**Konstanty, Statické metody a vlastnosti, Anonymní metody (Lambda), Ukazatel na metodu (delegát)**

**Konstanty**

Konstanty jsou neměnné hodnoty, které jsou známy v době kompilace a nemění se po dobu života programu.

Využití z konstantních hodnot je když víme, že se hodnota nebude měnit běham životu programu a umožní kompilátoru programu optimalizovat kód a zrychlit tím běh (o trošku, kompilátor vám uvěří, že se hodnota nebudě měnit)

Klíčové slovo “const“ se hlavně využívá, třeba u práce více lidí na jednom projektu, tak to slovo řekne ostatním, že se ta hodnota nikdy nemění

Musí mít vždy hodnotu, nemůže nastat situace, kdy bude null

Jsou tedy readonly

Konstanty jsou deklarovány s modifikátorem – const

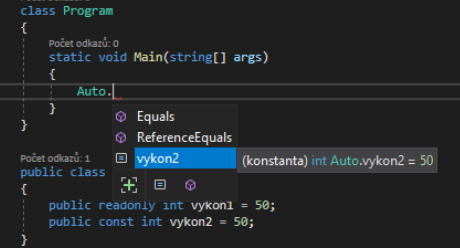
Pouze předdefinované typy jazyka C# (s výjimkou System.Object ) mohou být deklarovány jako const

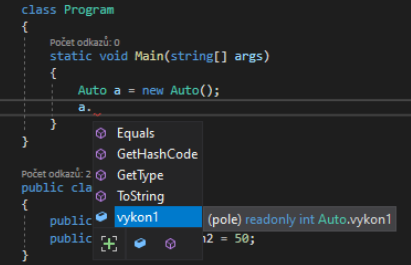
Uživatelsky definované typy, včetně tříd, struktur a polí, nemůžou být const .

**Rozdíl mezi const a readonly**

Const jsou statické (společné pro všechny instance)

Readonly může obsahovat rozdílné hodnoty pro instance (nastaví se její hodnota v konstruktoru)





**Statika**

Nemusí být existovat instance nějaké třídy, aby se statický prvek dokázal zavolat.

Volá se na třídě.

Třeba Třída Console má statické metody WriteLine a Write atd…

OOP umožňuje definovat atributy a metody na samotné třídě. Tyto prvky jsou statické a jsou nezávislé na instanci.

Jde dát skoro před všechno: pole, metody, vlastnosti, operátory, eventy a konstruktory

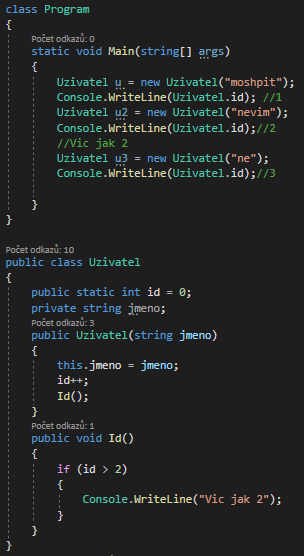
?? v podstatě narušují objektový model. OOP je obsahuje jen pro speciální případy a obecně platí, že vše jde napsat bez statiky. Vždy musíme pečlivě zvážit, zda statiku opravdu nutně potřebujeme.

**Statické vlasnosti**

statické prvky patří třídě

Data v nich uložená tedy můžeme číst bez ohledu na to, zda nějaká instance existuje.

Jsou společné pro všechny instance (i když s nima vůbec nesouvisí, ale nejde, aby instance měly rozdílné hodnoty u statických prvků)



**Statické metody**

Statické metody se volají na třídě.

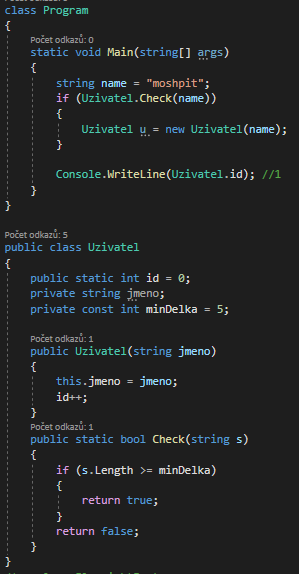
Jedná se zejména o **pomocné metody**, které potřebujeme často používat a nevyplatí se nám tvořit instanci.

