Biðraðir

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-02-08

Flakkarar

Íhugum skiladæmi 4.3 (RPN reiknivél í Java):

- Hvernig útfærum við p (print) skipunina?
- Þurfum að geta lesið öll stökin á hlaðanum
- Viljum helst ekki þurfa að breyta/skemma hlaðann

- Möguleikar:
  1. Notum pop() – Ljótt því það breytir hlaðanum
  2. Bætum print() í Stack.java – Ljótt, brýtur aðskilnað
  3. Notum flakkara (iterator) – Falleg, almenn leið

Flakkari (e. iterator)

Flakkari "flakkar" í gegnum gagnamót og leyfir notandanum að skoða gildin sem það inniheldur

Kostir flakkara

Af hverju viljum við nota flakkara?

- Flakkarar virða aðskilnað milli forritara og notanda
- Kóði sem þarf að skoða öll gildi er bara háður flakkaranum
- Ekki háður sérhæfðum viðbótum (print() aðferð í Stack.java)
- Ekki háður innviðum hlaðans eða tiltekinni útfærslu á hlaða

$\Longrightarrow$ Einföld, almenn og endurnýtanleg lausn

java.util.Iterator

Flakkarar í Java:

- Allir flakkarar í Java útfæra viðmótið java.util.Iterator
- Innbyggðir flakkarar fyrir öll helstu gagnamót í Java
- Getum skrifað okkar eigin flakkara fyrir okkar eigin gagnamót

Flakkari hefur þrjár aðferðir:

```
public interface Iterator<T>
{
    boolean hasNext();
    T next();
    void remove();      // optional
}
```

Dæmi um notkun á flakkara

```
public static void main(String[] args)
{
    Stack<String> s = new Stack<String>();

    s.push("A");
    s.push("B");
    s.push("C");

    Iterator<String> it = s.iterator();

    while (it.hasNext())
    {
        String p = it.next();
        System.out.println(p);
     }
  }
}
```

Flakkað með for-lykkju

```
public static void main(String[] args)
{
    Stack<String> s = new Stack<String>();

    s.push("A");
    s.push("B");
    s.push("C");

    for (String p : s)
    {
        System.out.println(p);
    }
}
```

print í RPN

```
Stack<Integer> s = new Stack<Integer>();

// ...
} else if (w.equals("p"))
{
    System.out.println("Stack (from top to bottom)");

    for (Integer i : s)
        System.out.printf("  [%d]\n", i);
}
```

Iterable

Eintök klasa sem útfærir Iterable styðja flakk með flakkara.

```
public interface Iterable<T>
{
    Iterator<T> iterator();
}

class X implements Iterable<T>
{
    Iterator<T> iterator()
    {
        // ...
    }
}
```

Iterator demo

Skoðum útfærslu á flakkara fyrir Stack.java

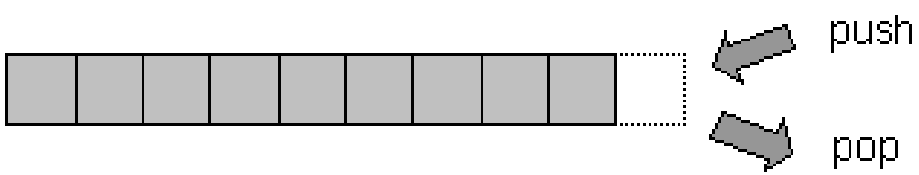
Biðröð

Biðröð er gagnamót sem uppfyllir eftirfarandi lýsingu:

- Biðröð geymir safn af stökum í s.k. FIFO röð
- FIFO: fyrst-inn-fyrst-út – eins og biðröð í verslun
- Biðröð hefur að lágmarki aðgerðirnar enqueue og dequeue:
  1. enqueue(x): Setur x aftast í biðröðina
  2. dequeue(): Tekur fremsta stakið úr biðröðinni og skilar því

Biðröð:

Hlaði:



