050 5980451 4150

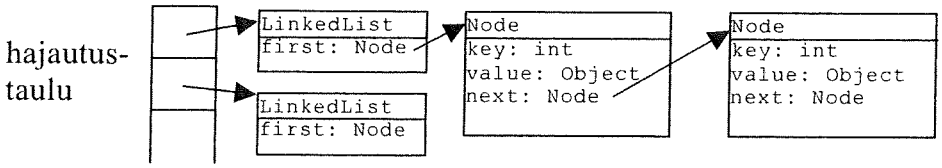
T-106.213 Ohjelmoinnin peruskurssi L1: tentti 14.01.2005 / Kary Främling

Kaikki kysymykset tässä tentissä liittyvät liitteissä 1 - 3 annettuun ohjelmakoodiin ja dokumentaatioon. Kuitenkin kysymykset ovat siinä määrin toisistaan riippumattomia, että vaikka et osaisi vastata yhteen tiettyyn kysymykseen, on ihan mahdollista osata vastata muihin.

Luokka `IntHashMap` liitteessä 1 on yksinkertainen toteutus niinsanotulle sanakirjarakenteelle (engl. *map*, *dictionary*), johon voi varastoida avain-arvopareja (engl. *key-value mappings*), ja josta voi hakea arvoja (engl. *value*) avaimen (engl. *key*) perusteella. Annettu luokka eroaa Javan standardiluokasta `java.util.HashMap` (jonka yksityiskohtia ei tarvitse tässä tarkemmin tuntea) ennen kaikkea siten, että `IntHashMap`-luokassa vain `int`-tyyppisiä kokonaislukuja käytetään avaimina.

Luokka `IntHashMap` on toteutettu ns. *hajautustaulun* (engl. *hashtable*) avulla, jonka toimintaperiaate on seuraava. Hajautustaululla on tietty vakiokoko `n` ja sen sisältö pidetään tallessa tämän kokoisessa taulukossa. Kun hajautustauluun halutaan lisätä tietty arvo tietylle avaimelle, lasketaan avaimesta (tässä: avaimena toimivasta kokonaisluvusta) ns. *hajautusfunktiolla* (engl. *hashing function*) arvon sijoittamisindeksi, joka on välillä 0 - (n-1). Luokassa `IntHashMap` käytetään hajautusfunktiona yksinkertaisesti jakojäännöstä `avain % n`.

Hajautusfunktio voi antaa saman sijoittamisindeksin usealle eri avaimelle. Esim. `IntHashMap`-luokan tapauksessa hajautustaulussa, jonka koko on 11, avaimet 5 ja 27 saavat saman sijoittamisindeksin. Luokka `IntHashMap` ratkaisee tämän ongelman siten, että kutakin indeksiä vastaa yksi linkitetty lista (engl. *linked list*), johon tallennetaan kaikki ne avain-arvoparit, joilla on kyseinen sijoittamisindeksi. Annetut luokat `LinkedList` ja `Node` (eli listasolmu) toimivat tähän tarkoitukseen sopivana listatoteutuksena. Seuraavassa kuvassa on esitetty miten hajaustustaulu koostuu taulukollisesta linkitettyjä listoja, joiden solmuissa puolestaan on tallessa avain-arvopareja.



Tutustu annettuun ohjelmakoodiin ja dokumentaatioon huolellisesti!

1. Laadi metodit `clear`, `get` ja `put` liitteen 1 luokkaan `LinkedList` siten, että luokan `IntHashMap` metodit toimivat liitteen 3 dokumentaation kertomalla tavalla. 20/100

2. Kirjoita rekursiivinen toteutus mainitulle `get`-metodille. Vinkki: sinun täytyy kenties kirjoittaa toinen apumetodi saadaksesi ratkaisun aikaiseksi. 10/100

3. Luokassa `IntHashMap` on paha virhe, joka ilmenee kun liitteen 2 main-metodi suoritetaan (riippumatta `LinkedList`-luokan toteutuksesta). Mikä on virhe ja miten sen voi korjata? 15/100

4. Mitä tapahtuu, jos yritetään käyttää negatiivisia avainarvoja? Miten tämän voisi korjata? 10/100

5. Mitä tulostuu kun liitteen 2 main-metodi suoritetaan, olettaen että ohjelma toimii liitteen 3 dokumentaation kuvaamalla tavalla? 15/100