Metody na Console třeba, nejde vytvořit její instance, protože konzole je jen jedna a bylo by zbytečné tvořit si z ní instanci

Další třeba metoda Round() ve třídě Math

**Příklad**

Pomocné statické metody na zkontrolování délky jména, aby mohla být instance v pořádku vytvořena



Ve statické metodě nemůžeme použít proměnné, které nejsou statické (const je)

**Delegát**

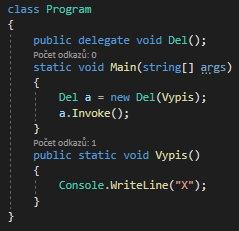
Je to ukazatel na funkci (metodu) = uložení metody do proměnné

Při jeho vytváření je nutné určit návratový typ a vstupní parametry metody.

obsahuje referenci metody

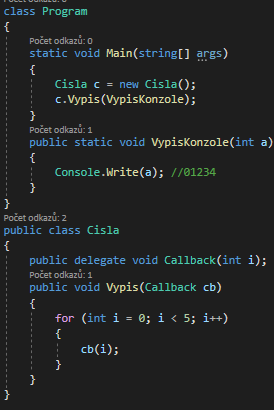
objektově orientované a typově bezpečné ukazatele na funkce. Je možné deklarovat funkci, která bude mít jako parametr delegáta.

Existuje i multi-část delegate, když se mu přidávají více metod +=



Tohle se dá napsat mnohem kratší bez delegáta

Výhoda: Callback – datová komunikace

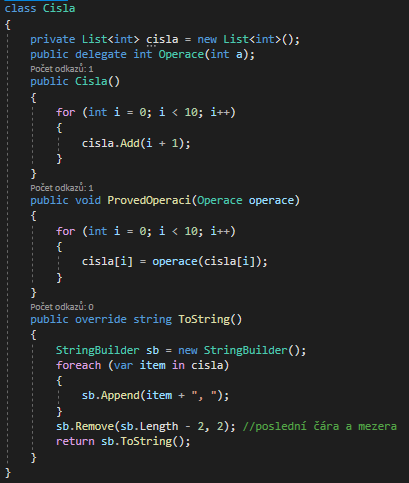


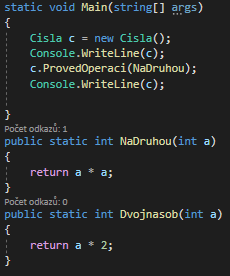
Když potřebujme někomu passnout metodu

**Použití callback**

Když potřebujeme zopakovat nějakou stejnou akci pro metodu

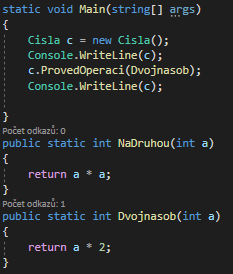
**Příklad 2**







A změnění





Delegát umožnil lehce změnit vzorec

**Event**

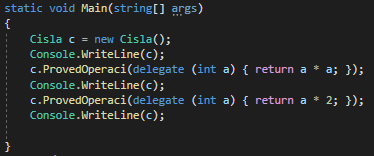
Podobrý delegátu, taky slouží pro datovou komunikaci

K jeho vytvoření je zapotřebí delegáta

**Anonymní metody**

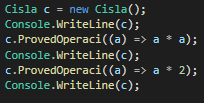
Vytvoření dočasné proměnné

je "vložený" příkaz nebo výraz, který lze použít všude, kde je očekáván typ delegáta



Zápis anonymních delegátů můžeme ještě zkrátit a to pomocí lambda výrazů.

Jedná se o zkrácený zápis definice delegáta výše.



Příklad zkrácení psaní metod pomocí lambda