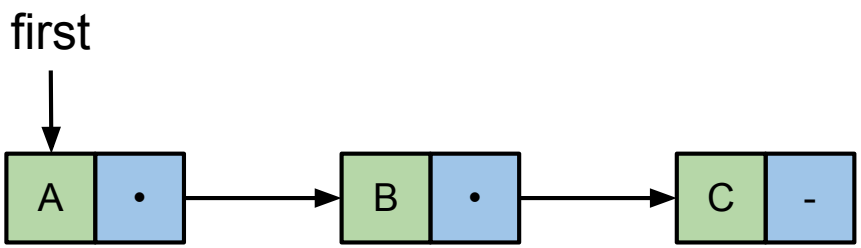
Útfærsla á biðröð

Upprifjun:

- Við notuðum eintengdan lista til að útfæra hlaða
- Héldum utan um vísun á fyrsta hlekk í keðjunni
- Keðjan inniheldur öll stök á hlaðanum frá efsta til neðsta

Hvað með að útfæra biðröð á sama hátt?

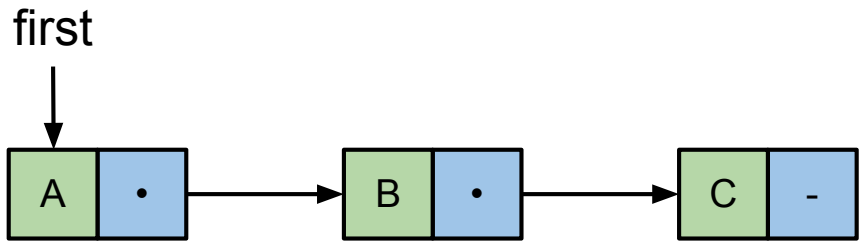
- Keðjan inniheldur öll stök í biðröðinni frá fremsta til aftasta
- Gott eða slæmt?



Útfærsla á biðröð

Hvað með að útfæra biðröð á sama hátt?

- Keðjan inniheldur öll stök í biðröðinni frá fremsta til aftasta
- Gott eða slæmt? Slæmt!



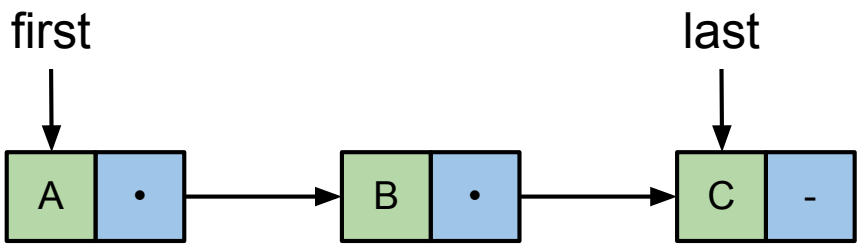
- enqueue þarf að ganga alla keðjuna til að bæta við gildi
- m.ö.o. enqueue er O(N) þar sem N er fjöldi gilda í biðröðinni

- dequeue er hinsvegar í lagi, það kemst beint í fremsta gildið
- dequeue er O(1)

Næsta tilraun

Næsta tilraun:

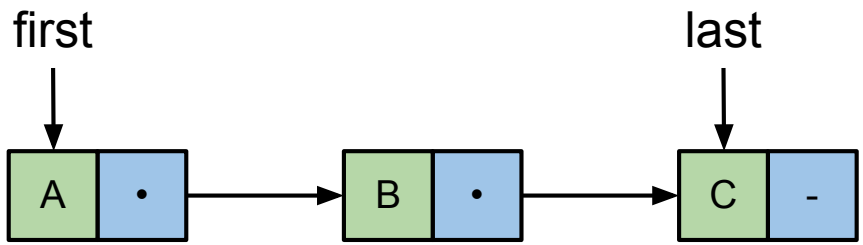
- Notum eintengdan lista
- Geymum tilvísun á fremsta og aftasta hlekk í keðjunni
- Keðjan inniheldur öll stök í biðröðinni frá fremsta til aftasta
- Gott eða slæmt?



