Kolekce

Na co to je?

Datové struktury

Množina dat uložená na základě logických pravidel.

Způsob uložení dat rozhoduje o efektivitě algoritmu.

Operace s daty - řidávání, odebírání, vkládání, třídění, apod.

Přehled

Array (pole)	Práce s pevně daným počtem elementů stejného typu
List (seznam)	Dynamicky alokované pole – počet ukládaných elementů není nutné znát předem
Dictionary (slovník)	Efektivní pro ukládání párů klíč – hodnota.
ArrayList	Ukládání objektů jakéhokoliv typu, ale vyžaduje casting.
BitArray	Pro práci s bitovými poli.
KeyValuePair	Obsahuje dvojici klíč – hodnota. Používá se pro přístup k elementům Dictionary
Stack (zásobník)	LIFO – Last In First Out struktura. Položky, které jsou přidané poslední jsou první k dispozici.
Queue (fronta)	FIFO – First In First Out struktura

Přehled

HashSet (množina)	Uložení bez ohledu na pořadí s množinovými operacemi
LinkedList	Vpřed i vzad propojené elementy
SortedList	Seřazený seznam elementů
SortedDictionary	Slovník elementů seřazený podle klíčů ve dvojici klíč – hodnota

Hlavní kolekce (using System.Collections.Generic;)

Array	Práce s pevně daným počtem elementů stejného typu
List	Dynamicky alokované pole – počet ukládaných elementů není nutné znát předem
Dictionary (SortedDictionary)	Efektivní pro ukládání párů klíč – hodnota.
HashSet	Uložení elementů bez ohledu na pořadí a s množinovými operacemi
Queue	FIFO – First In First Out struktura. Položky, které jsou přidané jako první jsou první k dispozici.
Stack	LIFO – Last In First Out struktura. Položky, které jsou přidané poslední jsou první k dispozici.

Array


```
int[] array = new int[] { 35, 33, 42, 10, 14, 19, 27, 44, 26, 31 };

// nebo

var altArray = new int[10] { 35, 33, 42, 10, 14, 19, 27, 44, 26, 31 };
```

Konečný počet elementů stejného typu.

List


```
List<string> listOfStrings =
      new List<string> { “A”, “B”, “C”, “D”, “E”, “D”, “E” };

// nebo bez duplicit v popisu typu
var listOfStrings =
      new List<string> { “A”, “B”, “C”, “D”, “E”, “D”, “E” };

// nebo s konstruktorem přijímajícím pole
string[] strs = { “První”, “Druhý”, “Třetí” };
var listOfStrings =
      new List<string>(strs);
```

Předem neurčený počet elementů stejného typu.
List automaticky rozšíří svou velikost paměti, pokud je to potřeba.
Umožňuje duplicity – neexistuje unikátní klíč.

