Programovanie v jazyku C#

Vybrané kolekcie a dátové štruktúry

prednáška 6
Ing. Lukáš Hruška
ak. rok. 2021/2022 ZS

Property (C# specific)

- Umožňujú pristupovať k členským premenným ako keby boli verejné
  - Accessor / mutator je volaný implicitne
  - Možnosť obmedziť prístup rovnako ako pri metódach (napr. public get a private set), taktiež možné prekrývať
    - Od C# 9.0 možnosť povoliť nastavenie iba pri vytváraní objektu (init)
  - Jednoduché (automatické) explicitne nedefinujú členskú premennú, v ostatných prípadoch je ju potrebné definovať
- Data (model) class
  - Obdobné použitie ako struct – zvyčajne neobsahuje žiadne implementačné detaily
  - V C# sa využívajú properties
  - Slúži na uloženie/mapovanie dát (napr. pri komunikácií s databázou, čítaní štrukturovaných súborov)

Kolekcie

- Implementujú IEnumerable a IEnumerator
  - Spoločné rozhrania na prístup k dátam uloženým v rôznych typoch kolekcií (nepotrebujeme vedieť implementačné detaily)
  - IEnumerator (iterator design pattern) umožňuje použitie foreach
  - LINQ (language integrated queries)
- Niektoré typy kolekcií:
  - List, Dictionary, HashSet, Queue, Stack, ArrayList, LinkedList, Hashtable…

List

- Rieši problém práce s poľami
  - Abstrahuje alokáciu / realokáciu miesta, prístup k prvkom
- Môžeme pristupovať k ľubovoľnému prvku
- Prvky sú zoradené v poradí, v ktorom boli pridané
- Jeden prvok môže byť zapísaný v zozname viac krát

Hashset

- Každý prvok je identifikovaný pomocou svojho hashu (`object.GetHashCode()`)
  - To znamená, že sa objekt nesmie meniť (immutable)
- Garantuje unikátnosť záznamov, nerieši poradie
- Určený na prácu s množinami
- Veľmi rýchle vyhľadávanie (blízko O(1))

Dictionary

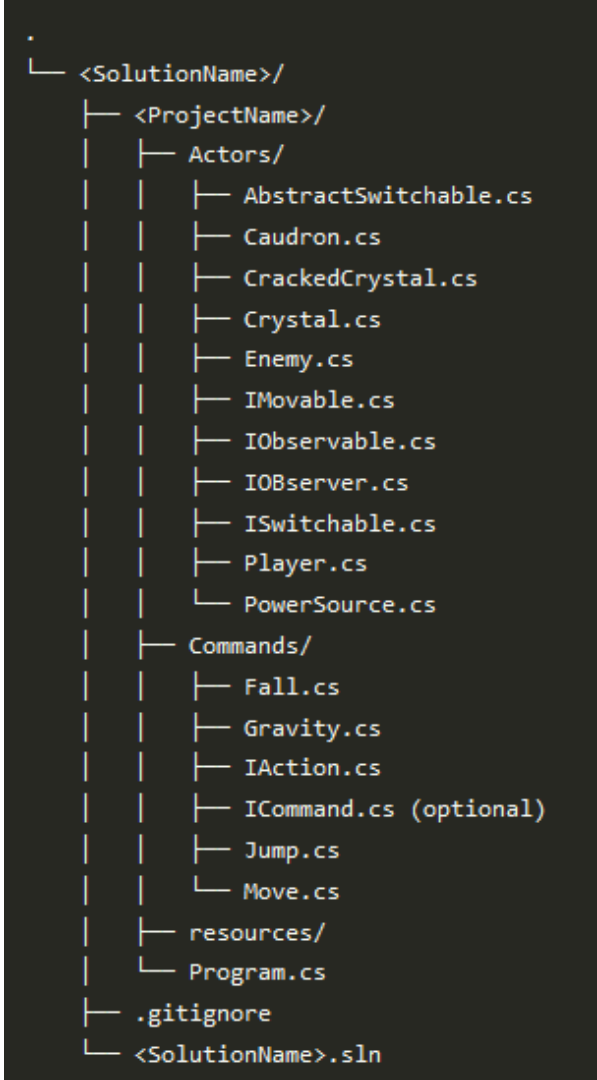
- Hodnoty sú uložené v tvare <kľúč, hodnota>
  - Kľúč musí byť unikátny, nesmie byť null, identifikuje záznam
    - Využíva sa hash, tzn. Platia rovnaké podmienky, ako pri Hashset
  - Rôzne kľúče môžu mať priradenú tú istú hodnotu, aj null (ak sú referenčné)
- Veľmi rýchle vyhľadávanie vďaka využitu hashovacej tabuľky

Queue & stack

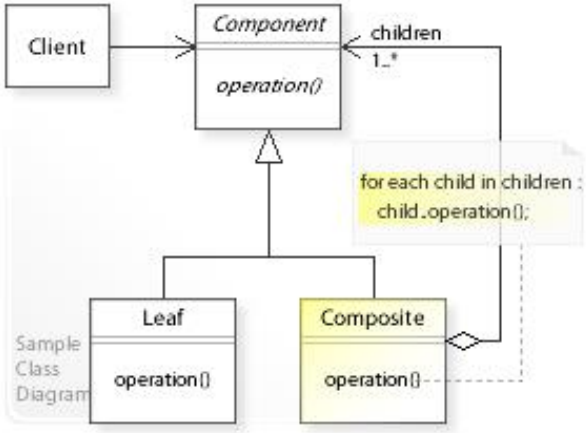
- Queue - FIFO zásobník
- Stack – LIFO zásobník
- Určené dočasné uloženie dát, napríklad pri implementácií vyrovnávacej pamäte (buffer), undo/redo, pri prehľadávaní stromov

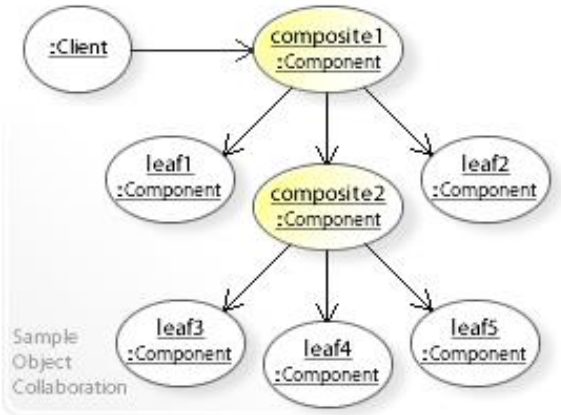
Stromy

- Koreňový uzol
- Medziľahlé uzly
- Listové uzly
- Composite pattern
  - Umožňuje reprezentovať objekt a jeho súčasti jednotným spôsobom
  - Kompozit je možné rozdeliť na menšie komponenty
  - 2 spôsoby implementácie
    1. Rozlišuje medzi listovým a nelistovým uzlom
    2. Všetky uzly používajú rovnakú implementáciu



Composite pattern





otázky?