Betri útfærsla

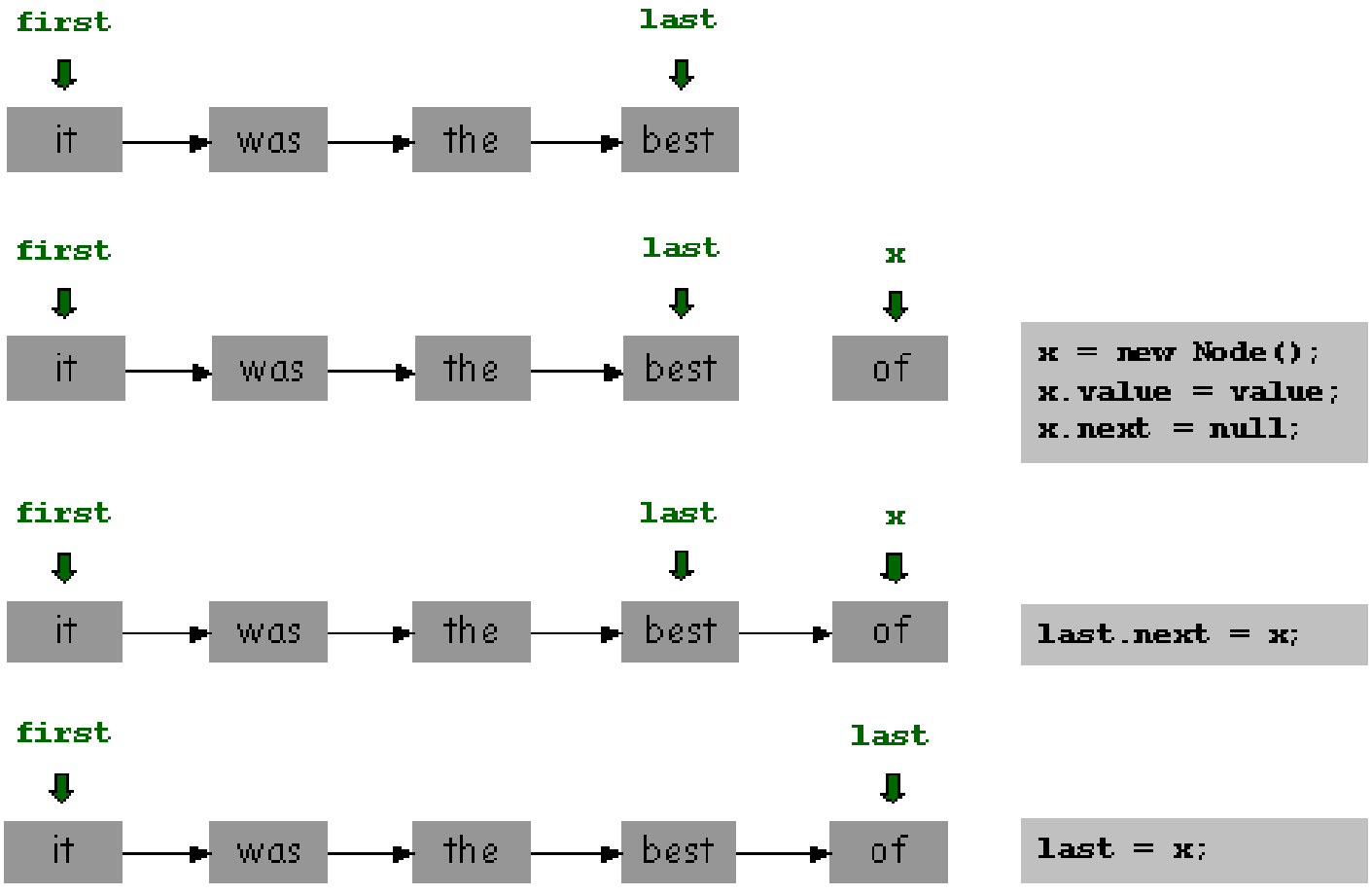
Næsta tilraun:

- Notum eintengdan lista
- Geymum tilvísun á fremsta og aftasta hlekk í keðjunni
- Keðjan inniheldur öll stök í biðröðinni frá fremsta til aftasta
- Gott eða slæmt? Gott!



