ADSO
3. Posloupnosti a operace s nimi

(prezentace k učebnici)

Ivan Ryant

Agenda

- Základní pojmy
- Úloha abstraktních datových typů
- **Posloupnosti a operace s nimi**
- Vyhledávací datové struktury
- Vyhledávací posloupnost
- Algoritmy řazení
- Stromy
- Rozptýlené tabulky
- Prohledávání do hloubky a do šířky
- Práce s grafy
- Techniky návrhu efektivních algoritmů

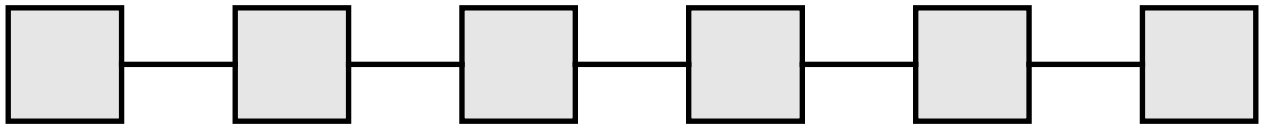
Agenda

- Základní pojmy
- Úloha abstraktních datových typů
- **Posloupnosti a operace s nimi**
  - Posloupnost jako ADT
  - Zásobník
  - Fronta
  - Obecná posloupnost
  - Implementace posloupnosti seznamem
  - Shrnutí
- Vyhledávací datové struktury
- Vyhledávací posloupnost
- Algoritmy řazení
- Stromy
- Rozptýlené tabulky
- Prohledávání do hloubky a do šířky
- Práce s grafy
- Techniky návrhu efektivních algoritmů

Posloupnosti a operace s nimi

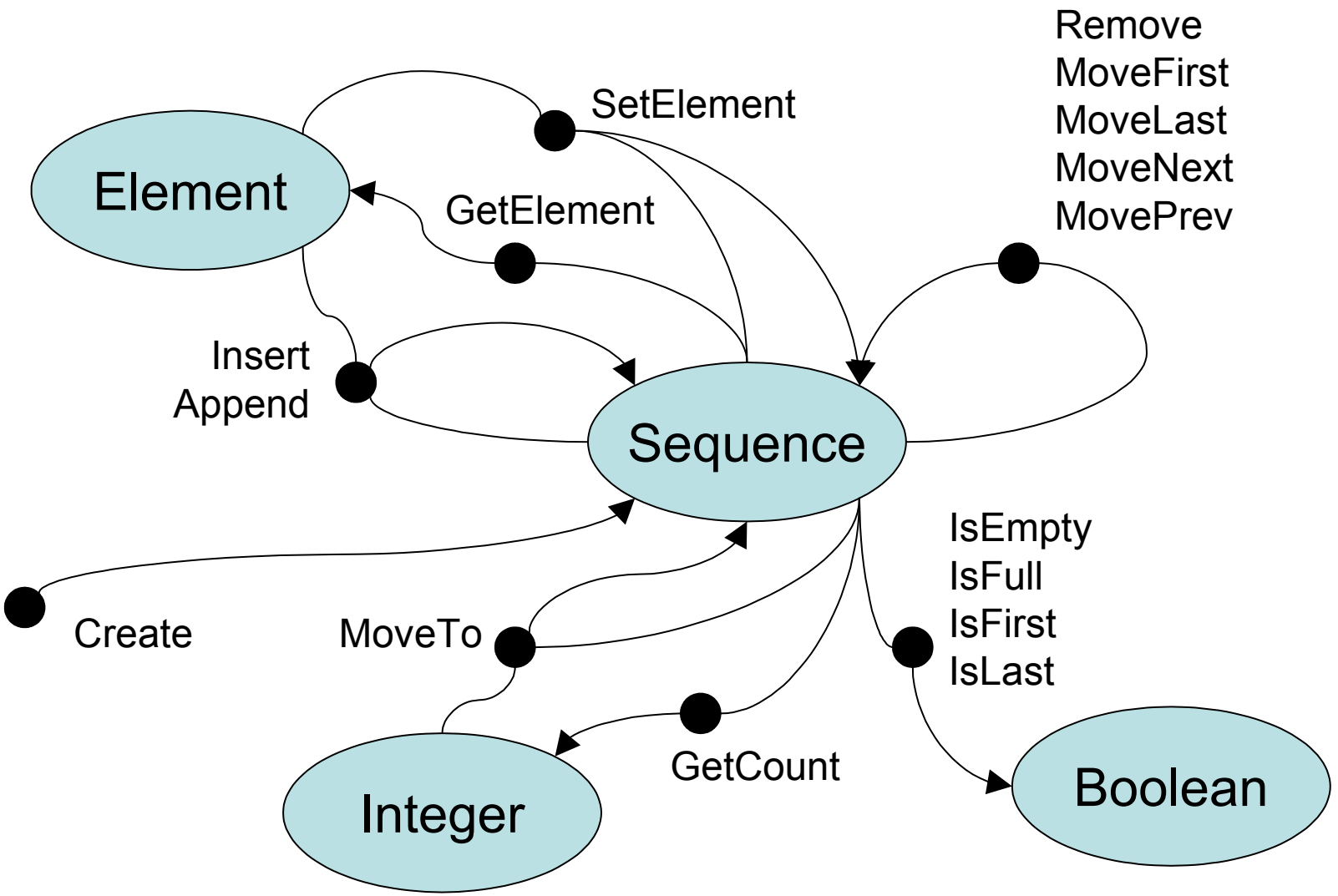
- **Posloupnost jako ADT**
- Zásobník
- Fronta
- Obecná posloupnost
- Implementace posloupnosti seznamem
- Shrnutí

Posloupnost jako ADT



Posloupnost je tvořena prvky, které jsou uspořádány lineárně za sebou.

Posloupnost jako ADT



Signatura datového typu *posloupnost*

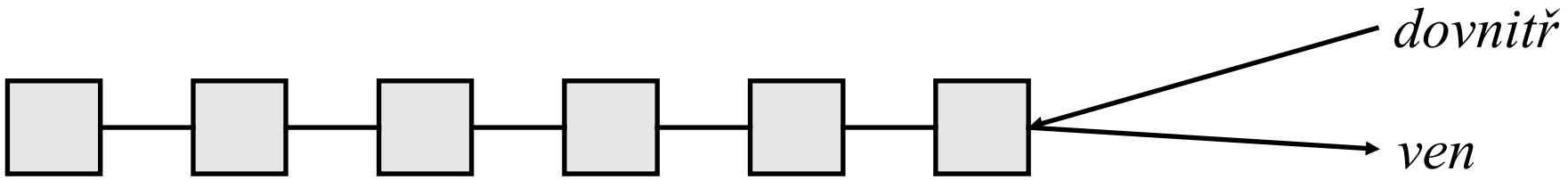
Posloupnost jako ADT

- Axiomy ADT *posloupnost* jsou uvedeny až u obecné posloupnosti

Posloupnosti a operace s nimi

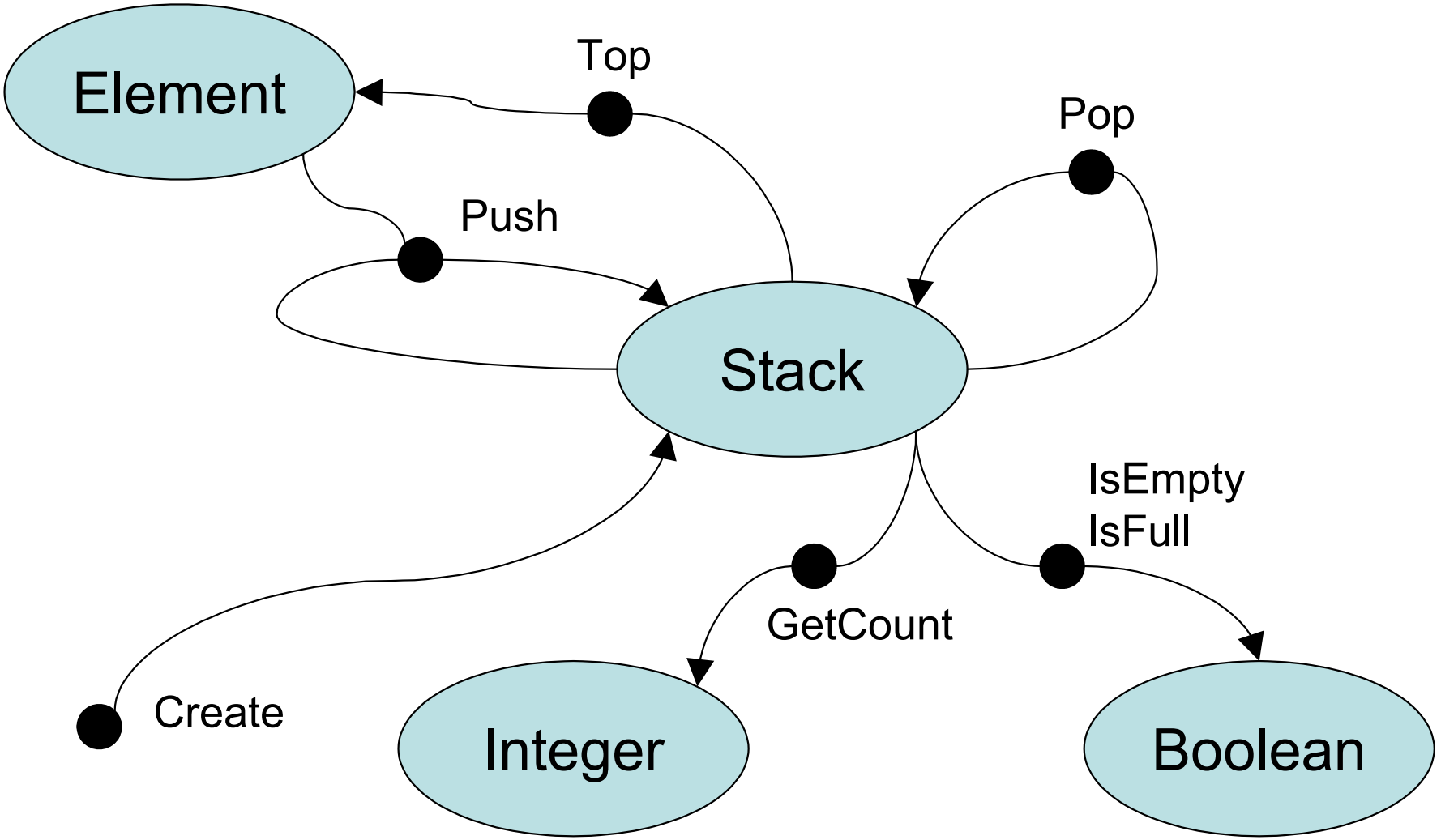
- Posloupnost jako ADT
- **Zásobník**
  - Příklad: polská kalkulačka
  - Implementace obecného zásobníku
  - Implementace zásobníku polem
  - Implementace zásobníku spojovým seznamem
- Fronta
- Obecná posloupnost
- Implementace posloupnosti seznamem
- Shrnutí

Zásobník



Zásobník je posloupnost, do které se přidává vždy na týž konec a z toho konce se taky vždy odebírá.

Zásobník



Signatura datového typu *zásobník*

Zásobník

$\forall s \in$ Stack, $\forall x \in$ Element:

Pop (Create) = Error
Pop (Push(*s*, *x*)) = *s*
Top (Create) = Error
Top (Push (*s*, *x*)) = *x*
IsEmpty (Create) = True
IsEmpty (Push (*s*, *x*)) = False
IsFull (*s*) = False
GetCount (Create) = 0
GetCount (Push (*s*, *x*)) = GetCount (*s*) + 1

Axiomy datového typu *zásobník*

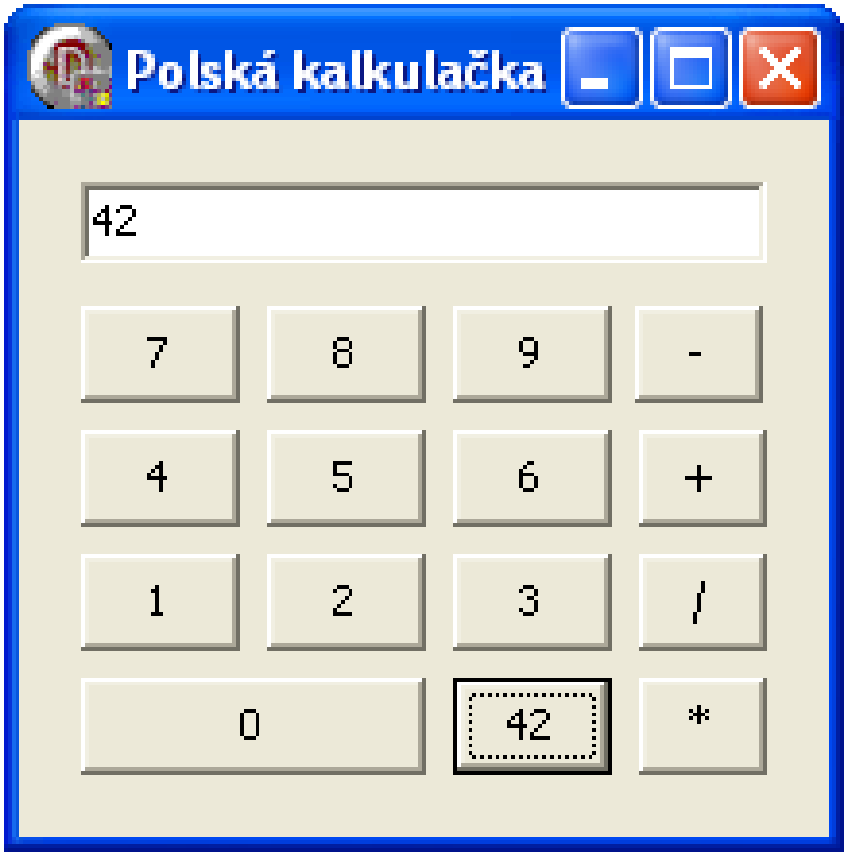
Zásobník

- Kontrakty metod odvodíme z axiomů, např.:
  - Pop ()
    - Vyžaduje: ¬IsEmpty
    - Splní: ano
    - Mění: Odstraní prvek z vršku zásobníku.
  - Top (): Element
    - Vyžaduje: ¬IsEmpty
    - Splní: Vrátí prvek na vršku zásobníku.
    - Mění: Nic nemění.
  - Push (x: Element)
    - Vyžaduje: ¬IsFull
    - Splní: ano
    - Mění: Přidá prvek x na vršek zásobníku.

