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

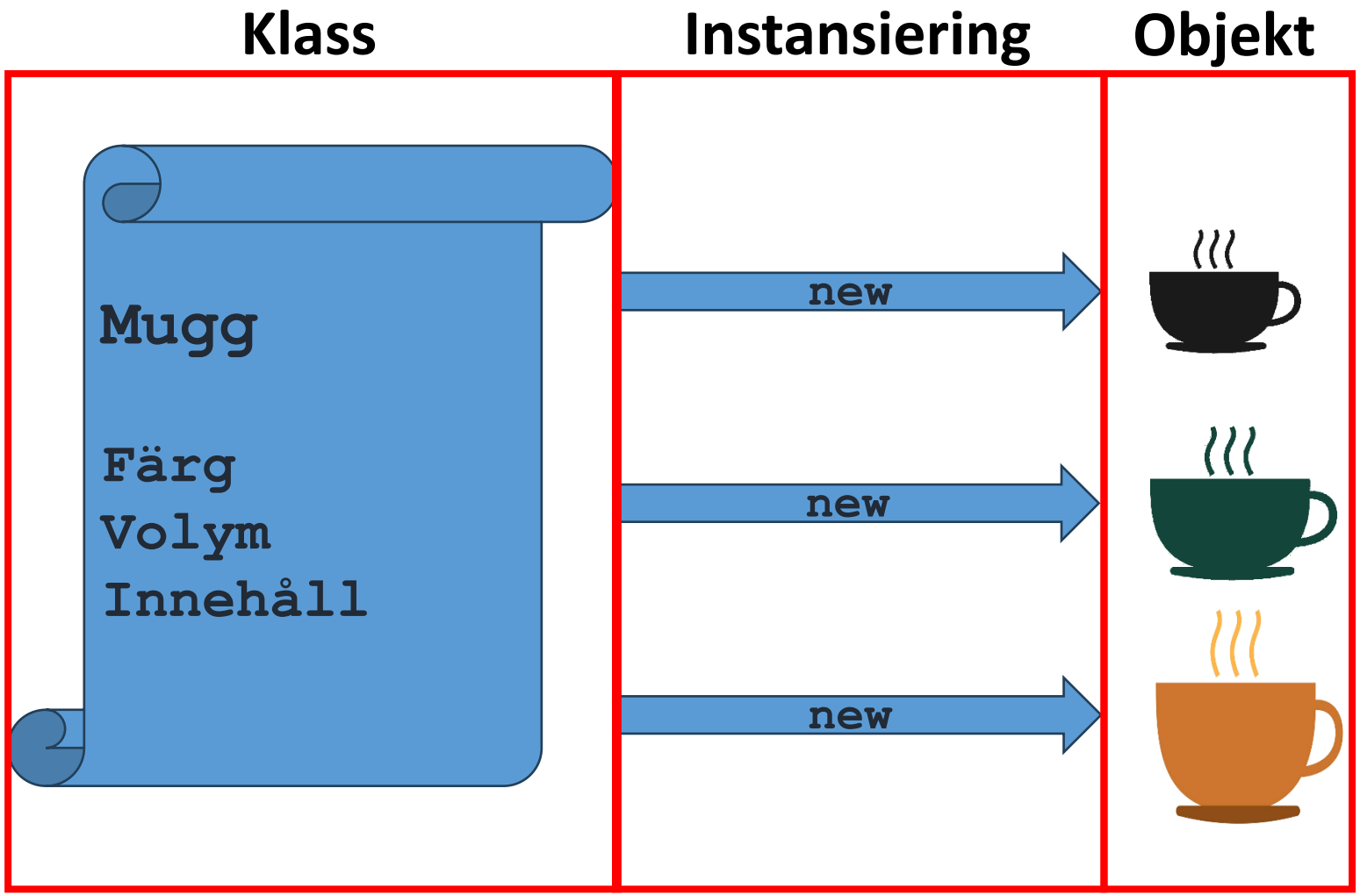
- En Klass kan liknas vid en ritning för ett objekt
- Ett objekt är en instans utav en Klass
- Ett objekt har en Typ – vilken klass den är en instans av
- Man kan skapa flera instanser av samma klass
- Men inte flera klasser för samma objekt
- Exempel: Niklas är av Typen människa. Alltså Niklas är en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

Namnkonventioner

- Projekt, Namespace, Klassser, Metoder, Properties
  - PascalCase – Varje nytt ord startar med versal följt av gemener, inga understreck
- Fält
  - _understreckFörst – Fält startar med understreck följt av camelCase
- Lokala variabler och parametrar
  - camelCase – Startar med gemen följt av versal först i varje nytt ord
- Konstanter
  - ALL_CAPS – Allt i versaler med understreck mellan orden