Abstract Data Types forskellige collektions

Peter Levinsky, IT Roskilde

19.02.2024

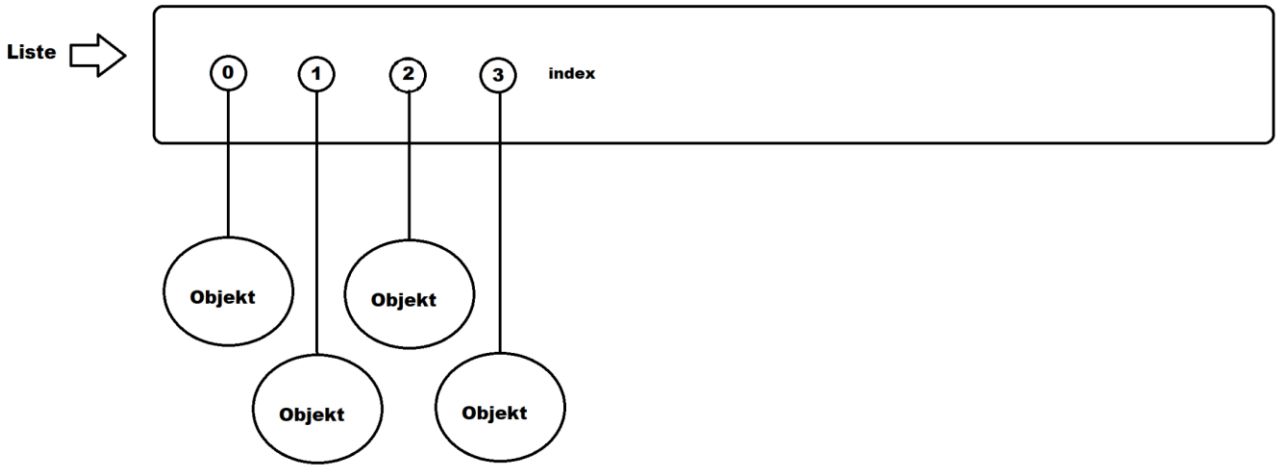
Abstrakte Data Typer (collections)

- Tidligere set:
  - List
  - Dictionary
  - Array
- Desuden
  - Enum

- Flere
  - LinkedList
  - Queue
  - Stack
  - HashSet

Datastucture: List

Ide: Opbevarer objekter (simple typer) I en rækkefølge (array ligge begved)



Brug:

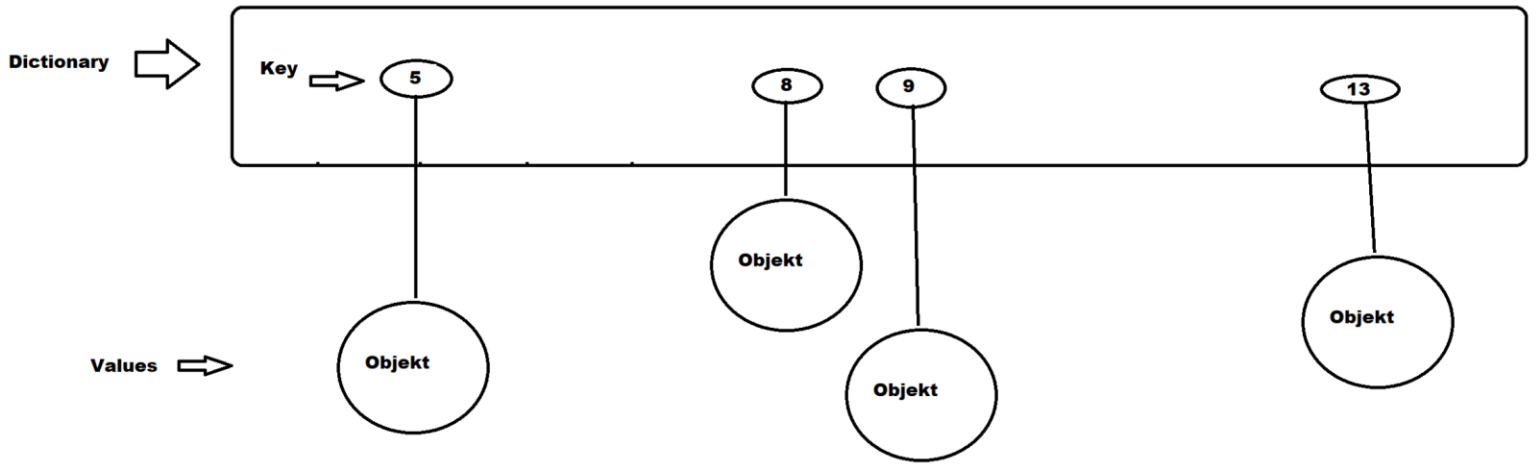
```
List<Person> personer = new List<Person>();

personer.Add(person); // tilføjer objekt
Person p = personer[2]; // henter index 2 dvs plads 3

foreach (Person p in personer){ … } // gennemløb
```