6. Luokka `IntHashMap` olisi mahdollista toteuttaa Javan perusluokilla `ArrayList`, `HashSet` tms. annettujen `LinkedList`- ja `Node`-luokkien sijaan. Mitä muutoksia tällöin tulisi tehdä luokkaan `IntHashMap`? Mitä hyötyä tai haittaa tästä toisenlaisesta toteuttamisesta voisi olla? Ohjelmakoodia ei tarvitse esittää, vaan ainoastaan kuvailla uusi toteutus lyhyesti. 15/100

7. Millaisia muutoksia tarvittaisiin luokkaan `IntHashMap`, jos haluttaisiin, että sen tulisi toimia millaisilla tahansa (Object-tyyppisillä) avaimilla nykyisten `int`-avainten sijaan? Ohjelmakoodia ei tarvitse esittää, vaan ainoastaan kuvailla uusi toteutus lyhyesti. 15/100

Liite 1

```
public class IntHashMap {
    public static final int SIZE = 11;

    private LinkedList[] heads;

    public IntHashMap() {
        this.heads = new LinkedList[IntHashMap.SIZE];
    }

    public void clear() {
        for ( int i = 0 ; i < heads.length ; i++ )
            this.heads[i].clear();
    }

    public boolean containsKey(int key) {
        if ( this.get(key) != null )
            return true;
        return false;
    }

    public Object get(int key) {
        return this.heads[this.getHash(key)].get(key);
    }

    public Object put(int key, Object value) {
        return this.heads[this.getHash(key)].put(key, value);
    }

    public String toString() {
        String s = "";
        for ( int i = 0 ; i < this.heads.length ; i++ ) {
            if ( this.heads[i] != null )
                s += this.heads[i].toString();
            else
                s += "\n";
        }
        return s;
    }

    private int getHash(int key) {
        return key % IntHashMap.SIZE;
    }
}

public class LinkedList {
    private Node first = null;

    public void clear() {
        // puuttuu vielä...
    }

    public Object get(int key) {
        // puuttuu vielä...
    }

    public Object put(int key, Object value) {
        // puuttuu vielä...
    }



    // jatkuu seuraavalla sivulla...
```

```
    public String toString() {
        String s = "";
        for ( Node n = this.first ; n != null ; n = n.getNext() )
            s += "(" + n.getKey() + "," + n.getValue() + ") ";
        return s + "\n";
    }
}

public class Node {
    private int key;
    private Object value;
    private Node next = null;

    public Node(int key, Object value, Node next) {
        this.key = key;
        this.value = value;
        this.next = next;
    }

    public int getKey() {
      return this.key;
    }
    public void setKey(int key) {
      this.key = key;
    }
    public Object getValue() {
      return this.value;
    }
    public void setValue(Object value) {
      this.value = value;
    }
    public Node getNext() {
      return this.next;
    }
    public void setNext(Node next) {
      this.next = next;
    }
}
```

Liite 2. Testiluokka.

```
public class IntHashMapTest {
    public static void main(String[] args) {
      IntHashMap ihm = new IntHashMap();
      ihm.put(77,"Olle");
      ihm.put(2,"Nina");
      ihm.put(23,"Kalle");
      ihm.put(56,"Vivan");
      ihm.put(1,"Pelle");
      ihm.put(10,"Katarina");
      ihm.put(56,"Rasmus");
      ihm.put(98,"Maria");
      ihm.put(56,"Ivan");
      ihm.put(74,"Olivia");
      System.out.println(ihm);
    }
}
```

Liite 3 (muokattu rajapintaluokan `java.util.Map` dokumentaatiosta)

`class IntHashMap`

`public boolean` **`containsKey`**`(int key)`

Returns `true` if this map contains a mapping for the specified key.

`public Object` **`get`**`(int key)`

Returns the value to which this map maps the specified key. Returns `null` if the map contains no mapping for this key.

`public int` **`put`**`(int key,`
`Object value)`

Associates the specified value with the specified key in this map (optional operation). If the map previously contained a mapping for this key, the old value is replaced by the specified value.

Returns:

previous value associated with specified key, or `null` if there was no mapping for key.

`public void` **`clear`**`()`

Removes all mappings from this map (optional operation).