Zásobník

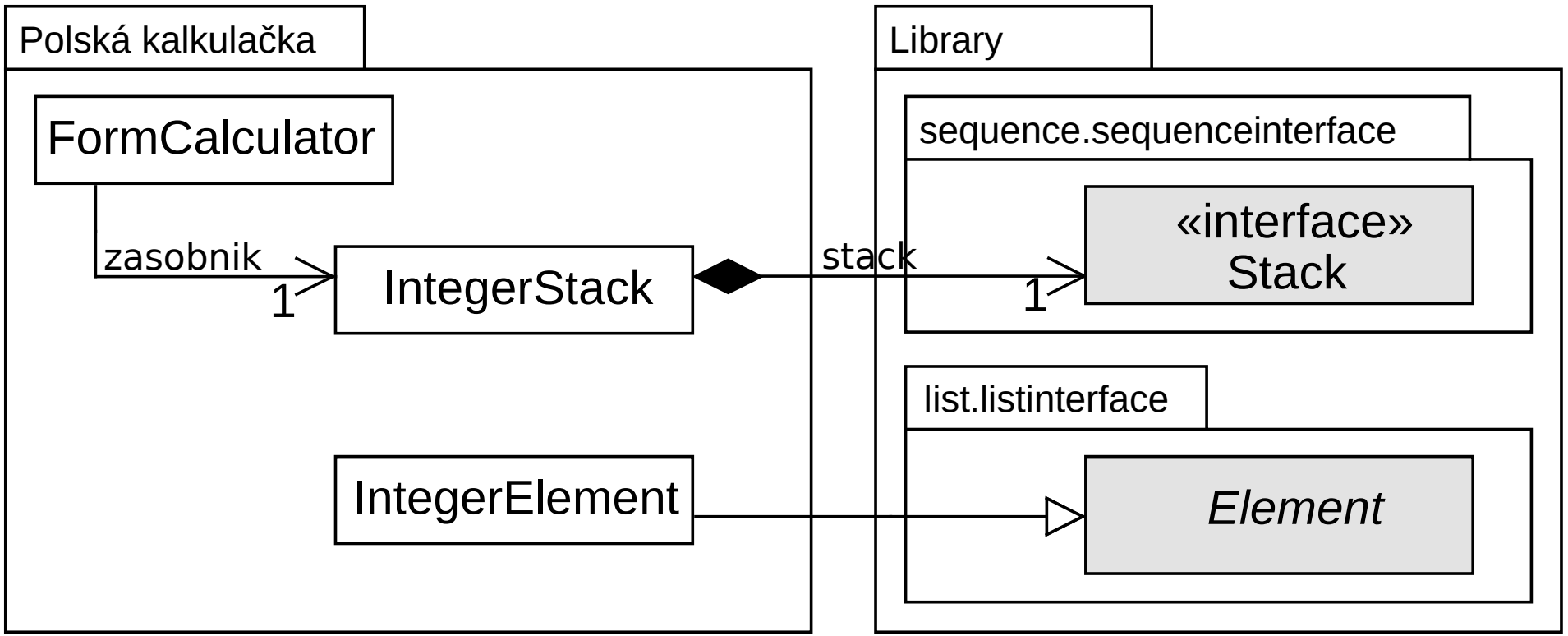
- **Příklad: polská kalkulačka**
- Implementace obecného zásobníku
- Implementace zásobníku polem
- Implementace zásobníku spojovým seznamem

Příklad: polská kalkulačka



Formulář programu „Polská kalkulačka“

Příklad: polská kalkulačka



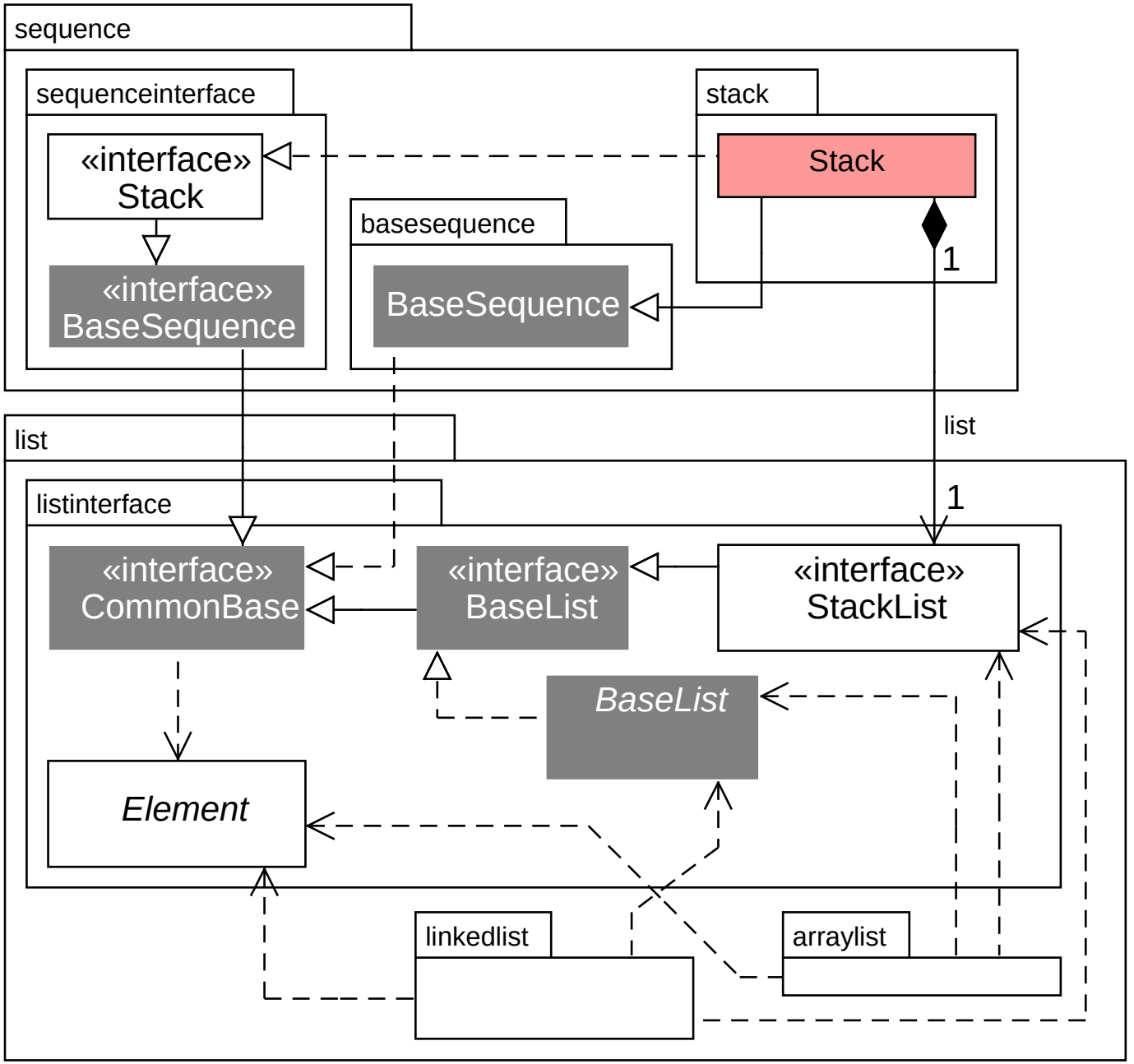
Design aplikace „Polská kalkulačka“

Zásobník

- Příklad: polská kalkulačka
- **Implementace obecného zásobníku**
- Implementace zásobníku polem
- Implementace zásobníku spojovým seznamem

Implementace obecného zásobníku

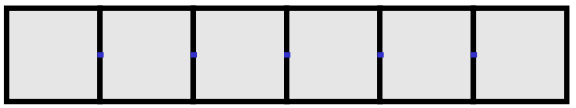
Design obecného zásobníku `Stack`



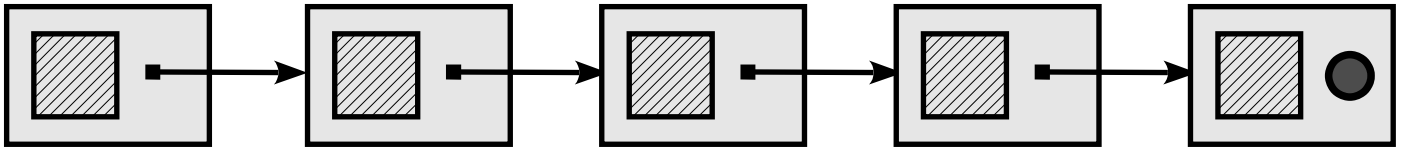
Implementace obecného zásobníku

- Schematické znázornění pole:

```
indexy: 0 1 2 3 4 5
```



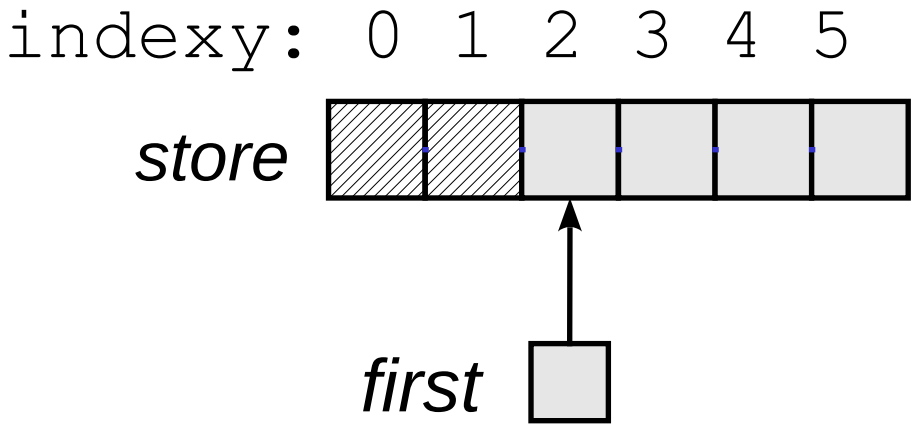
- Schematické znázornění spojového seznamu:



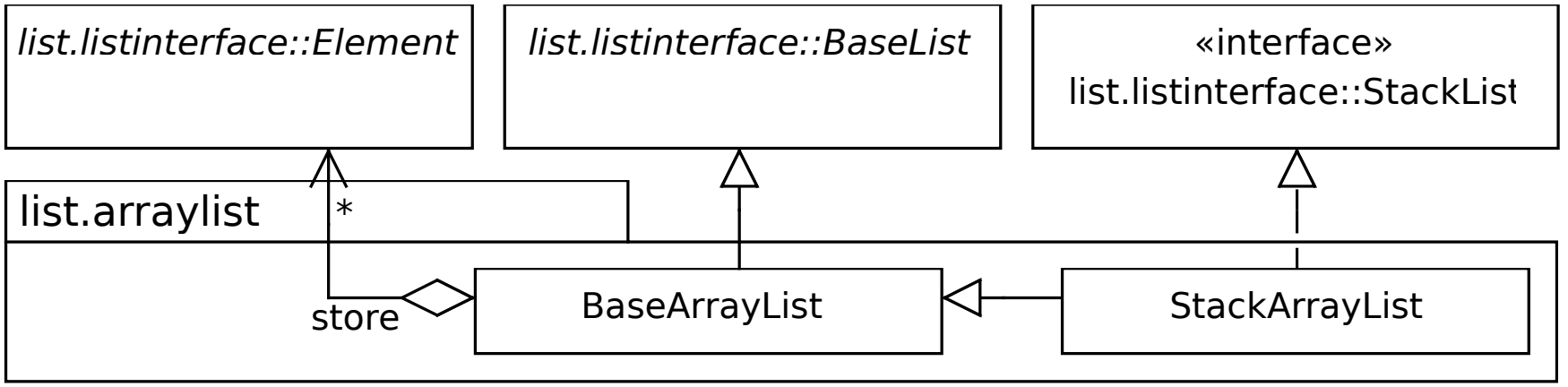
Zásobník

- Příklad: polská kalkulačka
- Implementace obecného zásobníku
- **Implementace zásobníku polem**
- Implementace zásobníku spojovým seznamem