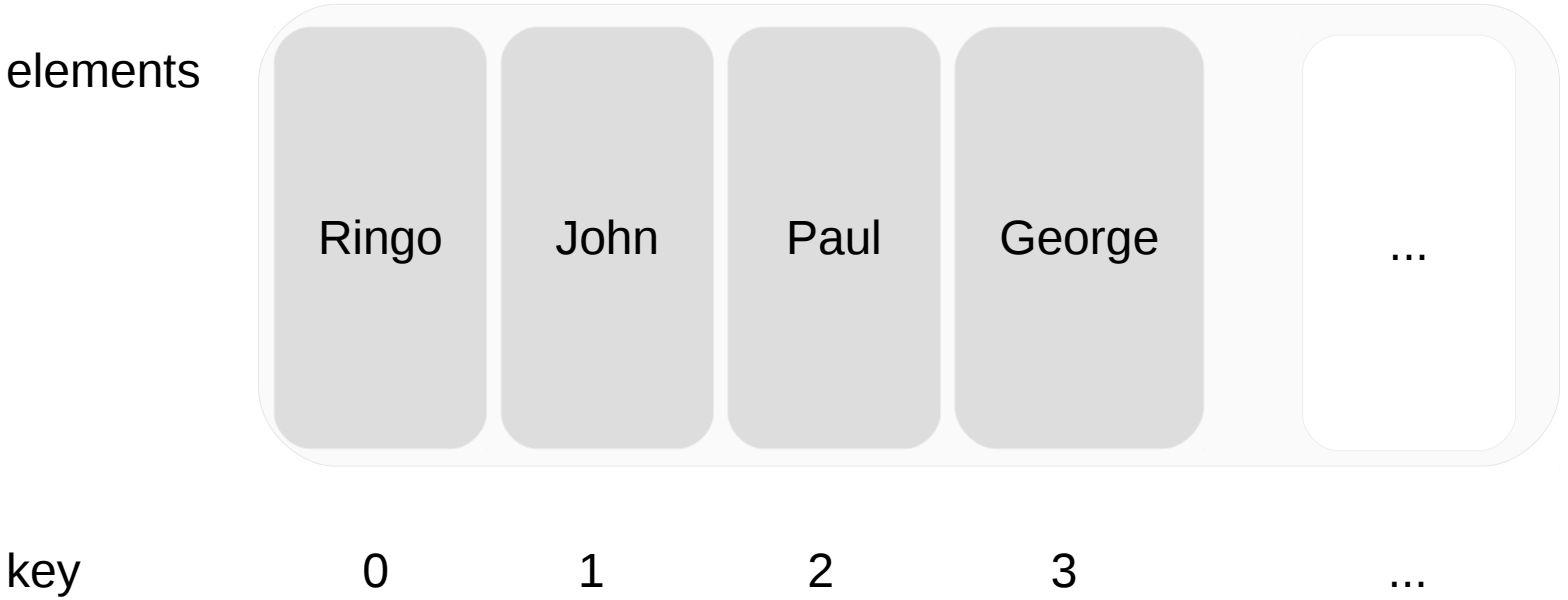
Příklad

Otevřete soubor SlovoOSlove.txt a načtěte každé slovo do seznamu proměnné typu List. Soubor SlovoOSlove.txt umístěte do adresáře bin/Debug ve vašem projektu. Ošetřete výjimky při otevírání souboru pomocí try – catch.

Vypište počet slov v seznamu a celkovou aktuální kapacitu seznamu.

Setříděte seznam a vypište jej.

