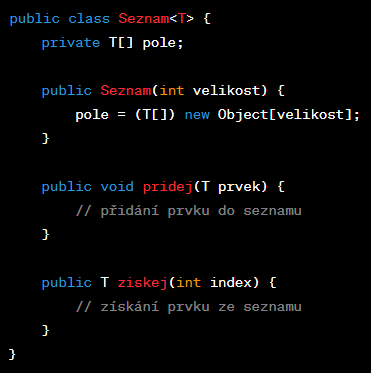
1. Generické programování

# Význam generického programování

* Programovací paradigma, které umožňuje psát kód nezávislý na konkrétních datových typech, ale spíše zobecnitelný na širší množinu typů
* To umožňuje vytvářet víceúčelové a znovupoužitelné kódy, což vede k menšímu množství kódu

# Generický typ T (písmenko libovolné)

* Generický typ T je konvence v mnoha programovacích jazycích pro označení generického typu, který může být nahrazen konkrétním typem při použití daného kódu
* Umožňuje psát obecný kód, který může pracovat s různými datovými typy
* Obecný seznam
* Seznam, který pracuje s celými čísly

# Výhody generického programování

* Zvýšená bezpečnost a stabilita
  + Generické typy umožňují kontrolovat správnost datových typů v době kompilace a zabraňují tak chybám typu v době běhu programu
  + To zlepšuje stabilitu a bezpečnost kódu
* Znovupoužitelnost kódu
  + Generický kód může být použit pro více různých datových typů, což znamená, že programátor nemusí psát stejný kód pro každý konkrétní datový typ
  + To vede k menšímu množství kódu a zvyšuje produktivitu programátora
* Flexibilita
  + Generické typy umožňují pracovat s různými datovými typy bez nutnosti vytvářet specifický kód pro každý typ
  + To znamená, že programátor může snadno měnit datové typy, s nimiž kód pracuje, aniž by musel měnit samotný kód
* Zlepšená čitelnost kódu
  + Generické kódy jsou obecně kratší a přehlednější než kódy, které pracují s konkrétními datovými typy
  + To zlepšuje čitelnost kódu a umožňuje snadnější údržbu
* Vylepšená výkonnost
  + V některých případech může být generický kód rychlejší než kód, který pracuje s konkrétními datovými typy
  + To je způsobeno tím, že generický kód může být optimalizován při kompilaci pro více různých datových typů

# Nevýhody a omezení generického programování

* Složitost syntaxe
  + Syntaxe generického kódu může být složitá a závisí na konkrétním programovacím jazyku
  + To může ztížit pochopení a psaní kódu pro méně zkušené programátory
* Omezení na statické typování
  + Generické programování vyžaduje statické typování, což znamená, že datový typ musí být určen před spuštěním programu
  + To může být omezující pro některé programy, které potřebují dynamické typování
* Výkonové omezení
  + V některých případech může generický kód být pomalejší než kód, který pracuje s konkrétními datovými typy
  + To je způsobeno tím, že generické typy vyžadují více paměti a zpomalují běh programu
* Omezení na omezené množiny datových typů
  + Některé programovací jazyky mohou mít omezení na to, které datové typy mohou být použity jako generické typy
  + To může být omezující pro některé programy, které potřebují pracovat s mnoha různými datovými typy
* Potřeba dalšího návrhu
  + Kód psaný s použitím generických typů musí být navržen tak, aby byl znovupoužitelný a flexibilní
  + To může být náročné pro programátory, kteří nejsou zvyklí na návrh obecného kódu

# Implementace a jazykové konstrukce

* Generické typy implementují pomocí parametrizovaných tříd, rozhraní a metod
* Parametrizované třídy
  + Jsou třídy, které obsahují jednu nebo více proměnných typu
  + Tyto proměnné typu jsou specifikovány v hranatých závorkách za názvem třídy

Obsah obrázku text

Popis byl vytvořen automaticky

* Parametrizované rozhraní
  + Fungují podobně jako parametrizované třídy, ale jsou používány pro definici metod, které mohou pracovat s různými typy dat

Obsah obrázku text

Popis byl vytvořen automaticky

* Generické metody
  + Umožňují specifikovat parametr typu pro metodu, která není součástí generické třídy
  + Parametr typu se specifikuje před návratovým typem metody

Obsah obrázku text

Popis byl vytvořen automaticky

# Prakticky:

* Nastavit třídu, aby byla generická, popis ukázky kódu