Implementace zásobníku polem



Schematické znázornění implementace zásobníku polem

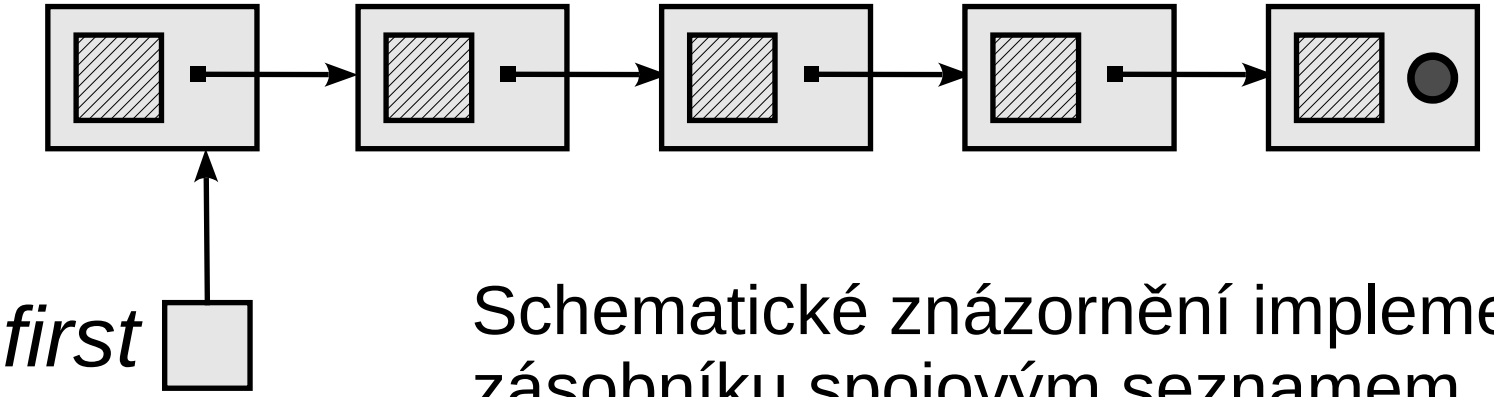


Design pole pro implementaci zásobníku

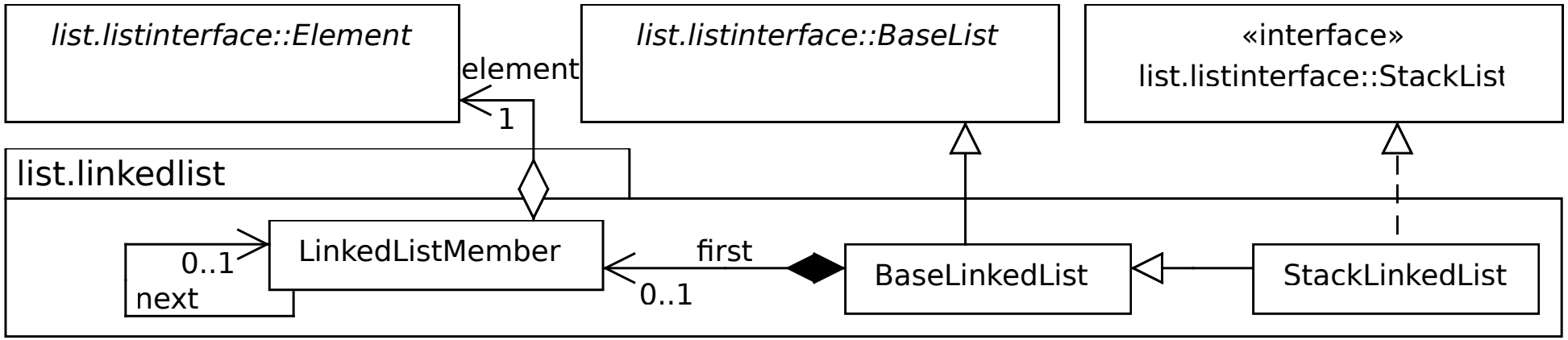
Zásobník

- Příklad: polská kalkulačka
- Implementace obecného zásobníku
- Implementace zásobníku polem
- **Implementace zásobníku spojovým seznamem**

Implementace zásobníku spojovým seznamem



Schematické znázornění implementace zásobníku spojovým seznamem

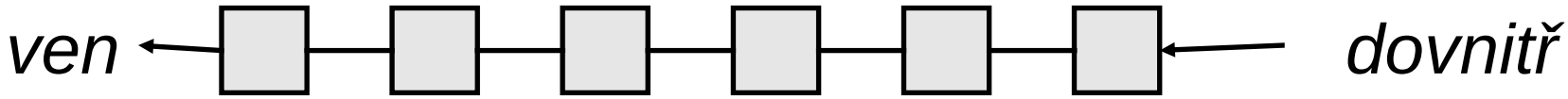


Design spojového seznamu pro implementaci zásobníku

Posloupnosti a operace s nimi

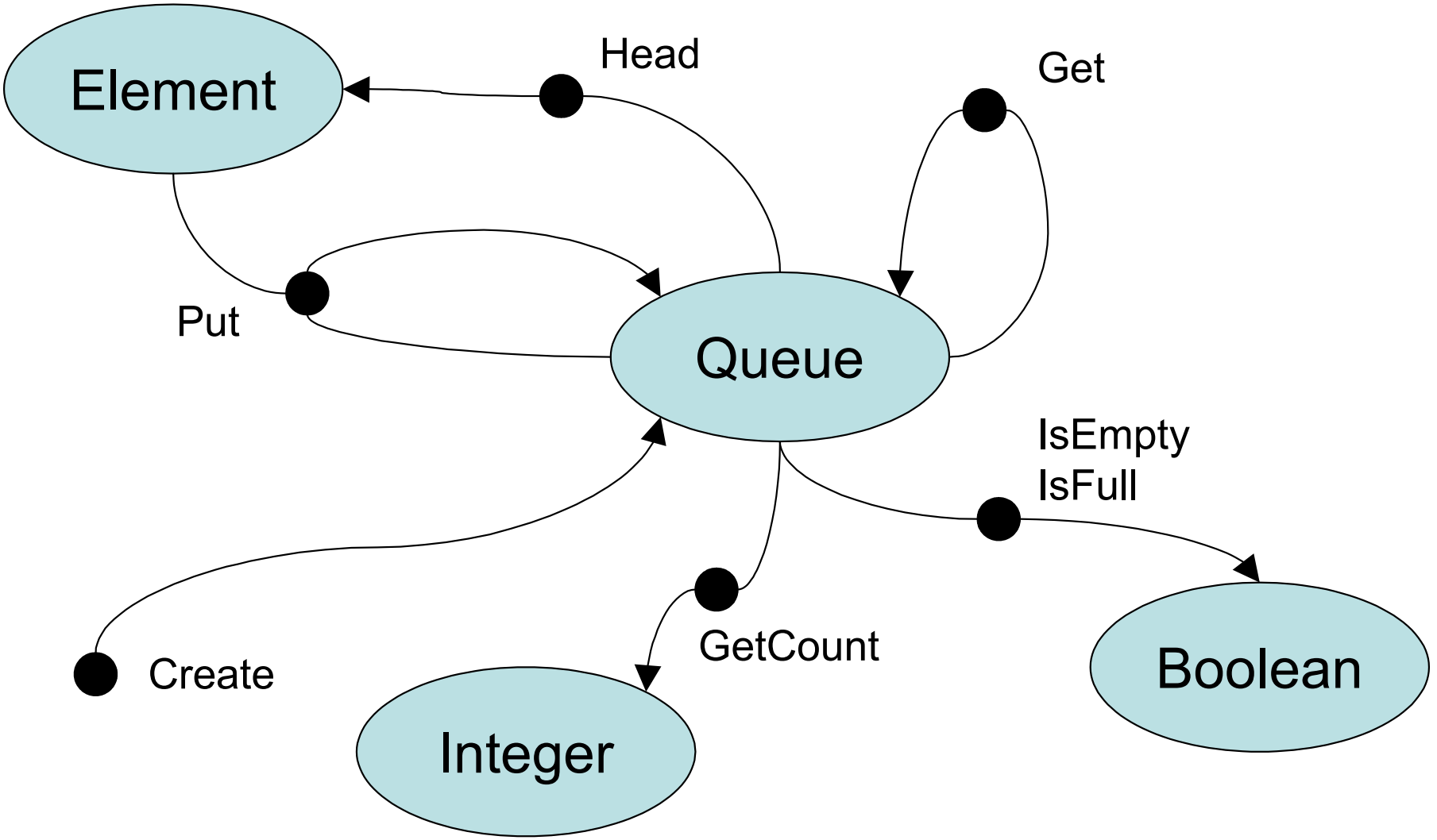
- Posloupnost jako ADT
- Zásobník
- **Fronta**
  - Příklad: klouzavý průměr
  - Implementace fronty polem
  - Implementace fronty spojovým seznamem
  - Implementace fronty dvojicí zásobníků
- Obecná posloupnost
- Implementace posloupnosti seznamem
- Shrnutí

Fronta



Fronta je posloupnost, do které se prvky přidávají vždy na konec a odebírají se vždy ze začátku.

Fronta



Signatura datového typu *fronta*

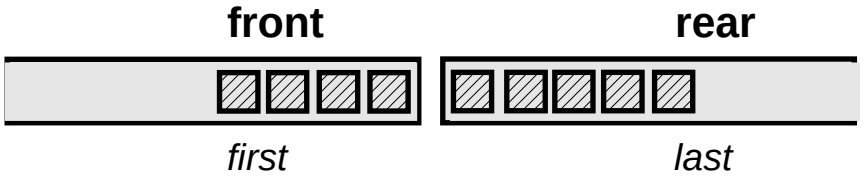
Fronta

$\forall q \in$ Queue, $\forall x, y \in$ Element:

Get (Create) = Error
Get (Put (Create, x)) = Create
Get (Put (Put (q, x), y)) = Put (Get (Put (q, x)), y)
Head (Create) = Error
Head (Put (Create, x)) = x
Head (Put (Put (q, x), y)) = Head (Put (q, x))
GetCount (Create) = 0
GetCount (Put (q, x)) = GetCount (q) + 1
IsEmpty (Create) = True
IsEmpty (Put (q, x)) = False
IsFull (q) = False

Axiomy datového typu *fronta*

Fronta



Specifikace datového typu *fronta* pomocí dvou zásobníků

$\forall q \in$ Queue, $\forall$*front, rear* $\in$ Stack, $\forall x \in$ Element:

Create = [Stack.Create, Stack.Create]
Put ([front, rear], x) = [front, Push (rear, x)]
Get (Create) = Error
Get ([Push (front, x), rear]) = [front, rear]
Get (unbury ([front, Stack.Create])) = Get ([front, Stack.Create])
Get (unbury ([front, Push (rear, x)]))
= Get (unbury ([Push (front, x), Pop (Push (rear, x))]))
Get ([Stack.Create, Push (rear, x)])
= Get (unbury ([Stack.Create, Push (rear, x)]))
Head (Create) = Error
Head ([Push (front, x), rear]) = x
Head (unbury ([front, Stack.Create])) = Head ([front, Stack.Create])
Head (unbury ([front, Push (rear, x)]))
= Head (unbury ([Push (front, x), Pop (Push (rear, x))]))
Head ([Stack.Create, Push (rear, x)])
= Head (unbury ([Stack.Create, Push (rear, x)]))

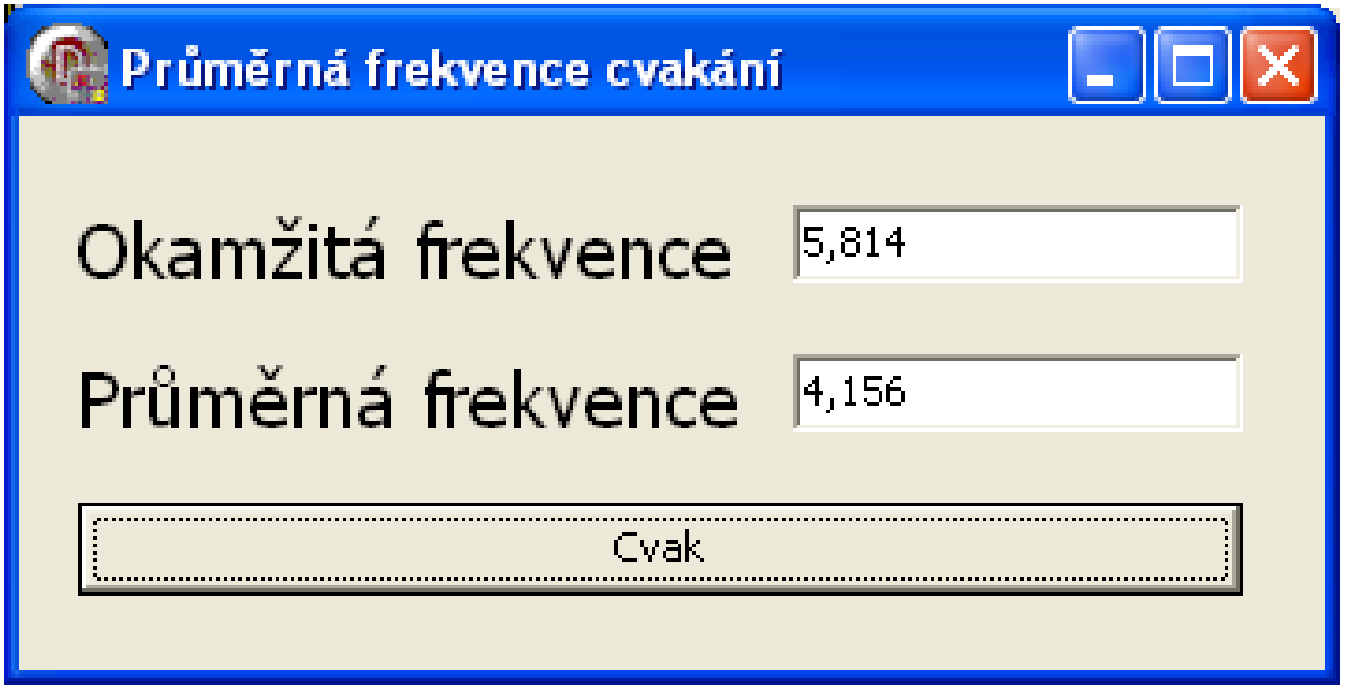
Posloupnosti a operace s nimi

- Posloupnost jako ADT
- Zásobník
- **Fronta**
  - Příklad: klouzavý průměr
  - Implementace fronty polem
  - Implementace fronty spojovým seznamem
  - Implementace fronty dvojicí zásobníků
- Obecná posloupnost
- Implementace posloupnosti seznamem
- Shrnutí