Datastucture: Dictionary

Ide: Opbevarer objekter (simple typer) I en slags ordbog men sorteret efter nøgler (Key)



Brug:

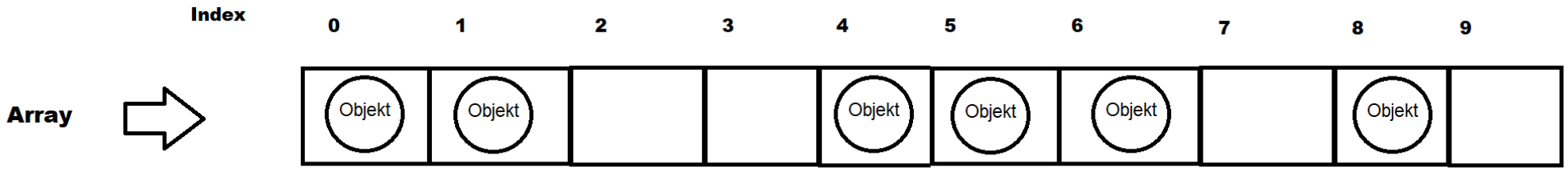
```
Dictionary<int, Person> personer = new Dictionary<int, Person>();

personer.Add(3, person); // tilføjer objekt til nøgle 3
Person p = personer[2]; // henter objekt til nøgle 2 – keyNotFoundException hvis ingen

foreach (Person p in personer.Values){ … } // gennemløb
```

Datastucture: Array

Ide: Opbevarer objekter (simple typer) I en buffer eller container – de behøver ikke at ligge samlet



Brug:

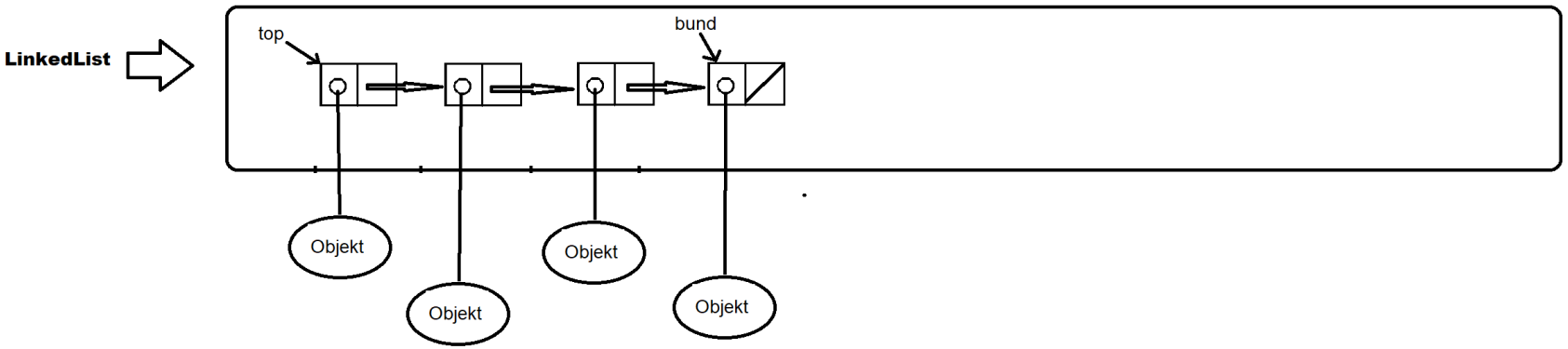
Person[] personer = new Person[10]; // der laves en buffer med plads til 10 objekter (0-9)

