

Programmering för språkteknologer II

OH-serie 2: Liststrukturer med labbuppgiftsinformation



UPPSALA
UNIVERSITET

Mats Dahllöf

Institutionen för lingvistik och filologi
September 2008

1

Programmering för språkteknologer II — HT 2008 (Mats Dahllöf)

Listor

- En ordnad, godtyckligt lång, sekvens av element (t.ex. objekt).
- Olika beteenden, t.ex. Last In First Out (LIFO; den typen kallas *stack*) eller First-In-First-Out (FIFO; *kö*).
- Det finns olika sätt att implementera en lista.
- Standardklasser i Java, t.ex. `LinkedList<E>` `ArrayList<E>`, som är s.k. **generiska klasser med parametriserade typer**.

2

Programmering för språkteknologer II — HT 2008 (Mats Dahllöf)

Stack, en abstrakt datatyp

En **abstrakt datatyp** definierar en struktur för information och de operationer som är tillåtna på denna struktur.

En konkret **implementation** kan fånga en abstrakt datatyp.

- **Abstrakta datatypen ”stack”**
Liststruktur – innehåller objekt som är linjärt ordnade. Last In First Out (LIFO) — Alltså, vi ser bara det ”färskaste” objektet. (Borde ha översatts som ”stapel”. ”Stack” är dock helt etablerat på svenska, fast dålig svengelska.)

3

Programmering för språkteknologer II — HT 2008 (Mats Dahllöf)

Stack, en abstrakt datatyp

- Två grundläggande operationer:
push: lägg nytt objekt (överst) på stacken.
pop: ta bort objektet (överst) på stacken.
- Andra operationer:
djup/length/size: fråga om antalet objekt på stacken.
peek: kika på objektet (överst) på stacken (och låta det ligga kvar).

4

Programmering för språkteknologer II — HT 2008 (Mats Dahllöf)

Stack, en abstrakt datatyp

- Två grundläggande operationer:
push: lägg nytt objekt (överst) på stacken.
pop: ta bort objektet (överst) på stacken och returnera det.
- Andra operationer:
djup/length/size: fråga om antalet objekt på stacken.
peek: kika på objektet (överst) på stacken (och låta det ligga kvar).
(Fråga: Skulle vi kunna utföra **djup** och **peek** med bara **push** och **pop**? Vad krävs i så fall?)

5

Programmering för språkteknologer II — HT 2008 (Mats Dahllöf)

Använda Javas `LinkedList<E>` som stack

- Javas standardklass `LinkedList<E>` ger en konkret implementation av en stack. (Men är mer rikhaltig än så!)
- Metodnamn i klassen `LinkedList<E>`

```
push          void addFirst(E o)
pop           E remove()
djup/length/size  int size()
peek         E peek()
```

6

Programmering för språkteknologer II — HT 2008 (Mats Dahllöf)

Uppgift: hemmagjord länkad lista (och stack)

- Består av noder. Speciell klass.
I min kod: `StackNode`. Lämpligen som **intern klass**. (Vad är det?) En `StackNode` har två variabler, själva elementet (`value`), samt en för nästa nod (`next`) och därmed resten av stacken.
- En instansvariabel (i `SimpleStack`) pekar på översta noden (`topNode`).
- ”Stopp” då `next`-värdet är `null`.

7

Programmering för språkteknologer II — HT 2008 (Mats Dahllöf)

`SimpleStack` (med intern klass `StackNode`)

```
class SimpleStack<E> { // Generisk klass
    private class StackNode { // Intern klass
        public E value; // Inst. var.
        public StackNode next; // Inst. var.
        StackNode(E v, StackNode n) { // Konstrukt.
            value=v;
            next=n;
        }
    }
}
```

8

SimpleStack: push-operationen

```
public void push(E v) {
    StackNode newNode =
        new StackNode(v, topNode);
    topNode=newNode;
}
```

Skapa ny nod med givet värde och gamla toppnoden som nästanod. Nya noden blir ny toppnod.

9

SimpleStack: pop-operationen

```
public E pop() {
    E v = topNode.value;
    topNode = topNode.next;
    return v;
}
```

Spara tillfälligt toppnodens värde (v). Låt toppnodens next-värde bli ny toppnod. Gamla värdet hos topNode blir otillgängligt. (Och kan skräphanteras.) Returnera v.

10

I Java: String toString()

- Metoder med namnet toString, inga argument, returtyp String, används i vissa lägen implicit för att konvertera värden till String-form.
- Kan ingå i vilken klass som helst.
- Används vid utskrifter, t.ex. om en icke-String ges som argument i kontexten System.out.println(...).
- Används då en String sätts ihop med en icke-String med hjälp av operatör + (konkatenering).

11

SimpleStack: toString(), typ "[c, b, a]"

```
public String toString() {
    StackNode inFocus = topNode;
    String str = "[";
    while (inFocus != null) {
        str = str + inFocus.value;
        // N.B. inFocus.value.toString() anropas.
        inFocus = inFocus.next;
        if (inFocus != null) {
            str = str + ", ";
        }
    }
    return str + "]" ;}
}
```

12

Indexering i linjär struktur

- Indexering = göra något med/vid element på plats *i*. T.ex. läsa av, ta bort, sätta in. Kräver *i* "hopp" i en enkellänkad lista, som SimpleStack.
- Fält (array) är skräddarsydda för indexering. Vi vet precis var (i minnet) varje element ligger. Nackdel: Fält har en fast längd. (Listor har godtycklig längd.)

13

Sätt att implementera en lista

- Enkellänkad lista, som SimpleStack. Dubbellänkad lista (som LinkedList<E>). Indexering kräver ett antal "hopp", typiskt i proportion till längden.
- Baserat på underliggande fält (array). Ex. standardklassen ArrayList<E> Indexering kostar minimalt. Om storleken växer kan fältet få byggas om. Det kostar: Vi måste skapa nytt fält och kopiera värden.

14