Fronta

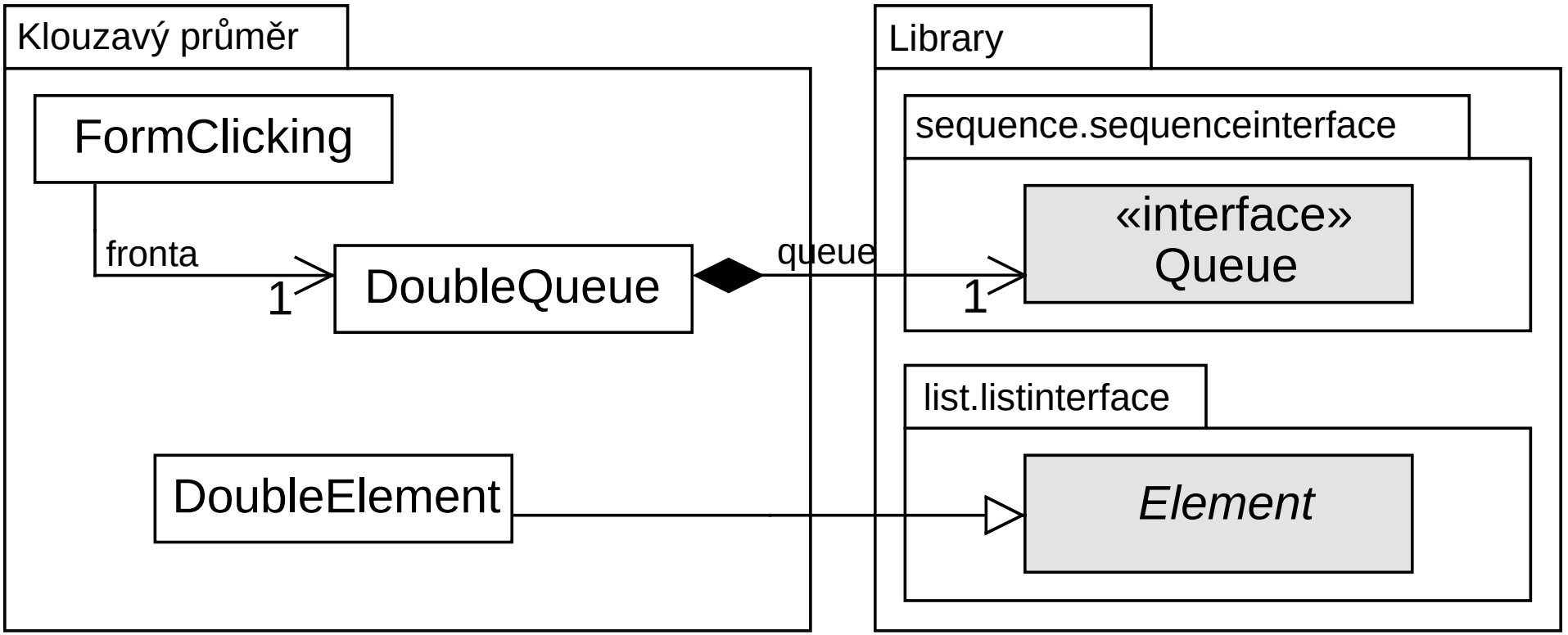
- **Příklad: klouzavý průměr**
- Implementace fronty polem
- Implementace fronty spojovým seznamem
- Implementace fronty dvojicí zásobníků

Příklad: klouzavý průměr



Formulář programu „*Klouzavý průměr*“

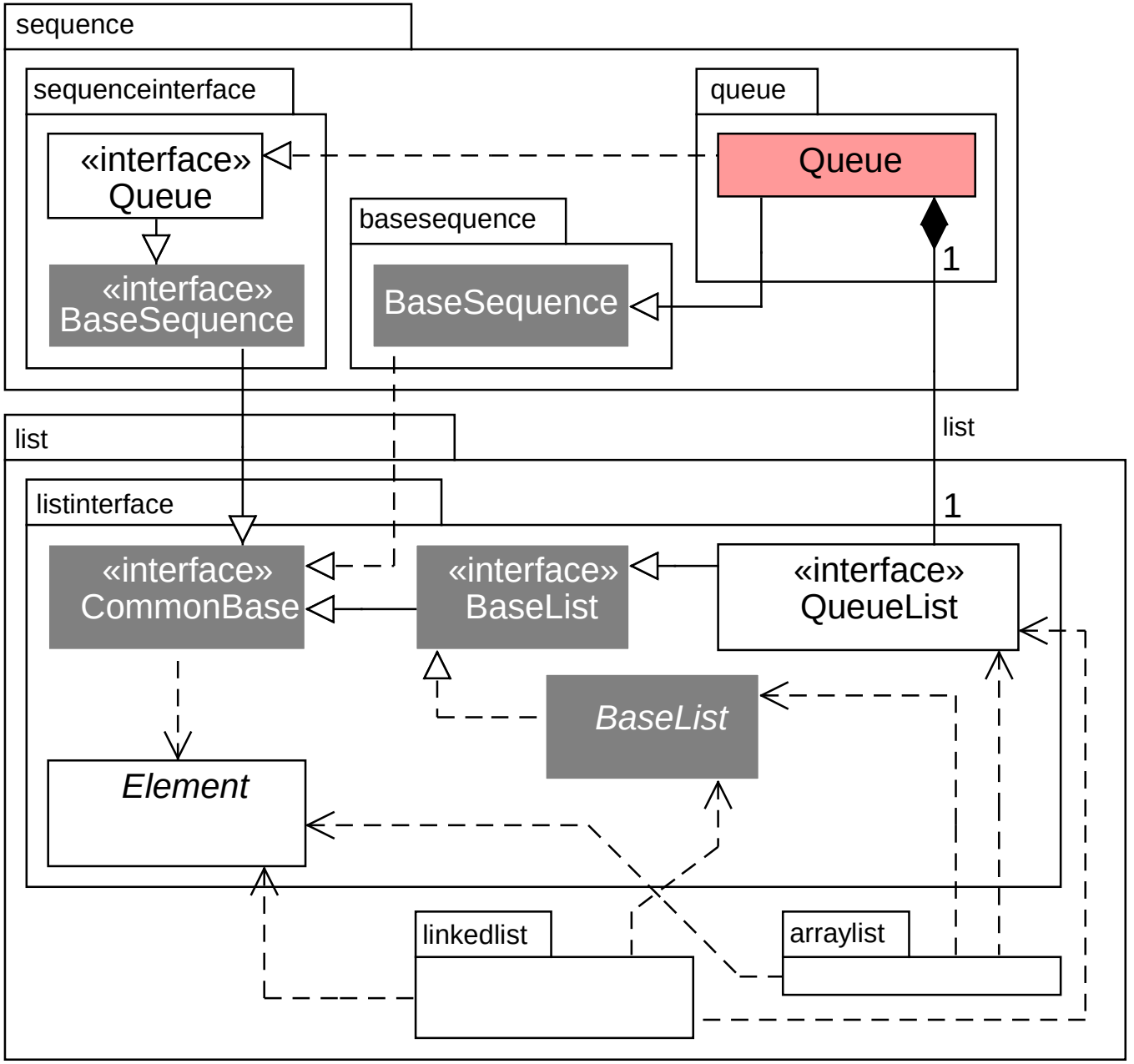
Příklad: klouzavý průměr



Design aplikace „*Klouzavý průměr*“

Design obecné fronty

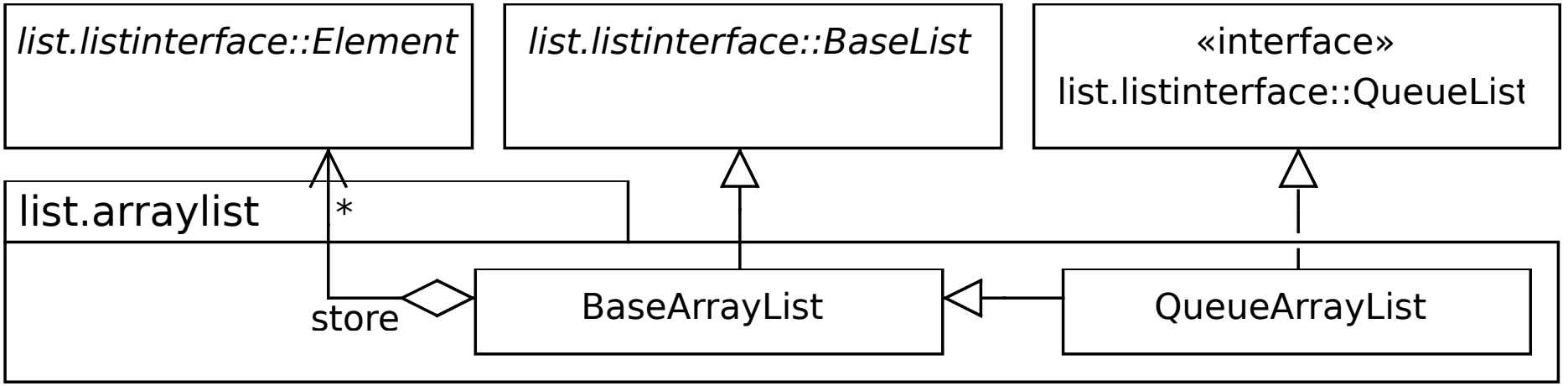
Design obecné fronty Queue



Fronta

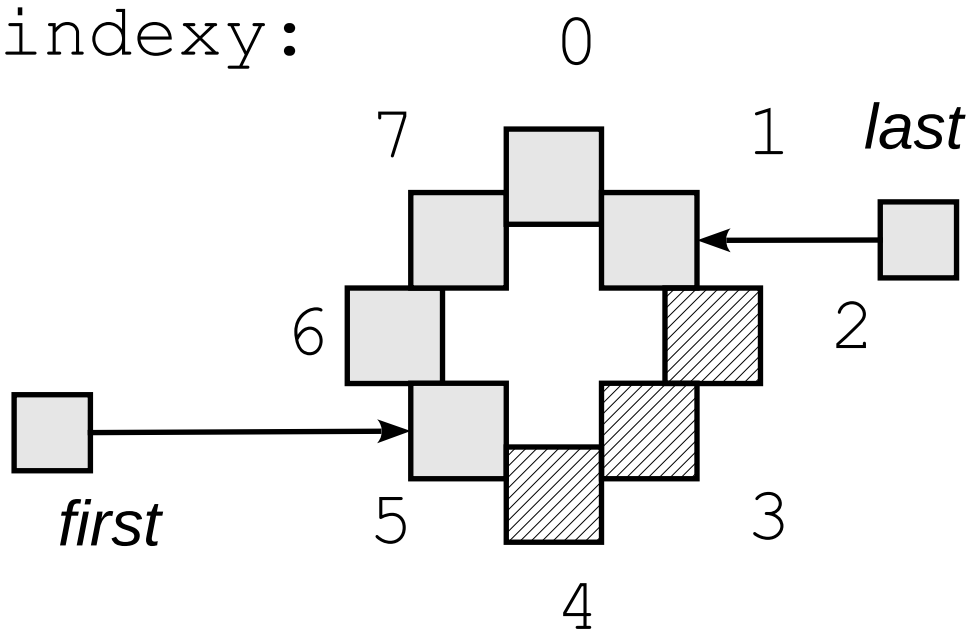
- Příklad: klouzavý průměr
- **Implementace fronty polem**
- Implementace fronty spojovým seznamem
- Implementace fronty dvojicí zásobníků

