Polymorfismus, virtuální metody, abstraktní metody, abstraktní třídy a rozhraní

V objektově orientovaném programování (OOP) jsou polymorfismus, virtuální metody, abstraktní metody, abstraktní třídy a rozhraní klíčovými koncepty, které umožňují vytváření flexibilních, udržitelných a znovupoužitelných kódů.

# 1. Polymorfismus

Polymorfismus znamená "mnoho podob" a v OOP to znamená schopnost objektů různých tříd reagovat na stejné volání metody, ale s různým chováním. To je dosaženo díky dědičnosti a přetížení metod, což umožňuje jednomu rozhraní obsluhovat různé datové typy.

## Typy polymorfismu:

1. Polymorfismus na úrovni metody (přetížení metod): Umožňuje mít více metod se stejným názvem, ale různými parametry.  
2. Polymorfismus na úrovni dědičnosti (přepisování metod): Umožňuje, aby metoda definovaná v základní třídě byla přepsána v odvozených třídách.

## Příklad polymorfismu na úrovni dědičnosti:

public class Zvíře  
{  
 public virtual void Zvuk()  
 {  
 Console.WriteLine("Neznámý zvuk");  
 }  
}  
  
public class Pes : Zvíře  
{  
 public override void Zvuk()  
 {  
 Console.WriteLine("Haf!");  
 }  
}  
  
public class Kočka : Zvíře  
{  
 public override void Zvuk()  
 {  
 Console.WriteLine("Mňau!");  
 }  
}  
  
Zvíře zvíře1 = new Pes();  
zvíře1.Zvuk(); // Vytiskne: Haf!  
  
Zvíře zvíře2 = new Kočka();  
zvíře2.Zvuk(); // Vytiskne: Mňau!

# 2. Virtuální metody

Virtuální metody jsou metody definované v rodičovské třídě, které mohou být přepsány v odvozených třídách. Slovo `virtual` označuje, že metoda může být přepsána.

## Příklad virtuální metody:

public class Zvíře  
{  
 public virtual void Zvuk()  
 {  
 Console.WriteLine("Zvíře dělá nějaký zvuk.");  
 }  
}

# 3. Abstraktní metody

Abstraktní metody jsou metody, které jsou deklarovány v abstraktní třídě, ale nemají žádnou implementaci. Třídy, které dědí od abstraktní třídy, jsou povinny implementovat abstraktní metody.

## Příklad abstraktní metody:

public abstract class Zvíře  
{  
 public abstract void Zvuk();  
}  
  
public class Pes : Zvíře  
{  
 public override void Zvuk()  
 {  
 Console.WriteLine("Haf!");  
 }  
}

# 4. Abstraktní třídy

Abstraktní třída je třída, která nemůže být instanciována přímo. Slouží jako základ pro další třídy, které ji dědí. Abstraktní třída může obsahovat jak abstraktní metody, tak metody s implementací.

## Příklad abstraktní třídy:

public abstract class Zvíře  
{  
 public string Jméno { get; set; }  
   
 public abstract void Zvuk();  
   
 public void PředstavSe()  
 {  
 Console.WriteLine($"Jsem zvíře jménem {Jméno}.");  
 }  
}

# 5. Rozhraní (Interface)

Rozhraní (interface) je jako šablona, která definuje metody, které musí třída implementovat, ale neobsahuje žádnou implementaci těchto metod. Třídy mohou implementovat více než jedno rozhraní.

## Příklad rozhraní:

public interface IZvuk  
{  
 void Zvuk();  
}  
  
public class Pes : IZvuk  
{  
 public void Zvuk()  
 {  
 Console.WriteLine("Haf!");  
 }  
}  
  
public class Kočka : IZvuk  
{  
 public void Zvuk()  
 {  
 Console.WriteLine("Mňau!");  
 }  
}

# Shrnutí a rozdíly mezi abstraktními třídami a rozhraními

| Funkce | Abstraktní třída | Rozhraní |  
|--------------------------------|----------------------------------------|-------------------------------------------|  
| Může mít implementaci metod | Ano | Ne |  
| Může mít konstruktory | Ano | Ne |  
| Může dědit více tříd | Ne (pouze jedna třída) | Ano (může implementovat více rozhraní) |  
| Povinná implementace metod | Ano (pro abstraktní metody) | Ano (pro všechny metody v rozhraní) |

# Závěr

- Polymorfismus umožňuje flexibilní práci s objekty různých typů pomocí stejného rozhraní.  
- Virtuální metody umožňují přepsání metod v děděných třídách.  
- Abstraktní metody a abstraktní třídy umožňují definovat "základní kostru" pro dědičné třídy.  
- Rozhraní poskytují způsob, jak zajistit, že různé třídy budou implementovat stejný soubor metod, ale bez potřeby dědit implementaci.