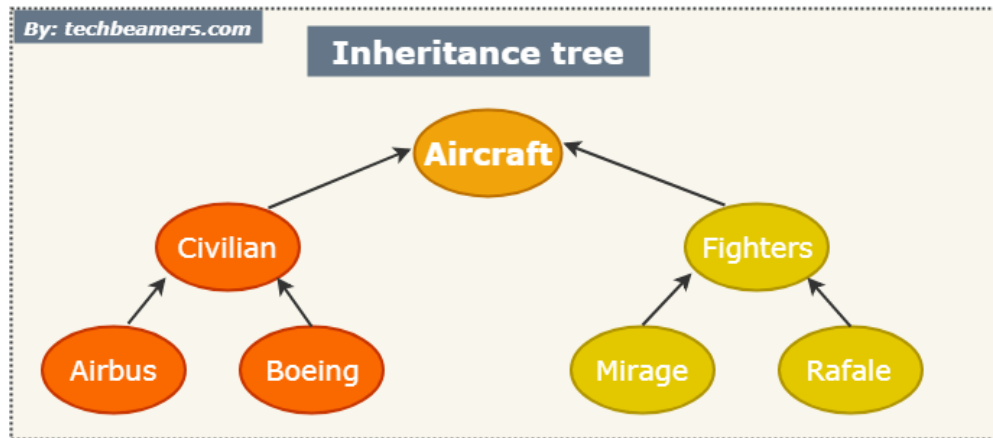
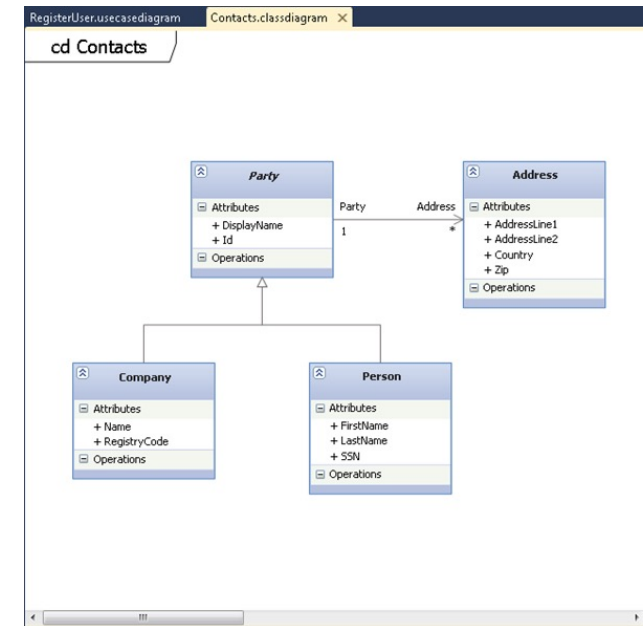


Nasljeđivanje

Princip kojim novoizgrađena klasa izvlači značajke (metode i svojstva) iz već postojeće klase. Koristeći nasljeđivanje podaci postaju dostupni hijerarhijskim redoslijedom.



UML notacija



Izvori:

<https://www.techbeamers.com/java-inheritance/>

<https://www.uml.org/>

<https://gunnarpeipman.com/visual-studio-2010-uml-modeling-projects/>

Nasljeđivanje

```
1  using System;
2  namespace Predavanje6.Nasljedivanje
3  {
4      public class Osoba
5      {
6          public string Ime { get; set; }
7          public string Prezime { get; set; }
8
9          public override string ToString()
10         {
11             return Ime + " " + Prezime;
12         }
13     }
14 }
```

```
1  using System;
2  namespace Predavanje6.Nasljedivanje
3  {
4      public class Student: Osoba
5      {
6          public string JMBAG { get; set; }
7
8          public override string ToString()
9          {
10             return base.Ime + " " + Prezime + "(" + JMBAG + ")";
11         }
12     }
13 }
```

```
1  using System;
2  namespace Predavanje6.Nasljedivanje
3  {
4      public class Profesor: Osoba
5      {
6          public int RedniBrojZnanstvenika { get; set; }
7      }
8  }
```

Izvor:

<https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/inheritance>

Nasljeđivanje

Prilikom rada s objektima ne budemo svjesni iz koje klase dolaze

Klase u C# programskom jeziku (htjele to ili ne) uvijek nasljeđuju **System.Object** koji ima sljedeće metode

Equals – uspoređivanje dva objekta.