Implementace fronty polem



Design pole pro implementaci fronty

Implementace fronty polem

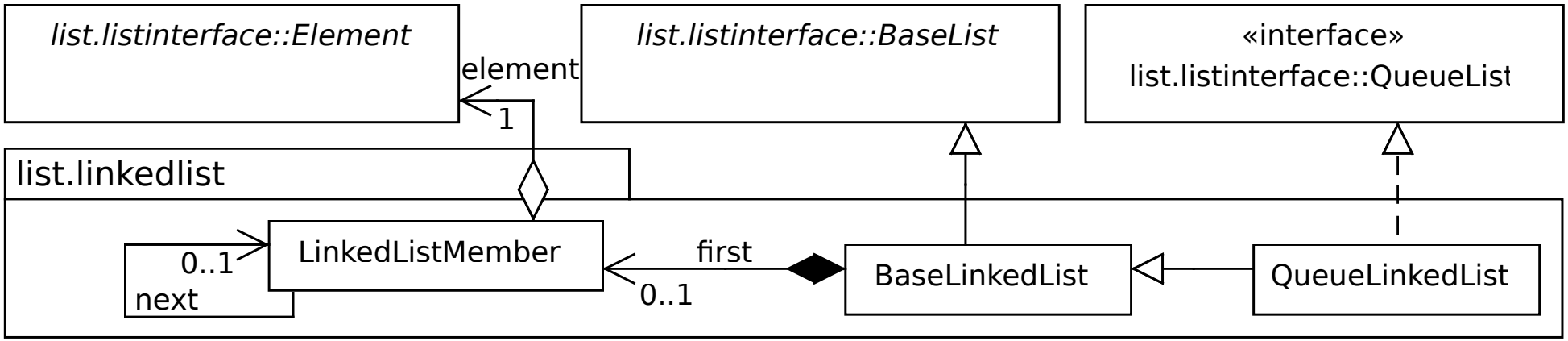


Znázornění implementace fronty polem (do kruhu)

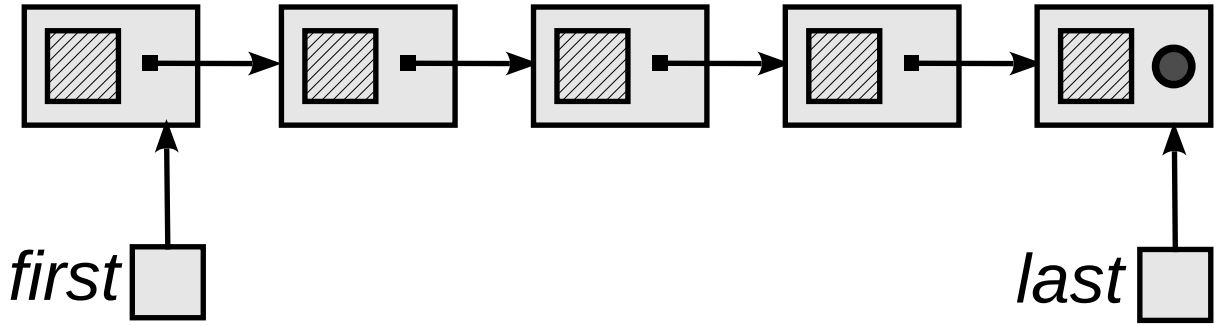
Fronta

- Příklad: klouzavý průměr
- Implementace fronty polem
- **Implementace fronty spojovým seznamem**
- Implementace fronty dvojicí zásobníků

Implementace fronty spojovým seznamem



Design spojového seznamu pro implementaci fronty

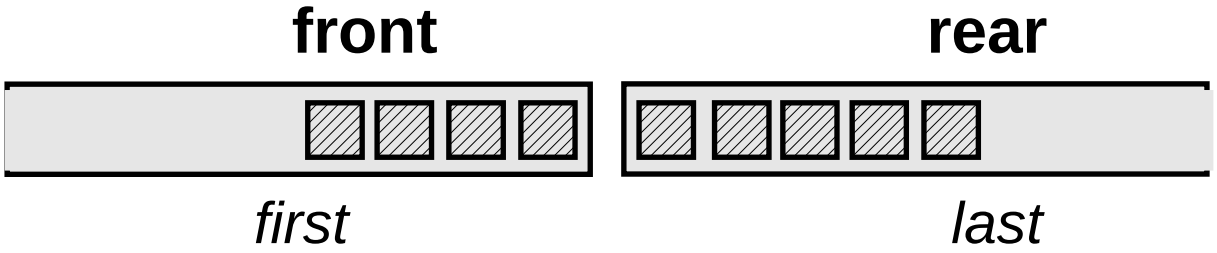


Znázornění implementace fronty spojovým seznamem

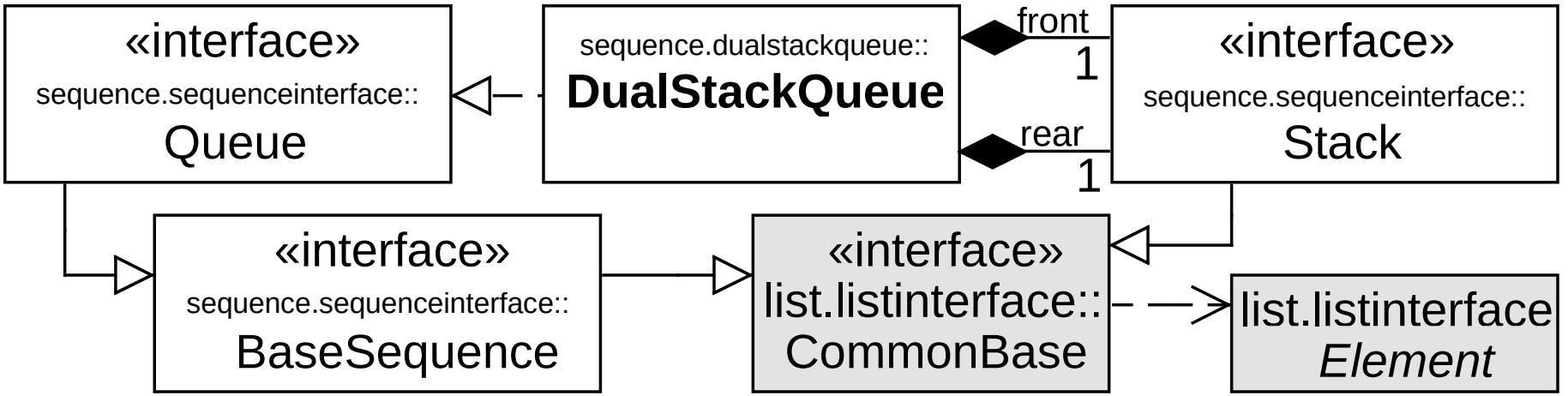
Fronta

- Příklad: klouzavý průměr
- Implementace fronty polem
- Implementace fronty spojovým seznamem
- **Implementace fronty dvojicí zásobníků**

Implementace fronty dvojicí zásobníků



Znázornění fronty implementované dvěma zásobníky



Design fronty implementované dvojicí zásobníků

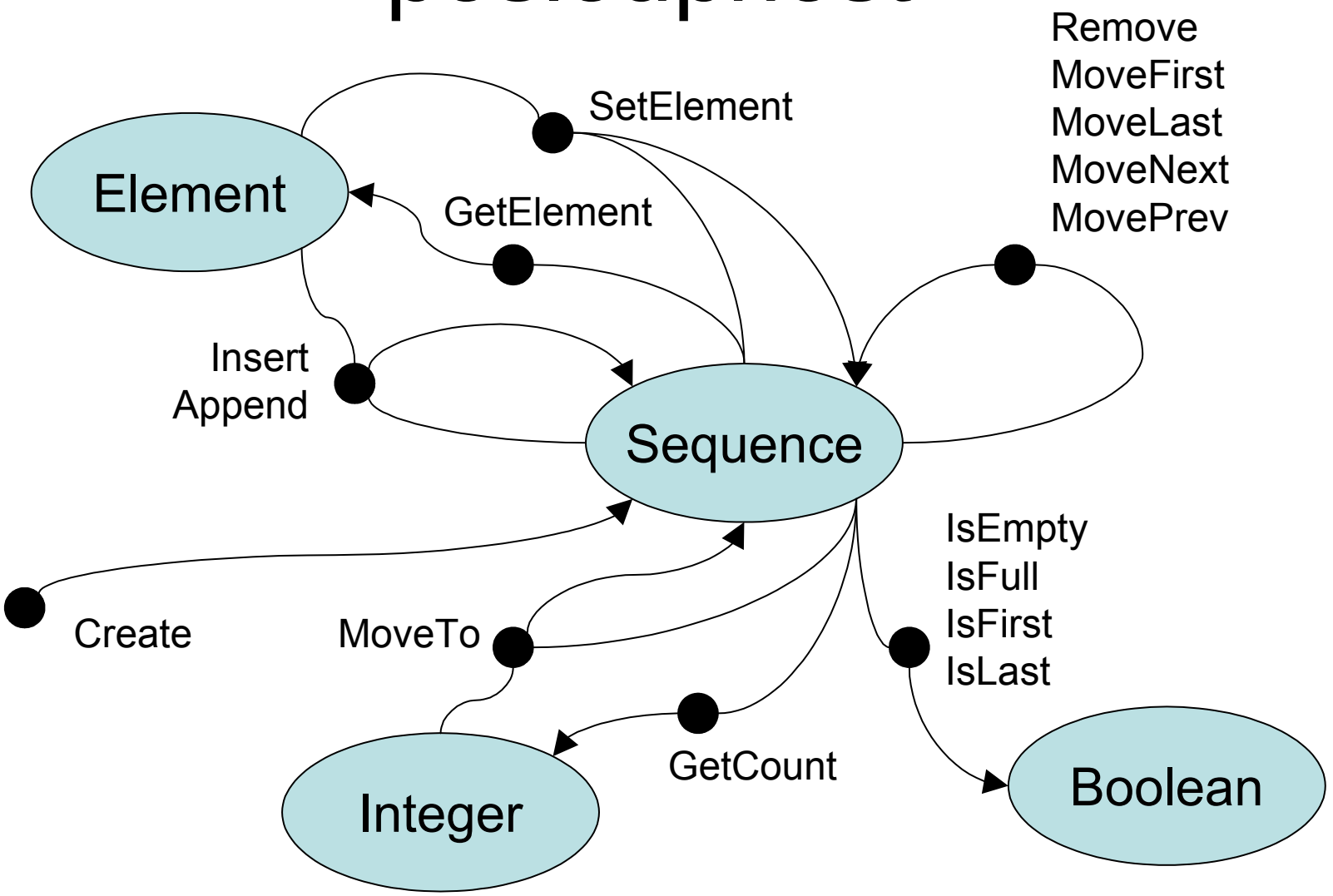
Posloupnosti a operace s nimi

- Posloupnost jako ADT
- Zásobník
- Fronta
- **Obecná posloupnost**
  - Specifikace datového typu posloupnost
  - Příklad: směrová čísla
  - Implementace posloupnosti dvojicí zásobníků
- Implementace posloupnosti seznamem
- Shrnutí

Obecná posloupnost

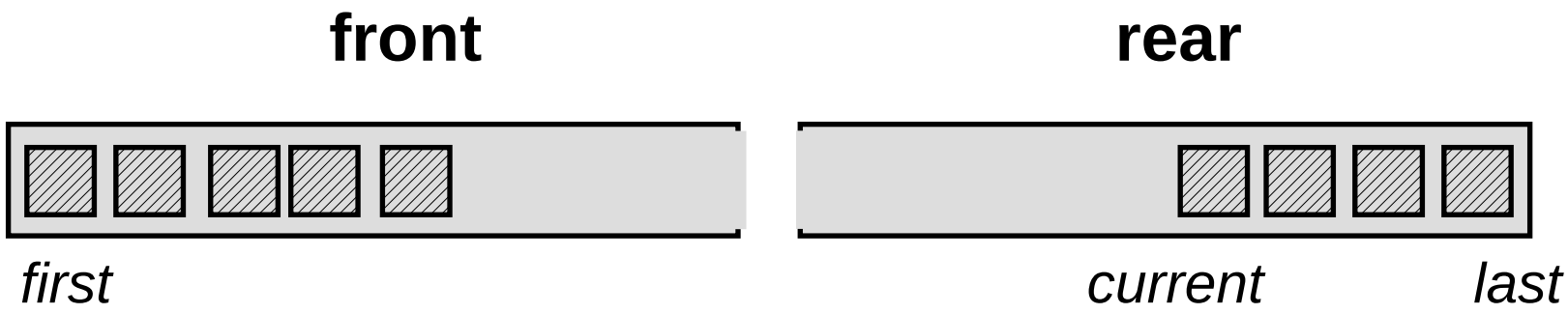
- **Specifikace datového typu posloupnost**
- Příklad: směrová čísla
- Implementace posloupnosti dvojicí zásobníků

Specifikace datového typu posloupnost



Signatura datového typu *posloupnost*

Specifikace datového typu posloupnost



Znázornění posloupnosti implementované dvěma zásobníky

Specifikace datového typu posloupnost

$\forall s \in$ Sequence, $\forall$*front, rear* $\in$ Stack, $\forall x, y \in$ Element:

Create = [Stack.Create, Stack.Create]
Insert ([*front, rear*], *x*) = [*front,* Push (*rear, x*)]

Append ([*front,* Stack.Create], *y*) = [*front,* Push (*rear, y*)]
Append ([*front,* Push (*rear, x*)], *y*) = Append ([Push (*front, x*), *rear*], *y*)

Remove (Create) = Error
Remove ([Stack.Create, Push (Stack.Create, *x*)]) = Create
Remove ([Push (*front, x*), Push (Stack.Create, *y*)]) = [*front,* Push (Stack.Create, *x*)]
Remove ([*front,* Push (Push (*rear, x*), *y*)]) = [*front,* Push (*rear, x*)]

