåller
  - Enqueue – Lägg till ett element sist i kön
  - Dequeue – Tar bort och returnerar det elementet som ligger först i kön
  - Peek – Returnerar det elementet som ligger först utan att ta bort det ur kön
  - Contains – Letar efter ett specifikt element i kön
  - Clear – Rensar kön
  - ToArray – Kopierar alla elemnt i kön till en ny array

Stack

- Stack<T> är en omvänd kö där det senast tillagda hamnar först
- Några av de metoder som Stack innehåller
  - Pop – Tar bort och returnerar det översta elementet i Stacken
  - Push – Lägger till ett element överst i stacken
  - Peek – Retrunerar det översta elementet i stacken utan att ta bort det
  - Constains – Letar efter ett specifikt element i stacken
  - Clear – Rensar stacken
  - ToArray – Kopierar alla element i stacken till en ny array

Dictionary

- Dictionary<Tkey,TValue> är en typ av collection som kan spara 2 element per plats
- TKey – Typen på key-elementen
- TValue – Typen på value-elementen
- Properties tillgängliga på Dictionary
  - Keys – En collection av alla Keys
  - Values – En collection av alla Values
- Några av de metoder som finns i Dictionary
  - Add – Lägg till en kombination av en Key och en Value

SortedList

- Fungerar samma som en Dicitonary men är sorterad på Key enligt en Comparer
- Om ingen Comparer anges så används default (Som i Array.Sort)

Livekod och Exempel

Parövning!

- Utgå ifrån repot PokeGame.
- Utöka Trainer med en property för PokemonCollection av typen List<Pokemon>
- Lägg till en publik metod i Trainer som heter catch som tar emot en Pokemon som parameter och lägger till den i PokemonCollection
- Lägg till en publik metod i Trainer som heter release som tar emot en Pokemon som parameter och tar bort en Pokemon ur sin PokemonCollection med samma namn.
- Instansiera en Trainer i Program.cs
- Instansiera 5 olika Pokemon och lägg till dem i trainer.PokemonCollection
- Skriv ut namnet på alla Pokemon i trainer.PokemonCollection
- Släpp ut alla Pokemon ur PokemonCollection en och en och skriv ut vilken som släpps ut varje gång.