Finalize – provodi operacije čišćenja.

GetHashCode – generira broj koji odgovara vrijednostima u objektu.

ToString – reprezentacija klase string tipom podatka.

```
public class Osoba: Object
```

Nije potrebno eksplicitno navoditi

Izvor:


<https://docs.microsoft.com/en-us/dotnet/api/system.object?view=net-5.0>

```
var student = new Student
{
    Ime = "Marija",
    Prezime = "Zimska",
    JMBAG = "0012232343321"
};
```

```
Console.WriteLine(student);
```

```
var profesor = new Profesor
{
    Ime = "Martin",
    Prezime = "Keh",
    RedniBrojZnanstvenika = 31223
};
```

```
Console.WriteLine(profesor);|
```

 Terminal – Predavanje6

```
Marija Zimska(0012232343321)
Martin Keh
```

Nasljeđivanje - konstruktori

Svaka klasa se "brine" za svoja svojstva, ostala „šalje” nadklasi

```
public class Osoba
{
    public Osoba()
    {
    }

    public Osoba(string Ime, string Prezime)
    {
        this.Ime = Ime;
        this.Prezime = Prezime;
    }

    public string Ime { get; set; }
    public string Prezime { get; set; }

    public override string ToString()
    {
        return Ime + " " + Prezime;
    }
}
```

```
public class Student: Osoba
{
    public Student() : base()
    {
    }

    public Student(string Ime, string Prezime,
        string JMBAG) : base(Ime, Prezime)
    {
        this.JMBAG = JMBAG;
    }

    public string JMBAG { get; set; }

    public override string ToString()
    {
        return base.Ime + " " + Prezime + "(" + JMBAG + ")";
    }
}
```

Nasljeđivanje - konstruktori

```
public class Profesor: Osoba
{
    public Profesor():base()
    {
    }

    public Profesor(string Ime, string Prezime,
        int RedniBrojZnanstvenika) : base(Ime,Prezime)
    {
        this.RedniBrojZnanstvenika = RedniBrojZnanstvenika;
    }

    public int RedniBrojZnanstvenika { get; set; }
}
```

```
var student = new Student("Marija","Zimska", "0012232343321");
Console.WriteLine(student);

var profesor = new Profesor("Martin", "Keh", 31223);
Console.WriteLine(profesor);
```

Terminal – Predavanje6

```
Marija Zimska(0012232343321)
Martin Keh
```

Apstraktna klasa

Apstraktna klasa je ona klasa koja nema implementaciju – nije moguće napraviti instancu te klase
Koristi se kao kontejner za zajednička svojstva i metode koja će posjedovati sve klase koje ju naslijeđe

```
public class Osoba
{
    public Osoba()
    {
    }

    public Osoba(string Ime, string Prezime)
    {
        this.Ime = Ime;
        this.Prezime = Prezime;
    }

    public string Ime { get; set; }
    public string Prezime { get; set; }

    public override string ToString()
    {
        return Ime + " " + Prezime;
    }
}
```

```
var osoba = new Osoba();
osoba.Ime = "Pero";
```

```
public abstract class Osoba
```

```
var osoba = new Osoba();
osoba.Ime = "Pero";
```



M `Osoba.Osoba()` (+ 1 overload)

CS0144: Cannot create an instance of the abstract type or interface 'Osoba'

[Show potential fixes](#)

Izvor:

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/abstract>


Apstraktna metoda

U klasi se definira potpis metode ali ne i implementacija

```
public abstract class Osoba
{
    public abstract void Pozdravi();
}
```

Klasa koja je naslijedila nadklasu s apstraktnom metodom mora ju implementirati ili sebe proglasiti apstraktnom klasom

```
public class Profesor: Osoba
{
    public Pozdravi()
    {
    }
}
```

 class Predavanje6.Nasljedicivanje.Profesor
[CS0534](#): 'Profesor' does not implement inherited abstract member 'Osoba.Pozdravi()'
[Show potential fixes](#)

```
public class Profesor: Osoba
{
    public override void Pozdravi()
    {
    }
}
```

```
public abstract class Profesor: Osoba
{
    ...
}
```