- enqueue er O(1) – tekur fastan tíma
- dequeue er O(1) – tekur fastan tíma

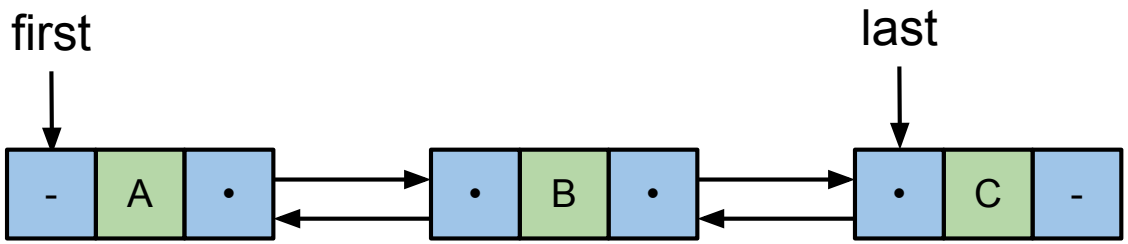
Innsetning aftast



Aðrar útfærslur

Útfærsla með tvítengdum lista:

- Gildi geymd í tvítengdum lista í röð frá fremsta til aftasta



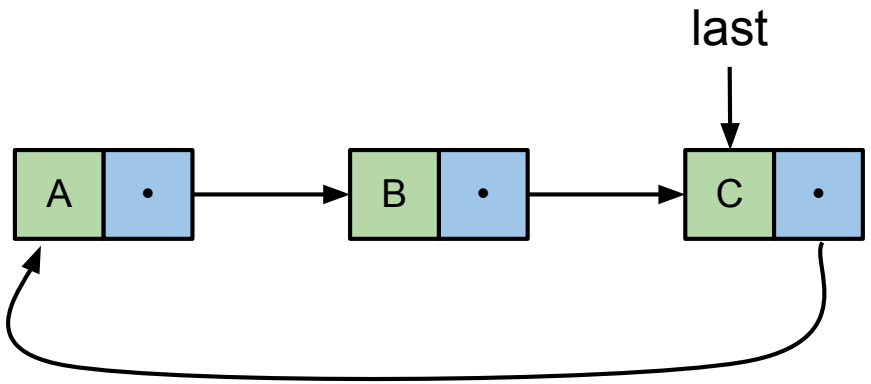
- Auðvelt að fjarlægja gildi
- Styður aukalega O(1) aðgerð sem fjarlægir aftasta gildi

- Biðröð sem styður aðgerðir á báða enda er kölluð deque
- deque styður innsetningu/úttak bæði fremst og aftast
- deque er eins og biðröð og hlaði saman í einu

Aðrar útfærslur

Útfærsla með hringtengdum lista:

- Gildi geymd í hringtengdum lista í röð frá fremsta til aftasta

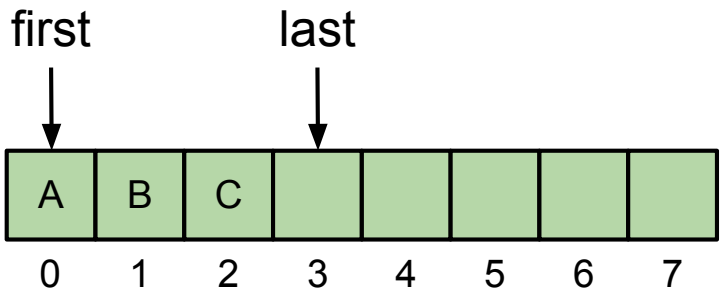


- Aftasta hlekkurinn er last
- Fremsti hlekkurinn er last.next
- enqueue er O(1)
- dequeue er O(1)