Personer[8] = person; // tilføjer objekt til index 8 (plads 9) programør skal selv styre index
Person p = personer[1]; // henter index 1 dvs plads 2

foreach (Person p in personer){ … } // gennemløb

Datastucture: LinkedList

Ide: Opbevarer objekter (simple typer) I en rækkefølge i en dynamisk liste



Brug:

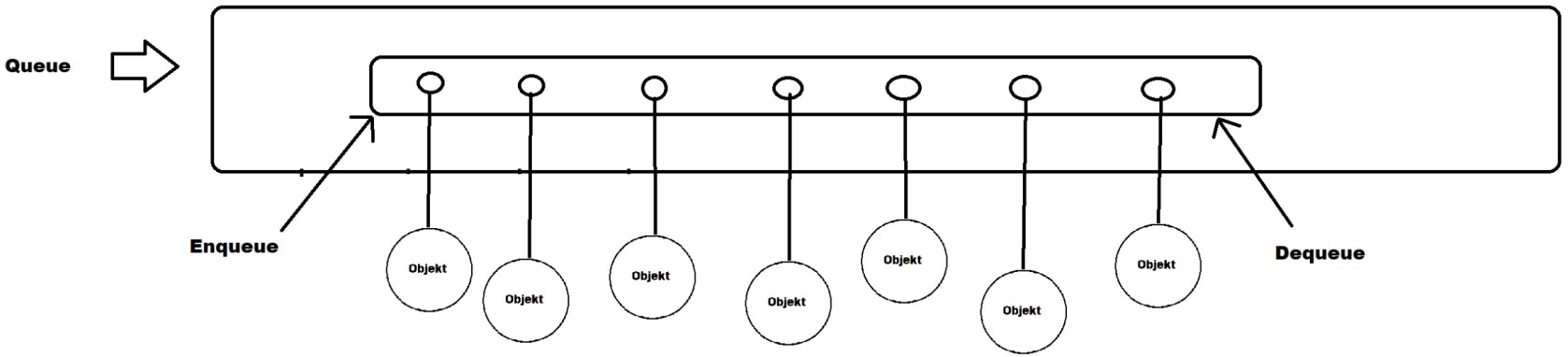
```
LinkedList<Person> personer = new LinkedList<Person>();

personer.Add(person); // tilføjer objekt
Person p = personer[2]; // henter index 2 dvs. plads 3

foreach (Person p in personer){ … } // gennemløb
```

Datastucture: Queue

Ide: Opbevarer objekter (simple typer) I en kø dvs. først I køen først ud (alm kø – kultur)



Brug:

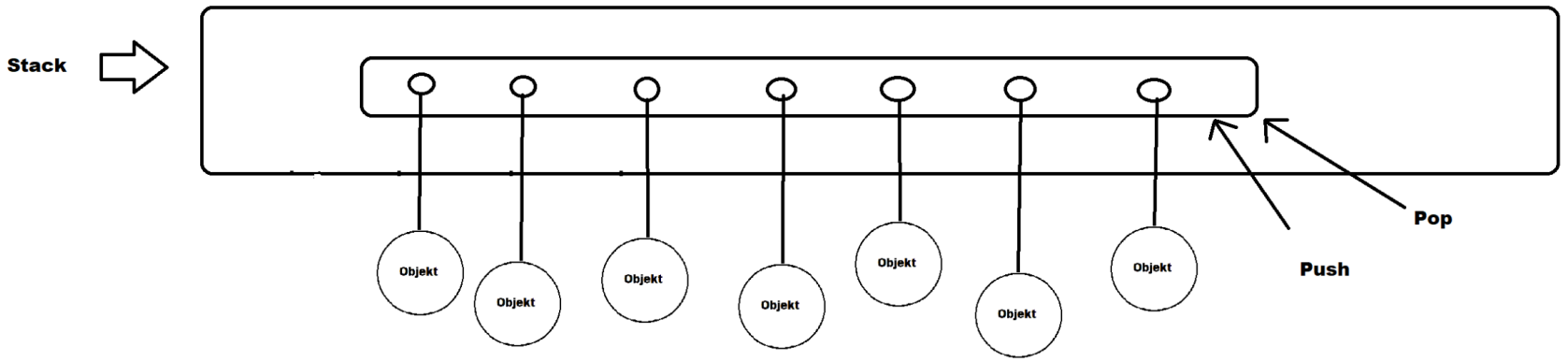
Queue<Person> personer = new Queue<Person>();