IsEmpty ([*front, rear*]) = IsEmpty (*rear*)
IsFull (*s*) = False

GetElement (Create) = Error
GetElement ([*front,* Push (*rear, x*)]) = *x*

SetElement (Create) = Error
SetElement ([*front,* Push (*rear, x*)], *y*) = [*front,* Push (*rear, y*)]

Specifikace datového typu posloupnost

IsFirst ([*front,* Stack.Create]) = Error
IsFirst ([*front,* Push (*rear, x*)]) = Stack.IsEmpty (*front*)

IsLast ([*front,* Stack.Create]) = Error
IsLast ([*front,* Push (*rear, x*)]) = Stack.IsEmpty (*rear*)

MoveFirst ([*front,* Stack.Create]) = Error
MoveFirst ([Stack.Create, Push (*rear, x*)]) = [Stack.Create, Push (*rear, x*)]
MoveFirst ([Push (*front, x*), *rear*]) = MoveFirst ([*front,* Push (*rear, x*)])

MoveLast ([*front,* Stack.Create]) = Error
MoveLast ([*front,* Push (Stack.Create, *x*)]) = [*front,* Push (Stack.Create, *x*)]
MoveLast ([*front,* Push (Push (*rear,x*), *y*)]) = MoveLast ([Push (*front,y*), Push (*rear,x*)])

MoveNext ([*front,* Stack.Create]) = Error
MoveNext ([*front,* Push (Stack.Create, *x*)]) = Error
MoveNext ([*front,* Push (Push (*rear, x*), *y*)]) = [Push (*front, y*), Push (*rear, x*)]

MovePrev ([*front,* Stack.Create]) = Error
MovePrev ([Stack.Create, *rear*]) = Error
MovePrev ([Push (*front, x*); *rear*]) = [*front,* Push (*rear, x*)]

Specifikace datového typu posloupnost

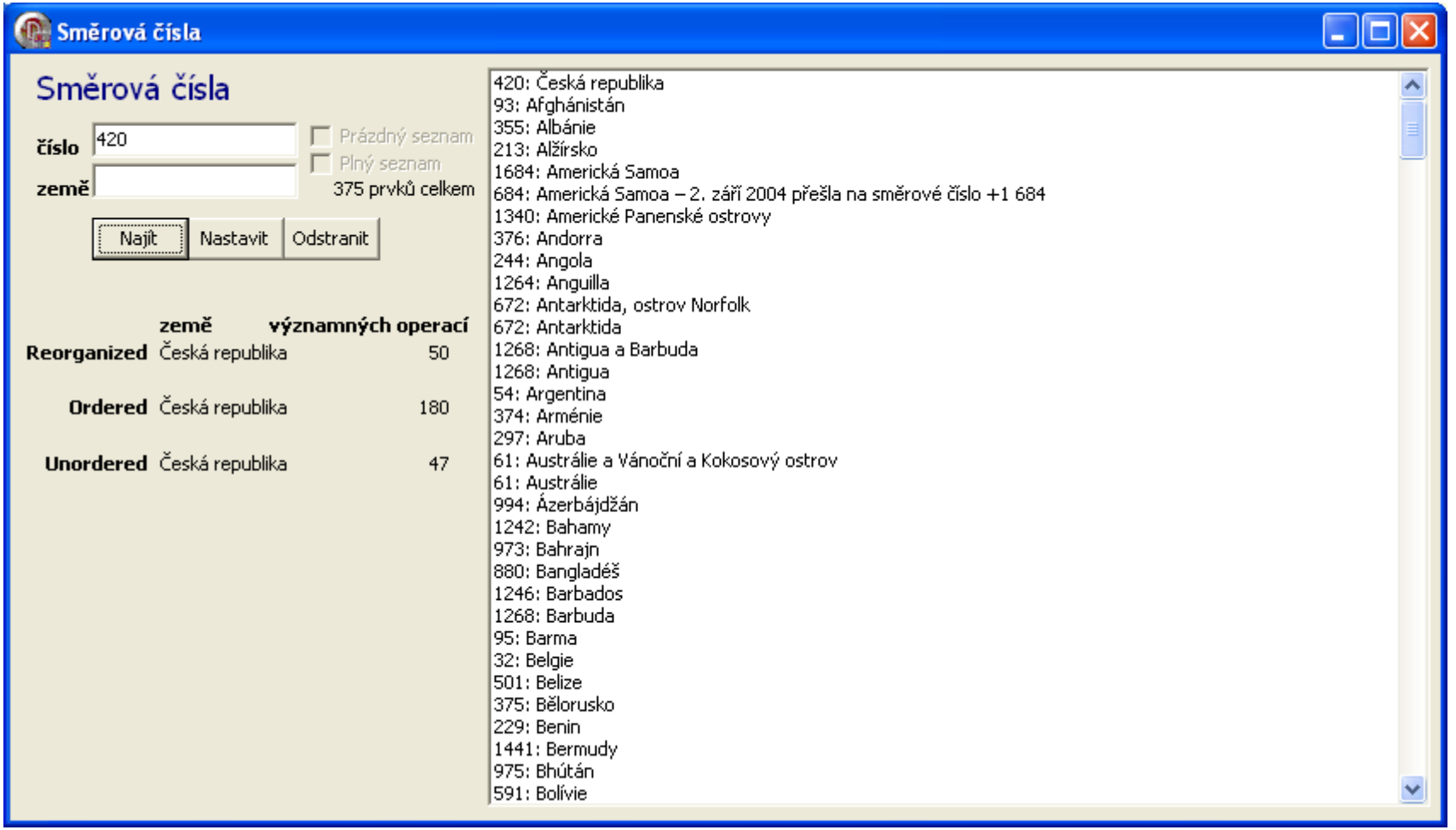
GetCount ([*front,* Stack.Create]) = 0
GetCount ([*front,* Push (*rear, x*)]) = GetCount (Remove ([*front,* Push (*rear, x*)])) + 1

MoveTo (*s,* 0) = MoveFirst (*s*)
MoveTo (MovePrev (*s*), *n* + 1) = MoveTo (*s, n*)
MoveTo (MoveNext (*s*), *n* – 1) = MoveTo (*s, n*)

Obecná posloupnost

- Specifikace datového typu posloupnost
- **Příklad: směrová čísla**
- Implementace posloupnosti dvojicí zásobníků

Příklad: směrová čísla

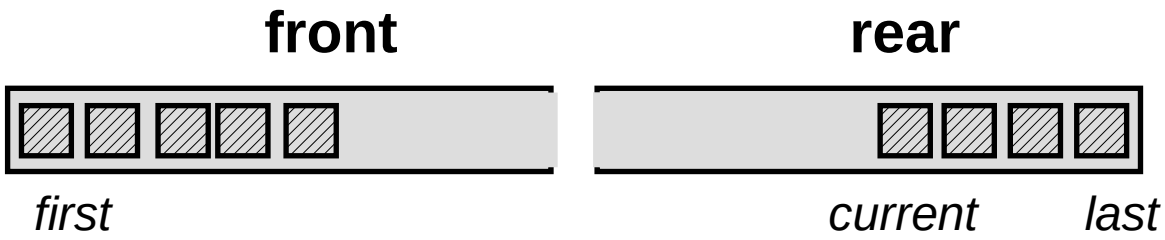


Formulář programu „*Směrová čísla*“

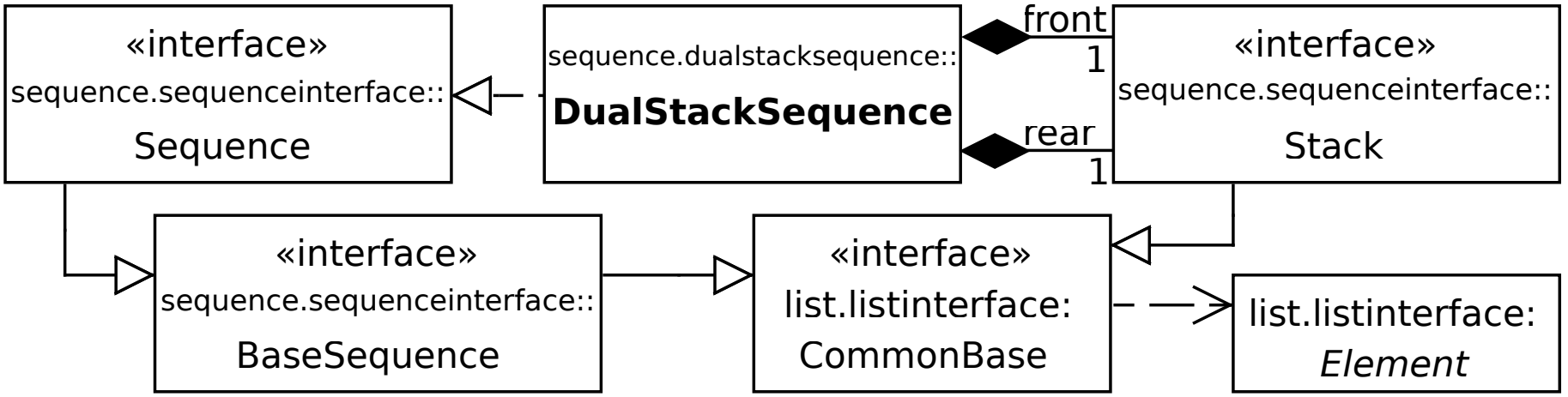
Obecná posloupnost

- Specifikace datového typu posloupnost
- Příklad: směrová čísla
- **Implementace posloupnosti dvojicí zásobníků**

Implementace posloupnosti dvojicí zásobníků



Schema posloupnosti implementované dvojicí zásobníků



Design posloupnosti implementované dvojicí zásobníků

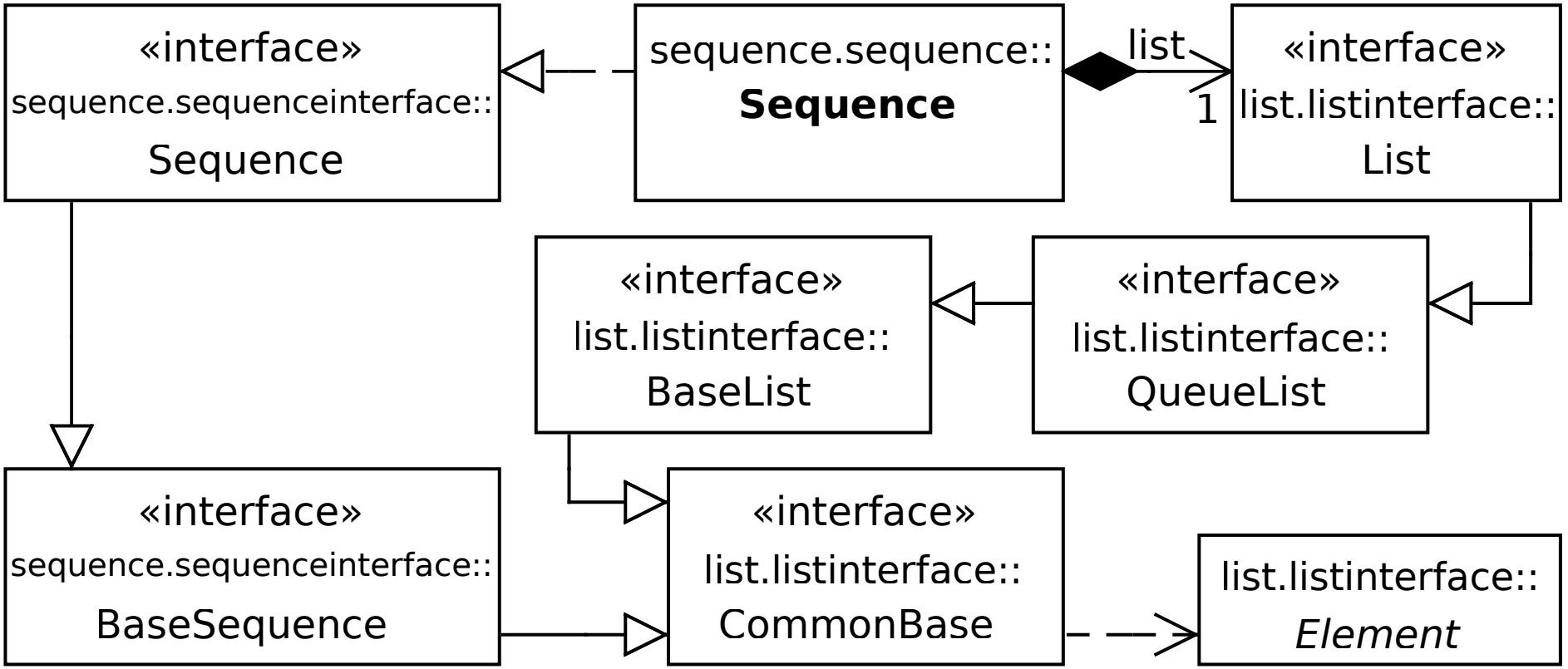
Posloupnosti a operace s nimi

- Posloupnost jako ADT
- Zásobník
- Fronta
- Obecná posloupnost
- **Implementace posloupnosti seznamem**
  - Implementace lineárního a kruhového seznamu polem
  - Prostý spojový seznam
  - Spojový seznam se zaostalým běžným členem
  - Spojový seznam s vlečeným členem
  - Obousměrně propojený seznam
- Shrnutí