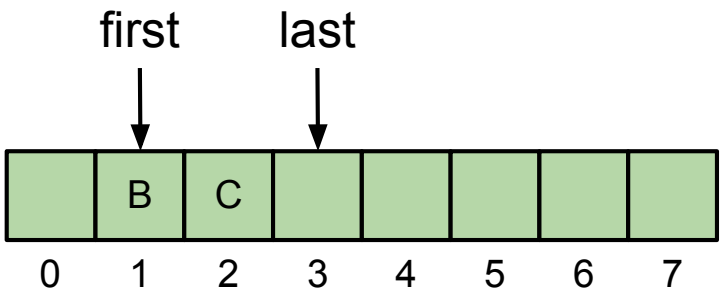
Aðrar útfærslur

Útfærsla með hringlaga fylki:

- Gildi geymd í fylki sem gengur í hring
- q.enqueue("A"); q.enqueue("B"); q.enqueue("C");



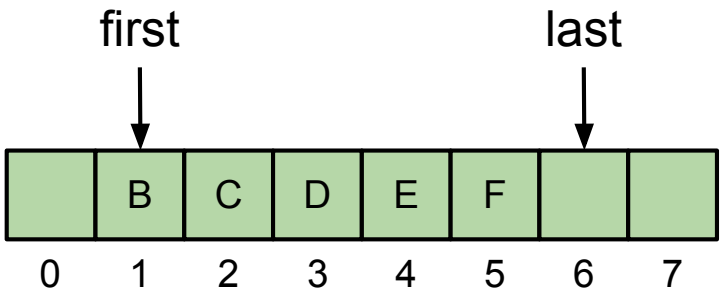
- q.dequeue(); – Skilar "A"



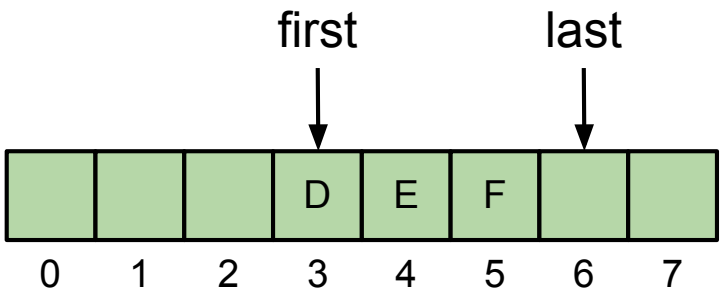
Aðrar útfærslur

Höldum áfram:

- q.enqueue("D"); q.enqueue("E"); q.enqueue("F");



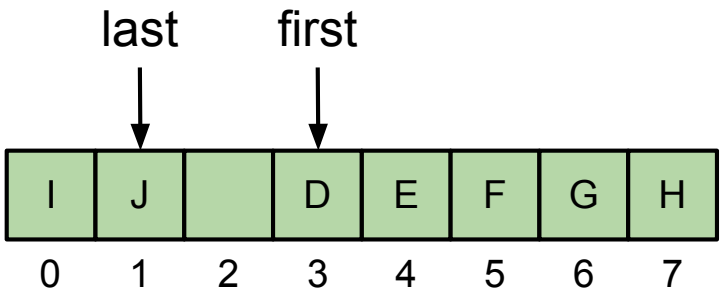
- q.dequeue(); – Skilar "B"
- q.dequeue(); – Skilar "C"



Aðrar útfærslur

Wrap around:

- q.enqueue("G"); q.enqueue("H");
- q.enqueue("I"); q.enqueue("J");



- Möguleiki á underflow og overflow
- enqueue er O(1)
- dequeue er O(1)