personer.Enqueue(person); // tilføjer objekt
Person p = personer. Dequeue() // henter den første I køen -- InvalidOperationException hvis køen er tom

foreach (Person p in personer){ … } // gennemløb

Datastucture: Stack

Ide: Opbevarer objekter (simple typer) I en stack dvs. først I stakken først ud (stabel tallerkener)



Brug:

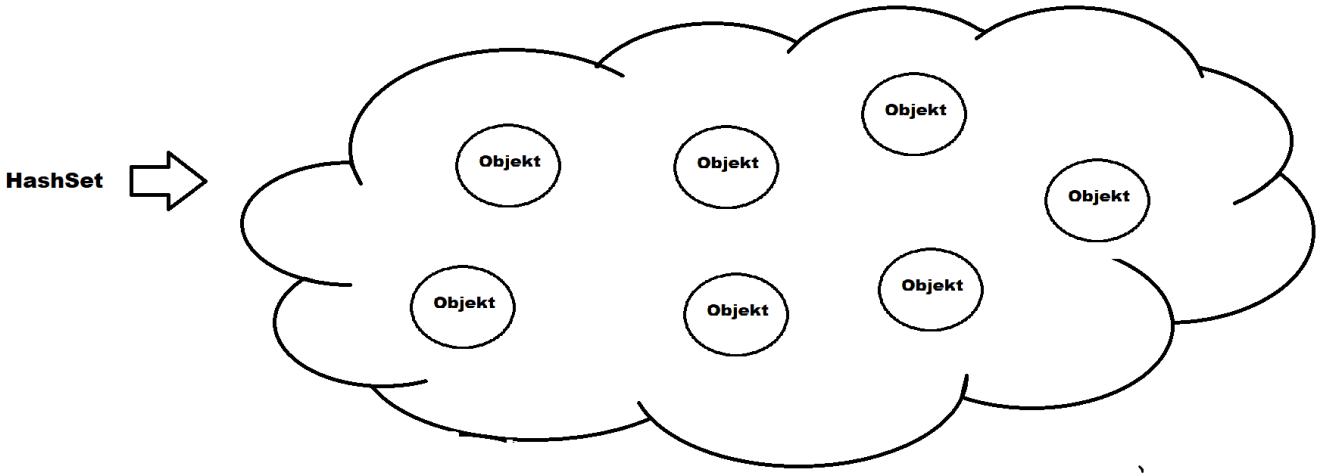
Stack<Person> personer = new Stack<Person>();

personer.Push(person); // tilføjer object øverst på stakken
Person p = personer.Pop() // henter den øverste I stakken (nyeste) -- InvalidOperationException hvis stakken er tom

foreach (Person p in personer){ … } // gennemløb

Datastucture: HashSet

Ide: Opbevarer objekter (simple typer) I en mængde ikke same rækkefølge og Objekt kun èn gang



Brug:

```
HashSet<Person> personer = new HashSet<Person>();

personer.Add(person); // tilføjer objekt
If (personer.TryGetValue(person, out helePerson) ) { … }  // finds ikke direkte – men kan prøve at finde udfra et objekt

foreach (Person p in personer){ … } // gennemløb
```

Opgave

Datastucture: Enum

Ide: Liste af mulige værdier

Brug:

```
public enum Kort { to, tre, fire, fem, seks, syv, otte, ni, ti, knægt, dame, konge, es }
```

Variable erklæring

```
Kort kort = Kort.fem;

int talværdi = Convert.ToInt32(kort); // = 3
```

Datastucture: Enum - forts

// hvis vi vil have tekst tallene svarer til tal-værdierne
public enum Kort { to=2, tre, fire, fem, seks, syv, otte, ni, ti, knægt, dame, konge, es }

// liste af værdier
List<Kort> korts = Enum.GetValues<Kort>().ToList();

// String to enum
Kort nytKort = Enum.Parse<Kort>("es"); // kaster en ArgumentException