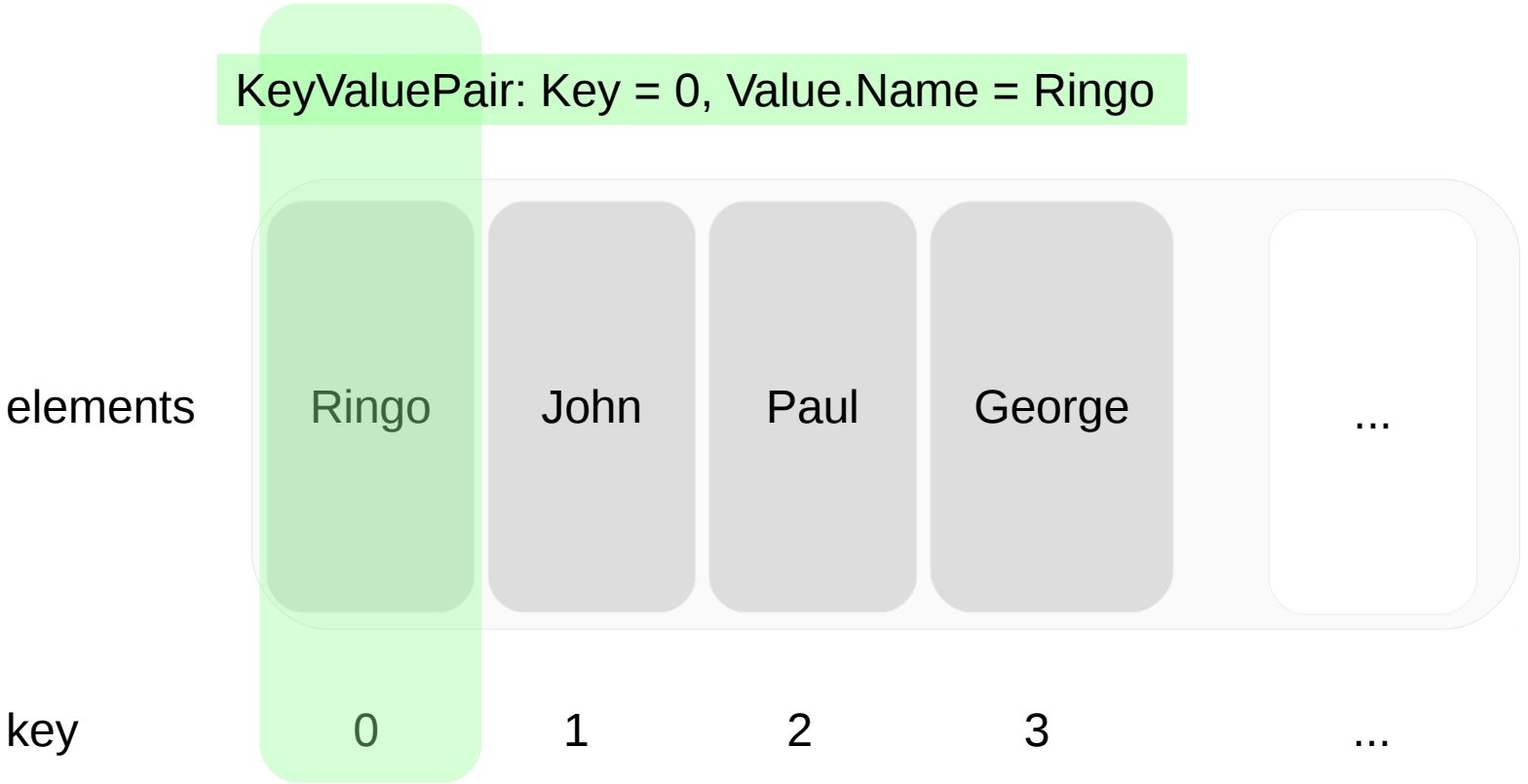
Dictionary




```
var dict = new Dictionary<int, Person>();
...
foreach (KeyValuePair<int, Person> v in dict)
```

Předem neurčený počet dvojic klíč - hodnota.
Dictionary automaticky rozšíří svou velikost paměti, pokud je to potřeba.
Klíčem může být int, string nebo cokoliv, co lze porovnávat.
Klíč je unikátní => Dictionary je bez duplicit!