Implementace posloupnosti seznamem

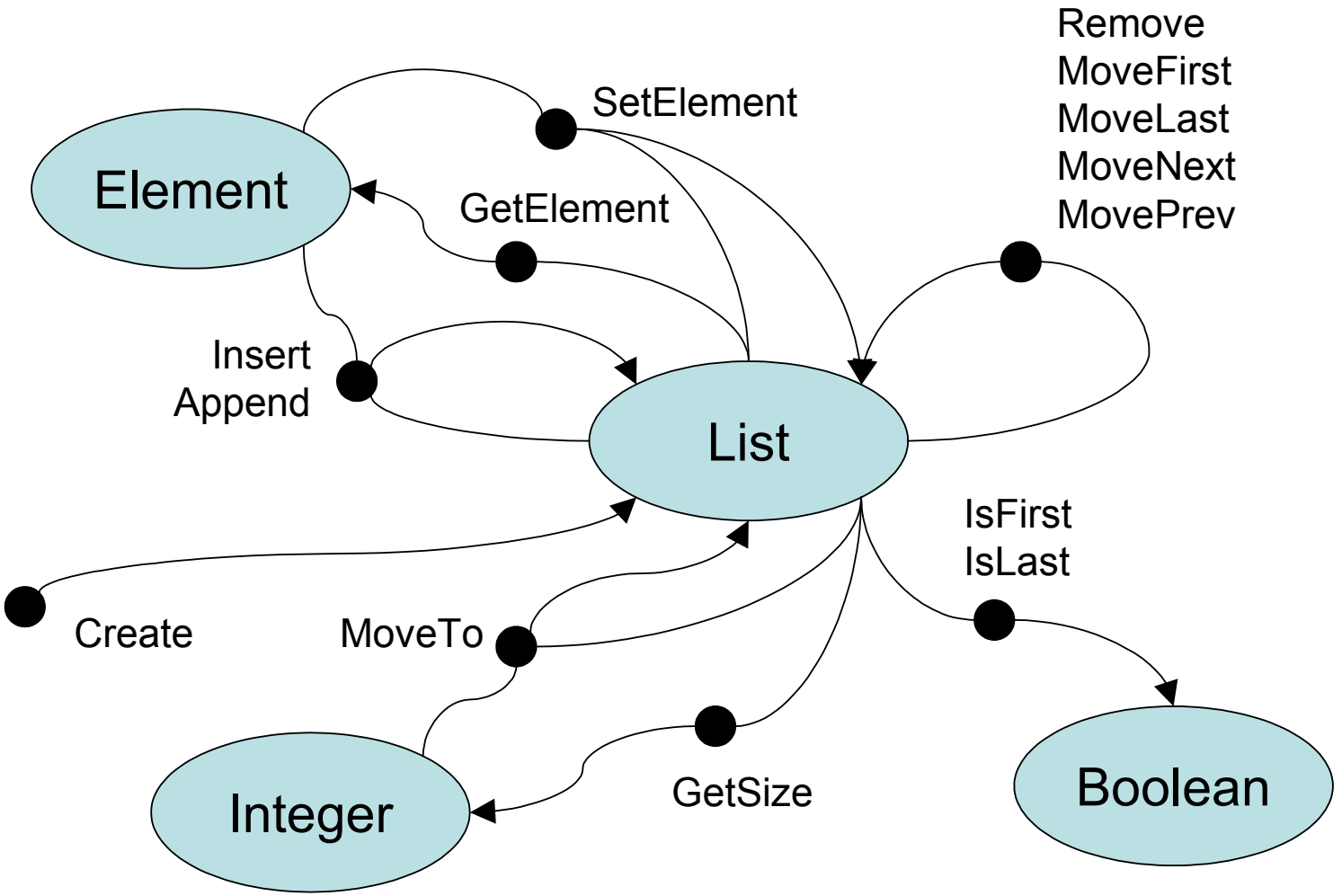
- Implementace lineárního a kruhového seznamu polem
- Prostý spojový seznam
- Spojový seznam se zaostalým běžným členem
- Spojový seznam s vlečeným členem
- Obousměrně propojený seznam

Implementace posloupnosti seznamem



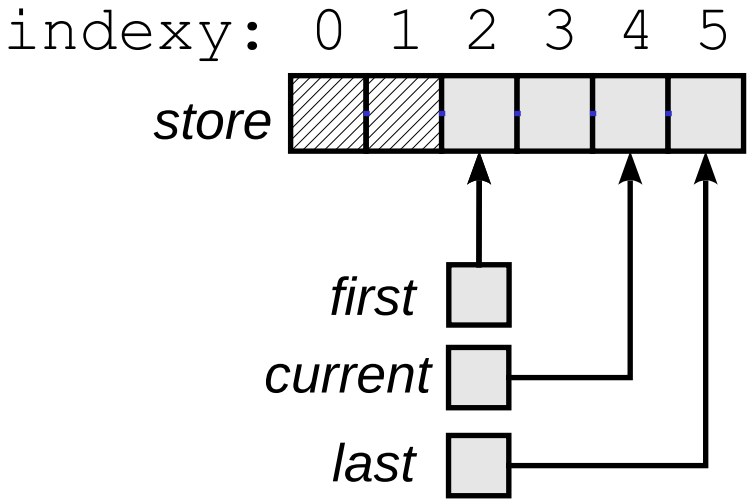
Design posloupnosti implementované seznamem

