Objekty a třídy v objektově orientovaném programování (OOP)

V objektově orientovaném programování (OOP) jsou základními stavebními kameny třídy a objekty. Tyto koncepty pomáhají vytvářet kód, který je modulární, opětovně použitelný, snadno rozšiřitelný a udržovatelný.

# 1. Třídy

Třída je šablona, která definuje vlastnosti a chování objektů. Můžeme ji vnímat jako návrh pro vytváření objektů, který popisuje, jaké atributy (vlastnosti) a metody (funkce) budou mít objekty této třídy.

## Struktura třídy:

- Atributy: Jsou to vlastnosti, které mají objekty třídy. Můžou to být proměnné (například jméno, věk, výška).  
- Metody: Jsou to funkce, které definují chování objektů. Můžou být například pohybuj\_se(), zobraz\_info() atd.

## Příklad třídy v C#:

public class Osoba  
{  
 public string Jméno { get; set; }  
 public int Věk { get; set; }  
  
 public void PředstavSe()  
 {  
 Console.WriteLine($"Ahoj, jmenuji se {Jméno} a je mi {Věk} let.");  
 }  
}

# 2. Objekty

Objekt je konkrétní instancí třídy. Když vytvoříme objekt, znamená to, že alokujeme paměť pro všechny atributy definované v třídě a můžeme používat metody této třídy.

## Vytváření objektů (instancí třídy):

Pro vytvoření objektu třídy použijeme klíčové slovo `new`.

## Příklad vytvoření objektu:

Osoba osoba1 = new Osoba();  
osoba1.Jméno = "Jan";  
osoba1.Věk = 30;  
osoba1.PředstavSe();

# 3. Konstruktory a destruktory

Konstruktory a destruktory jsou speciální metody, které se používají pro správu životního cyklu objektů.

## Konstruktory

Konstruktor je speciální metoda, která se automaticky volá při vytváření objektu. Jeho hlavní funkcí je inicializace objektu, tedy přiřazení počátečních hodnot atributů objektu.  
Konstruktor může být bez parametrů (implicitní) nebo může mít parametry, díky kterým můžeme při vytváření objektu zadat specifické hodnoty.

## Příklad konstruktoru:

public class Osoba  
{  
 public string Jméno { get; set; }  
 public int Věk { get; set; }  
  
 // Konstruktor s parametry  
 public Osoba(string jméno, int věk)  
 {  
 Jméno = jméno;  
 Věk = věk;  
 }  
  
 public void PředstavSe()  
 {  
 Console.WriteLine($"Ahoj, jmenuji se {Jméno} a je mi {Věk} let.");  
 }  
}

## Destruktory

Destruktor je metoda, která se automaticky volá při zániku objektu, obvykle když objekt přestane být použit a je odstraněn z paměti (garbage collector).  
Destruktory jsou v C# méně běžně používány, protože garbage collector se stará o správu paměti a uvolňování objektů.

## Příklad destruktora:

public class Osoba  
{  
 // Konstruktor  
 public Osoba(string jméno, int věk)  
 {  
 Jméno = jméno;  
 Věk = věk;  
 }  
  
 // Destruktor  
 ~Osoba()  
 {  
 Console.WriteLine("Objekt Osoba byl zničen.");  
 }  
}

# 4. Správa paměti a garbage collector

V C# je správa paměti automatizována prostřednictvím garbage collectoru (GC), což znamená, že programátor nemusí explicitně uvolňovat paměť pro objekty, jak je tomu v některých jiných jazycích (např. v C nebo C++). Garbage collector se stará o to, aby byly odstraněny objekty, na které již neexistují žádné reference.

## Jak funguje garbage collector:

- GC identifikuje objekty, na které již neexistují žádné reference, a označí je jako nepoužívané.  
- Tyto objekty jsou pak uvolněny, aby uvolnily paměť.  
- Generace: GC pracuje na základě generací (0, 1, 2), kde nové objekty jsou umístěny v generaci 0, starší objekty jsou přesunuty do generace 1 a tak dále. Generace 0 jsou čistě sbírány častěji.

## Výhody garbage collectoru:

- Automatická správa paměti: Programátor nemusí ručně uvolňovat objekty.  
- Zamezení paměťových úniků: GC automaticky detekuje objekty, které již nejsou použitelné, a uvolňuje jejich paměť.

## Nevýhody:

- Výkon: GC může zpomalit aplikaci, protože vyžaduje čas na detekci a uvolnění nepoužívaných objektů.  
- Nezaručené okamžité uvolnění: GC nezná přesný čas, kdy objekt bude odstraněn z paměti, což může vést k neefektivnímu využití paměti v některých scénářích.

# 5. Shrnutí

- Třídy a objekty jsou základními stavebními kameny objektově orientovaného programování.  
- Konstruktory umožňují inicializaci objektů při jejich vytváření, a destruktory jsou volány při zániku objektu (i když v C# jsou méně běžné).  
- Garbage collector se stará o automatické uvolňování paměti, což zjednodušuje správu objektů a zabraňuje paměťovým únikům, ale může zpomalit výkon aplikace.