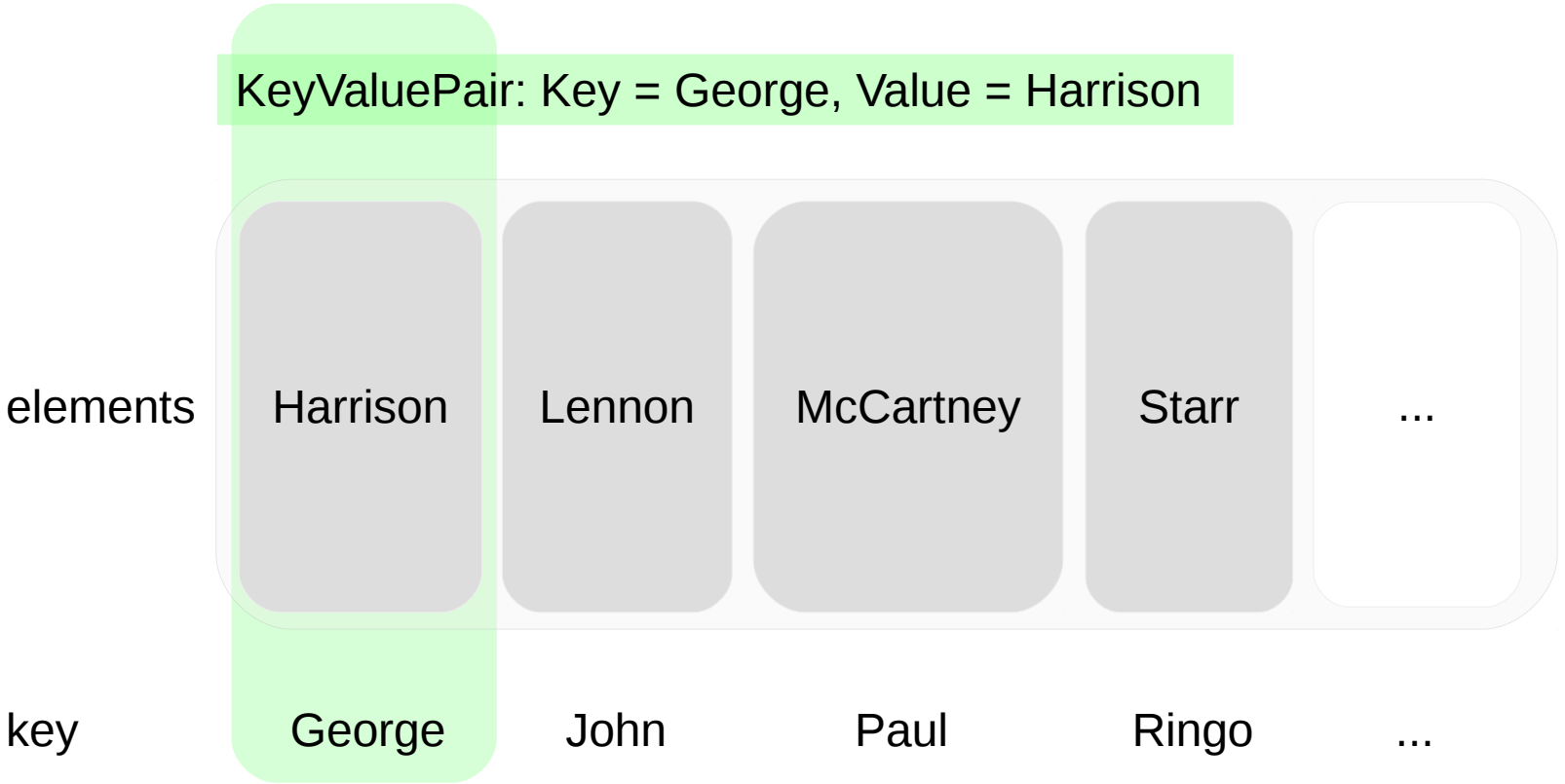
Dictionary




```
foreach (KeyValuePair<int, Person> v in dict)
    ...
```

Předem neurčený počet dvojic klíč – hodnota (key – value).
Dictionary automaticky rozšíří svou velikost paměti, pokud je to potřeba.
Klíčem může být int, string nebo cokoliv, co lze porovnávat.

SortedDictionary




```
var dictByString = new SortedDictionary<string, string>();
dictByString.Add ("Ringo", "Starr");
dictByString ["George"] = "Harrison";
```

!