Implementace posloupnosti seznamem

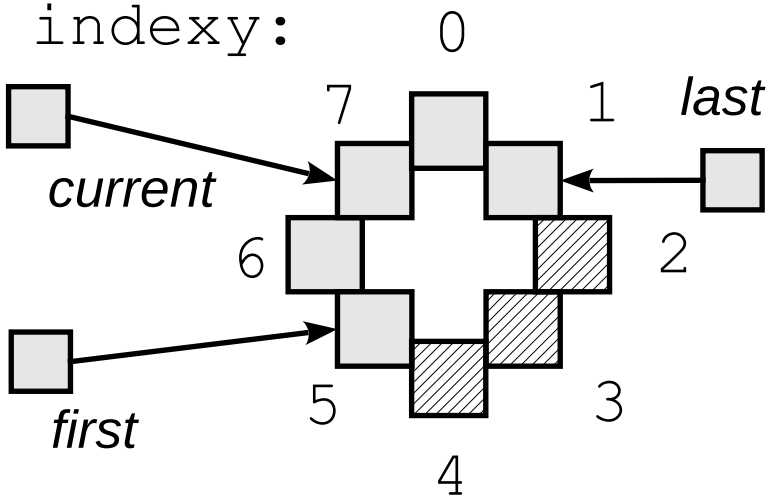


Signatura datového typu *seznam*

Implementace lineárního a kruhového seznamu polem

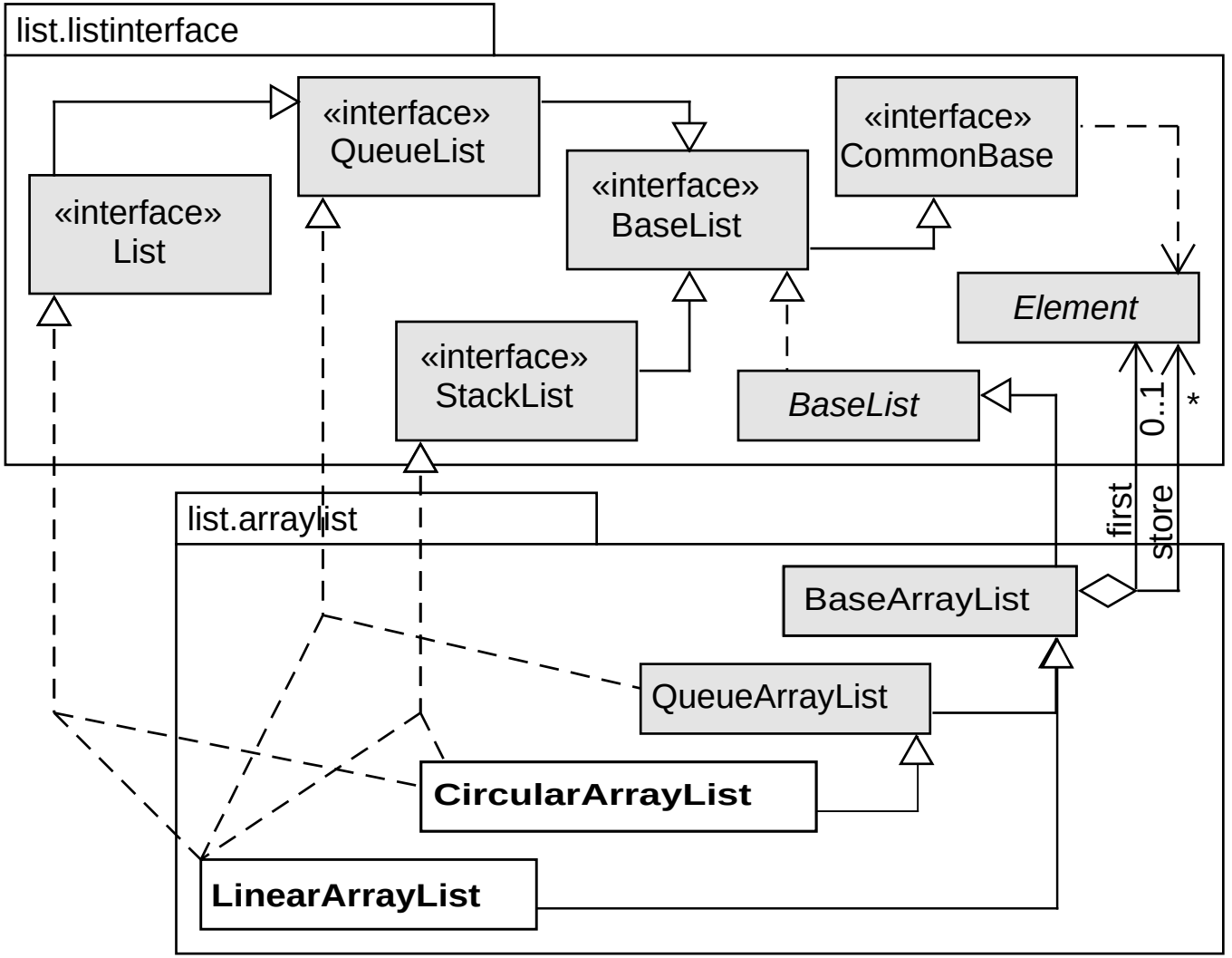


Znázornění polem implementovaného lineárního seznamu



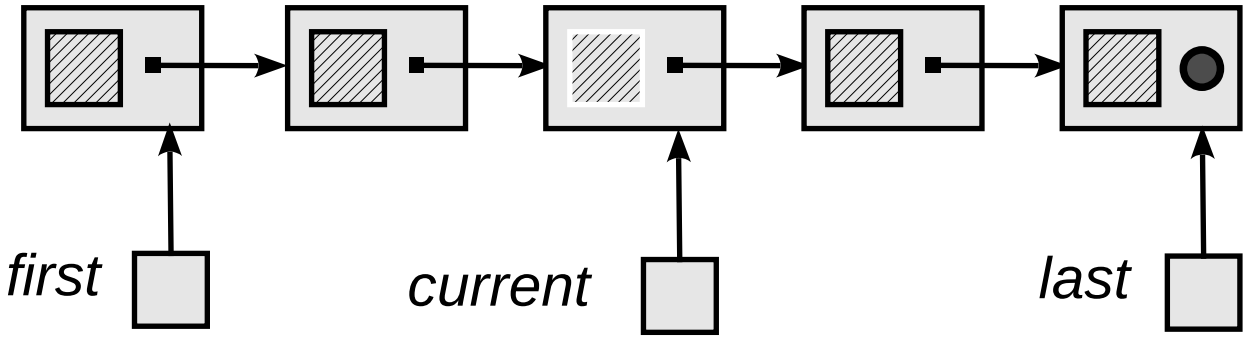
Znázornění pole jako kruhového seznamu

Implementace lineárního a kruhového seznamu polem



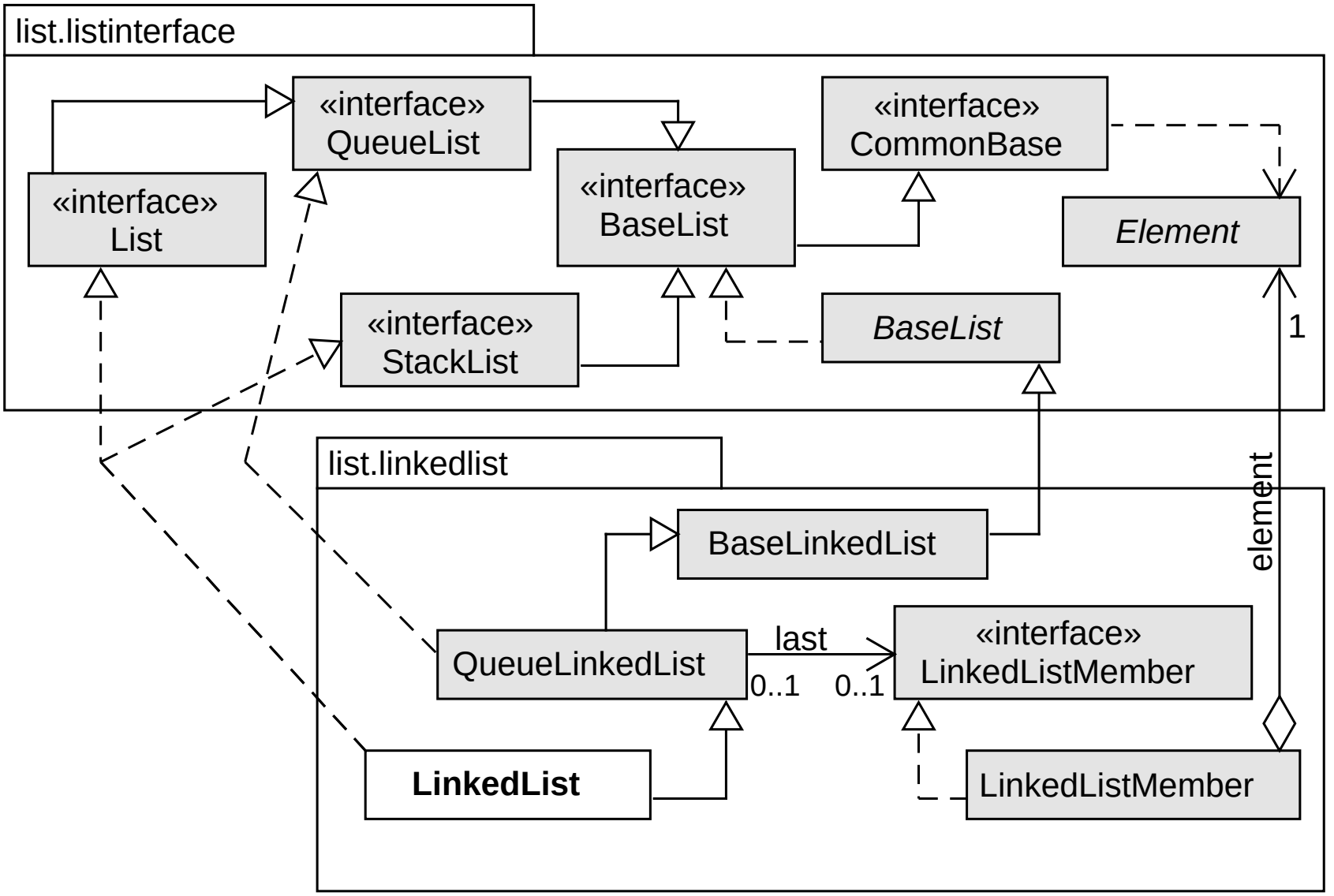
Návrh implementace seznamu polem

Prostý spojový seznam



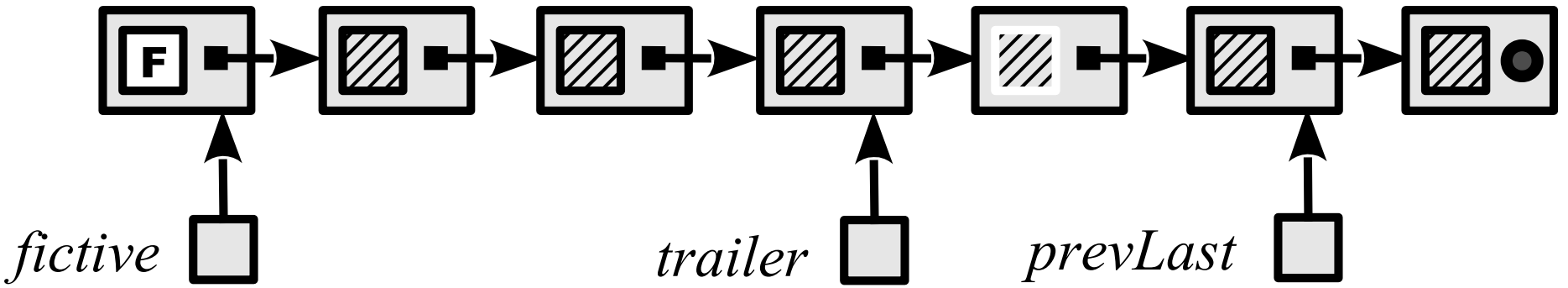
Znázornění prostého spojového seznamu

Prostý spojový seznam



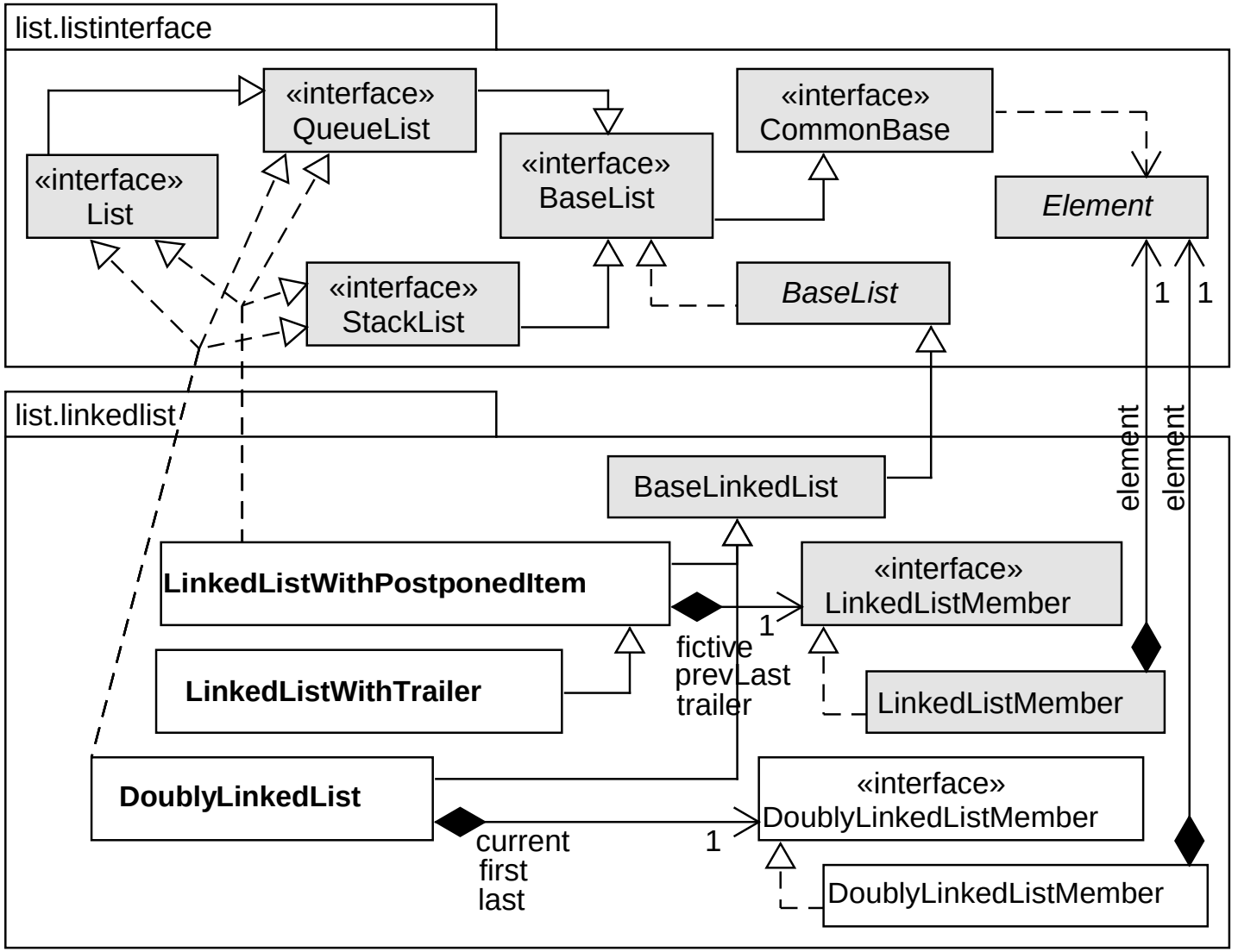
Implementace seznamu spojovým seznamem prostým

Spojový seznam se zaostalým běžným členem



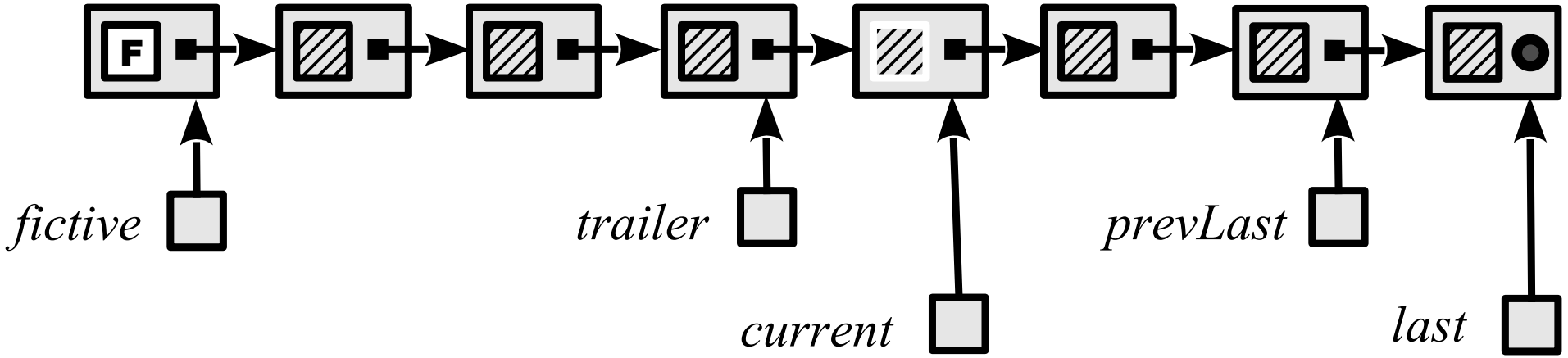
Znázornění spojového seznamu se zaostalým běžným členem

Spojový seznam se zaostalým běžným členem



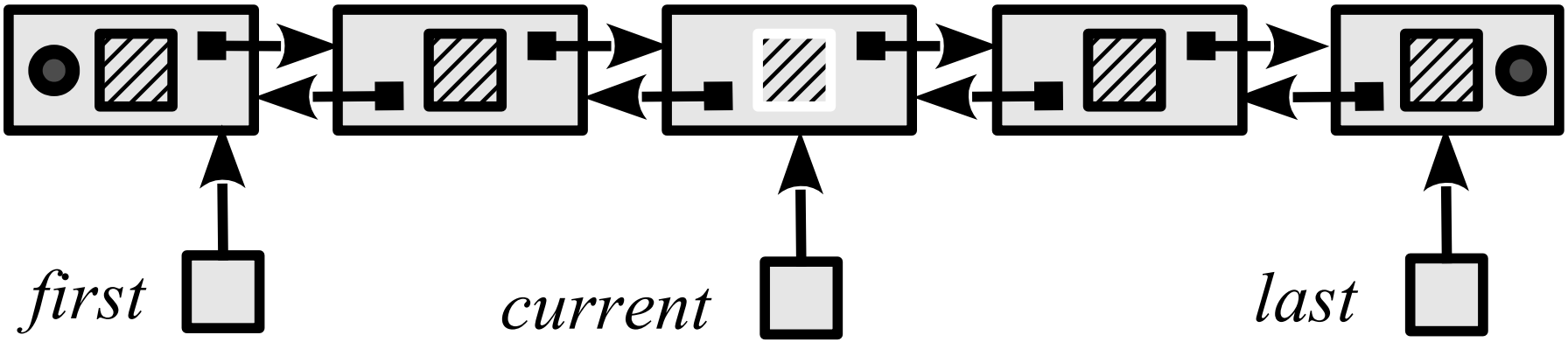
Další implementace seznamu spojovým seznamem

Spojový seznam s vlečeným členem



Znázornění spojového seznamu s vlečeným členem

Obousměrně propojený seznam



Znázornění obousměrně propojeného seznamu

Posloupnosti a operace s nimi

- Posloupnost jako ADT
- Zásobník
- Fronta
- Obecná posloupnost
- Implementace posloupnosti seznamem
- **Shrnutí**

Shrnutí

- **Posloupnost** je jednoduchá a přitom všestranně užitečná datová struktura. Např.:
  - Databázové tabulky nejsou nic jiného než posloupnosti záznamů.
  - Seznam (tj. posloupnost) je univerzální datovou strukturou v jazycích Lisp a Prolog.
  - Nekonečná páska Turingova stroje je také posloupnost.
- Zvláštními (zjednodušenými) případy posloupnosti jsou *zásobník* a *fronta.*
  - **Zásobník** se hodí k implementaci mechanismu rekurze, např. v inferenčních strojích, jako je např. virtuální stroj Prologu. Dvojice zásobníků umožňuje implementovat obecnou posloupnost (a proto také matematický stroj se dvěma zásobníky má sílu stroje Turingova).
  - **Fronta** se hodí např. ke komunikaci a synchronizaci mezi paralelními procesy.
- Posloupnosti lze implementovat mnoha způsoby, např. **polem** nebo **spojovým seznamem.**
  - Výhodou *pole* proti *spojovému seznamu* je to, že k libovolnému prvku se přistupuje stejně rychle – jedním elementárním krokem výpočtu.
  - Výhodou *spojového seznamu* je, že se může dynamicky prodlužovat a zkracovat přesně podle okamžité potřeby.

Konec

Tato prezentace patří k učebnici *Algoritmy a datové struktury objektově* od Ivana Ryanta. Obsahuje texty a obrázky z této učebnice.

RYANT, Ivan. *Algoritmy a datové struktury objektově.* Praha: Ivan Ryant, 2017. ISBN 978-80-270-1660-0