Polimorfizam

Princip kada podklasa definira svoje jedinstveno ponašanje a u isto vrijeme dalje dijeli iste funkcionalnosti ili ponašanje svoje nadklase.

eng. Polymorphism potječe od 2 grčke riječi: "poly" i "morphs". Riječ "poly" znači mnogo (više), a "morphs" znači oblik. Dakle, polimorfizam znači "više oblika" pa je stoga izraz višeobličje prikladan prijevod :)

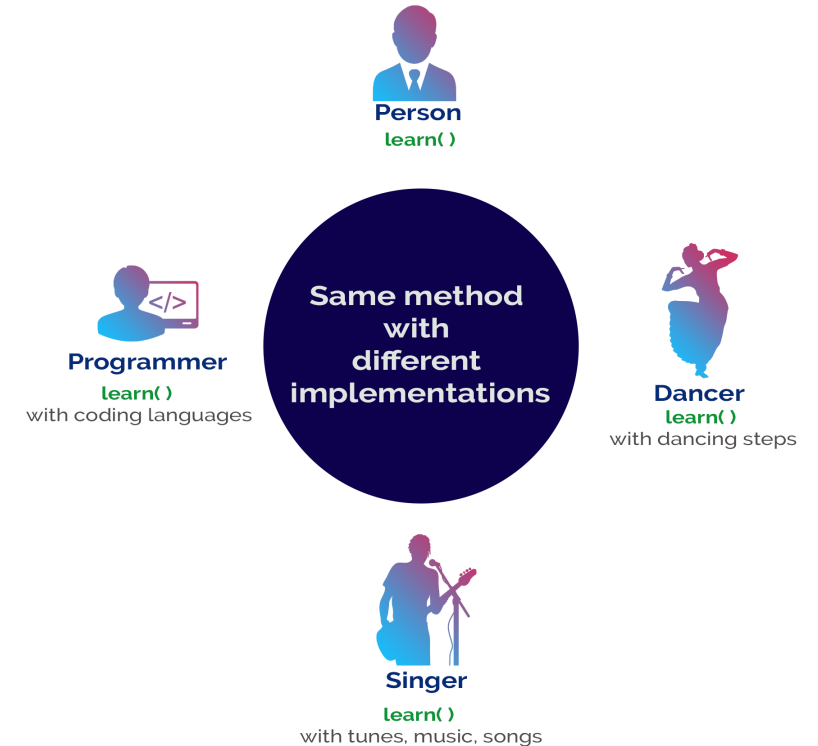
Realizira se na način da se u nadklasi definira apstraktna metoda (potpis ponašanja) a u podklasama se konkretno ponašanje i implementira.

Zašto polimorfizam?

- Dobijemo pregledniji kôd
- Kôd je na "pravom mjestu"

Izvor:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/polymorphism>



Polimorfizam

```
public abstract class Osoba
{
    public abstract string Pozdravi();

    public string Ime { get; set; }
    public string Prezime { get; set; }
    public string Spol { get; set; }
}

public class Profesor: Osoba
{
    public int RedniBrojZnanstvenika { get; set; }

    public override string Pozdravi()
    {
        return (Spol.Equals("Žensko") ? "Poštovana profesorice " : "Poštovani profesore ") +
            Prezime + " " + Ime + " (" + RedniBrojZnanstvenika + ")";
    }
}

public class Student: Osoba
{
    public string JMBAG { get; set; }

    public override string Pozdravi()
    {
        return (Spol.Equals("Žensko") ? "Poštovana kolegice " : "Poštovani kolega ") +
            Prezime + " " + Ime + " (" + JMBAG + ")";
    }
}
```

```
Osoba[] osobe = new Osoba[2];

var profesor = new Profesor
{
    Ime = "Marijan",
    Prezime = "Ketok",
    Spol="Muško",
    RedniBrojZnanstvenika = 31223
};

osobe[0] = profesor;

var student = new Student
{
    Ime = "Ivana",
    Prezime = "Jertić",
    Spol="Žensko",
    JMBAG="023232232233"
};

osobe[1] = student;

foreach(Osoba osoba in osobe){
    Console.WriteLine(osoba.Pozdravi());
}
```

Terminal – Predavanje6

```
Poštovani profesore Ketok Marijan (31223)
Poštovana kolegice Jertić Ivana (023232232233)
```