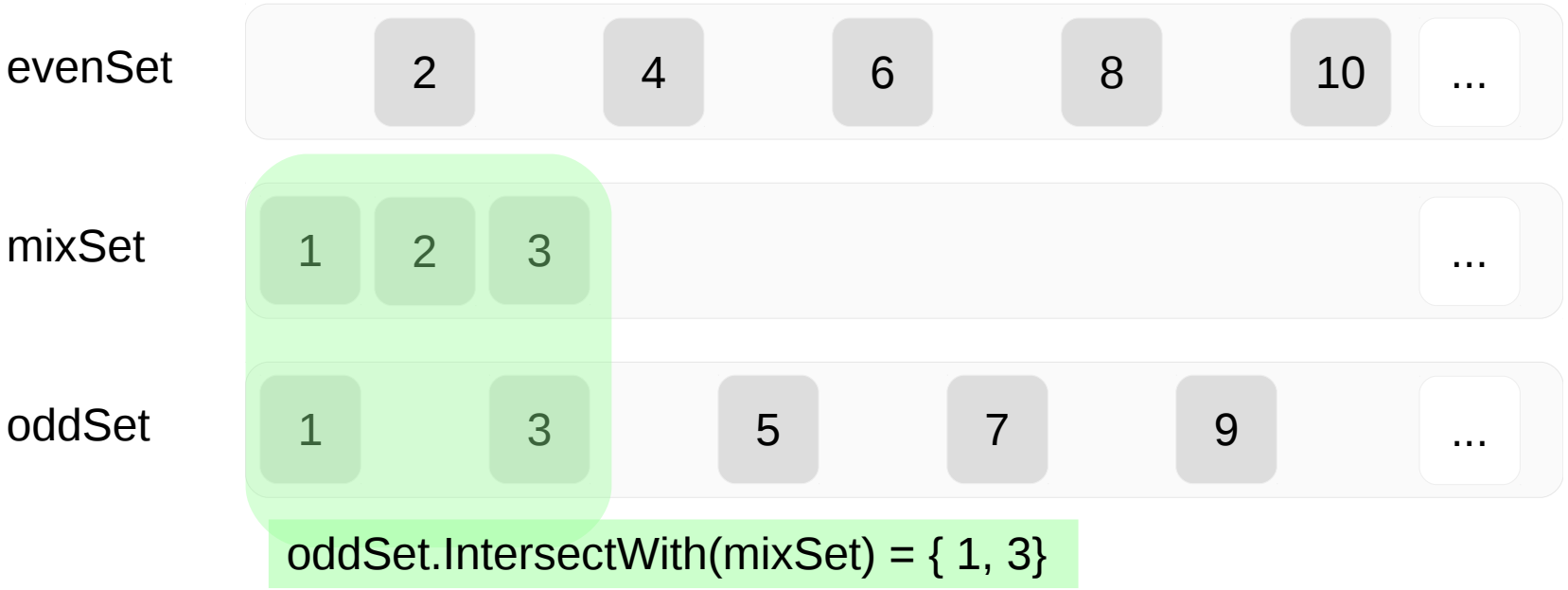
Všechny vlastnosti Dictionary +
„zadarmo“ setříděné podle klíče

Příklad

Přepište příklad, který počítal frekvenci výskytů jednotlivých slov tak, aby využil vlastností SortedDictionary.

(Příklad Četnost slov v textu v kapitole Práce s řetězci)

HashSet




```
HashSet<int> oddSet = new HashSet<int>();
oddSet.Add(1); oddSet.Add(3);
oddSet.UnionWith(evenSet);
```

Kolekce, která obsahuje elementy bez duplikace a bez předchozího určení jejich počtu.
Nepracuje s pořadím elementů a nemá index (klíč).
Umožňuje základní operace sjednocení, průniku, podmnožiny, nadmnožiny apod.

Příklad

Vytvořte třídu Man s vlastnostmi Retired (bool), Gender (enum Sex) a Name(string) a s parametrickým konstruktorem. Vytvořte redefinicí metodu ToString(), která vypíše stav objektu.

Vytvořte a inicializujte množiny setMen, setWomen, setRetired. Přidejte do množin setMen a setWomen po dvaceti členech typu Man s náhodně generovanou vlastností Retired (false nebo true). Do množiny setRetired vložte pouze ty členy typu Man, které mají vlastnost Retired == true.

Vypište sjednocení množin setMen a setWomen. Vypište průnik takto vzniklé množiny s množinou setRetired.

Queue

Peek()

elements

Dequeue()

Enqueue("X")

A

X

```
Queue<string> myQueue = new Queue<string>();
myQueue.Enqueue(“Hello”);
myQueue.Enqueue(“World”);
Console.WriteLine(myQueue.Dequeue()); // Hello
```

Kolekce, která obsahuje seřazené elementy podle času, ve kterém byly elementy vloženy do fronty.
Enqueue vkládá element na konec fronty, Dequeue odebírá element ze začátku fronty.
Peek prohlíží další element dostupný pro Dequeue, ale neodebírá ho.

Příklad

Vytvořte třídu Message, která obsahuje jedinou vlastnost Body typu string. Vytvořte její parametrický konstruktor a předefinujte metodu ToString() tak, aby vracela obsah vlastnosti Body

Vytvořte třídu Man s vlastnostmi rodné číslo, jméno a příjmení. Vytvořte její parametrický konstruktor a předefinujte její virtuální metodu ToString(), která bude vracet rodné číslo, jméno a příjmení. Vytvořte metodu SendMessage(), která vloží do fronty zpráv objektu typu Man textovou zprávu typu Message

Vytvořte frontu pěti objektů typu Man. Odebírejte postupně prvky z čela fronty a vypište jejich obsah. Pokud mají ve frontě zpráv nějaké zprávy, vypište je také.

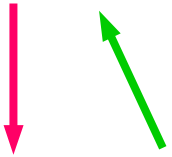
Stack

Peek()

elements

Count = 7

Pop()



Push("X")


```
Stack<string> myStack = new Stack<string>();
myStack.Push("Hello");
myStack.Push("World");
Console.WriteLine(myStack.Pop()); // World
```

Kolekce, která obsahuje seřazené elementy podle doby, ve které byly vloženy do fronty.
Push vkládá element na vrchol zásobníku, Pop odebírá element z vrcholu zásobníku.
Peek prohlíží další element dostupný pro Pop, ale neodebírá ho.

Příklad

Vytvořte třídu Stránka s vlastnostmi Číslo stránky (integer) a Text (string), vytvořte její parametrický konstruktor, který nastaví číslo stránky a text. Předefinujte její virtuální metodu ToString(), která bude vracet string uložený v Text a číslo stránky.

Vkládáním perací Push() vytvořte Dokument jako zásobník(Stack) tří Stránek.

Ověřte, zda je v zásobníku uložena první stránka pomocí metody Contains() a vypište výsledek ověření (true/false) na konzoli.

Vypište v cyklu Stránky v obráceném pořadí pomocí metody Pop(). Možnou chybu při čtení z Dokumentu ošetřete konstrukcí try – catch.

1) Vytvořte textový program Poet, který nabízí začínajícímu básníkovi následující funkce:

2) Založení nové básně – vložení a uchování vlastností Název, Pseudonym, Text básně a Celkový čas tvoření (ten je definovaný jako čas od otevření po uložení básně).

3) Vložení nového verše(řádku) na aktuální pozici v básni (neomezeně). Po každém vložení nebo jiné změně se báseň celá znovu vypíše.

4) Zpětný krok – zruší se poslední vložení od aktuální pozice (neomezeně až na začátek básně).

5) Dopředný krok – opak Zpětného kroku. Znovu se vloží to, co bylo zrušeno ve Zpětném kroku (neomezeně až do konce básně).

6) Uložení básně do textového souboru ve formátu: Název Pseudonym Celkový čas tvoření (součet všech časů, kdy byla báseň otevřená) Text básně (Název souboru bude totožný s názvem básně.)

7) Načtení rozdělané básně ze souboru, umožnění úprav podle pravidel 2 až 5. Název souboru bude možné zadat z příkazového řádku. Pokud nebude na příkazovém řádku, program si vyžádá zadání názvu souboru interaktivně. Pokud nebude soubor zadaného jména existovat, program ho vytvoří a nastaví Název básně na název souboru. Pokud zadaný název souboru nebude správným názvem (tedy bude nepovoleným), program skončí s chybou.

8) Samostatná práce je hodnocená stejně jako běžná písemná práce.

9) Hodnotí se: Úplnost: zadání má být splněno zcela, včetně ošetření chybových stavů. Správnost: program se musí chovat logicky správně a provádět všechny požadované operace. Styl: názvy tříd, metod a proměnných, komentáře, formátování kódu, volba vhodných datových a programových struktur. Originalita: opsané práce budou hodnocené jako nedostatečné.

10) Prostor pro konzultace a vyjasnění zadání bude během vyučovacích hodin.