Inlämningsuppgift 2 - Datastrukturer

Avsikt

Avsikten med inlämningssuppgiften är att du ska fördjupa din förståelse för hur man bygger en datastruktur för datalagring.

Resurser

I arkiv-filen **p2.zip** hittar du följande filer:

- TestProgP2.java
- capital.txt

Redovisning

Redovisning ska ske genom att

- Du presenterar dina lösningar för läraren. Då lösningen utgörs av kod ska du ha färdiga testprogram som visar att lösningen är korrekt.
- De delar av din lösning som utgör kod ska ska vara kommenterad.

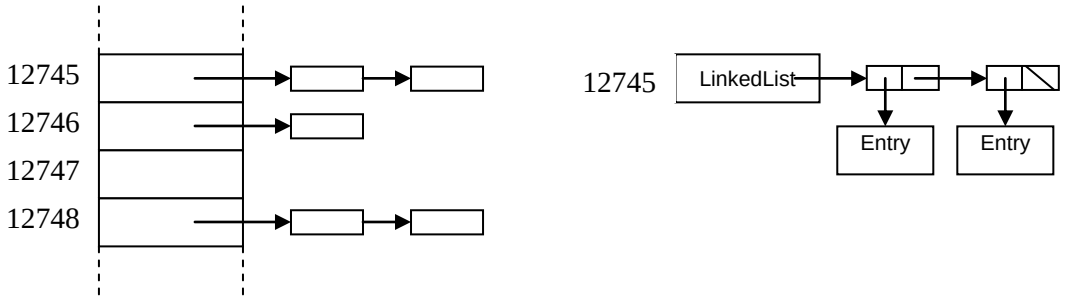
Dag för redovisning är måndagen den 18/4 och tisdagen den 19/4.

Uppgift 1

Du ska skriva klassen **Hashtabell** vilken ska medge snabb sökning bland ett stort antal lagrade objekt. När du löser uppgiften kan du utgå från klasserna **HashtableOH** och **Entry** från **f9**. Nedanstående punkter ska klassen **Hashtabell** uppfylla. Om du utgår från **HashtableOH** så är punkterna 1-4 uppfyllda och metoden **put** är färdig.

1. Vid konstruktion ska användaren ange storleken på tabellen.
   ```
   Hashtabell personer = new Hashtabell(100000);
   ```
2. Vid insättning av objekt i tabellen ska två objekt användas som argument, **nyckel**-objekt respektive **värde**-objekt.
   ```
   personer.put("630927-1111", "Samuel Ek");
   ```
3. Du kan förutsätta att nyckel-objektet innehåller korrekt implementering av metoderna **hashCode** respektive **equals**.
4. Tabellen ska aldrig bli full. Varje position i tabellen ska vara ett LinkedList-objekt (skapa ett LinkedList-objekt i varje element i arrayen i konstruktorn).

   Instansvariabler i klassen:
   ```
   private LinkedList[] table;
   private int count = 0;
   ```
   Tänkbar kod i konstruktorn:
   ```
   table = new LinkedList[ size ]; // size är parameter i konstruktorn, 100000 ovan)
   for( int i = 0; i < table.length; i++ )
       table[ i ] = new LinkedList();
   ```
   Nyckelns hashCode-metod ger ett heltalsvärde. Detta värde ska anpassas till tabellens storlek och ange den position i vilken (nyckel, värde)-paret ska lagras. Sedan lagras (nyckel, värde)-paret i LinkedList-objektet.



   För att lagra (nyckel, värde)-paret måste du använda någon typ av objekt. Om du vill kan du använda den givna klassen **Entry** (se Exempel 1 – 3 nedan). Det innebär att du ska lagra Entry-objekt i de länkade listorna (som i figuren ovan).
5. Följande metoder ska implementeras i klassen:

   public void clear()
   Hashtabellen töms på alla (nyckel-värde)-par. Alla *LinkedList*-objekt i tabellen måste alltså tömmas.

   public boolean containsKey(Object key)
   Metoden söker efter nyckeln i hashtabellen och returnerar *true* om nyckeln finns i hashtabellen och annars *false*.

   public Object get(Object key)
   Metoden söker efter nyckeln i hashtabellen och returnerar *value* om nyckeln finns i hashtabellen och annars *null*.

   public void put(Object key, Object value) // färdig i HashtabellOH att studera
   Sätt in (nyckel-värde)-paret i korrekt *LinkedList*-objekt. Men om nyckeln redan finns i listan ska ingenting göras.

   public void remove(Object key)
   Ta bort det (nyckel-värde)-par som har angiven nyckel.

public int size()
Returnerar antalet (nyckel-värde)-par som lagras i hashtabellen.

I testprogrammet **TestProgP2** ska du kunna ersätta klassen **HashMap** med din egen **Hashtabell** och körresultatet ska bli samma.

Hjälp vid lösning av Uppgift 1:

Klasserna *HashtableOH* och *Entry* (föreläsning 9) vilka innehåller grunderna för en lösning.

Metoden *put* i klassen *HashtableOH* visar hur man arbetar med strukturen, t.ex
* Hur man får en hashkod
* Hur man skapar ett *Entry*-objekt
* Hur man söker efter *Entry*-objekt i en *LinkedList* med metoden *indexOf*. Studera även *exempel 1-3* nedan.

Algoritmer för metoderna *get*, *put* och *remove* i föreläsningsunderlaget.

Exempel 1 – Klassen Entry lagrar ett (nyckel, värde)-par

```
package p2;
class Entry {
    Object key;
    Object value;
    public Entry( Object key, Object value ) {
        this.key = key;
        this.value = value;
    }
    // jämför två nycklar, returnerar true om lika
    public boolean equals( Object obj ) {
        Entry keyValue = ( Entry )obj;
        return key.equals( keyValue.key );
    }
}
```

Exempel 2 – Klassen Entry, equals jämför nycklarna

Kontrollera om två objekt av typen *Entry* är lika:

```
Entry e1 = new Entry("eka", "boat");
Entry e2 = new Entry("glad", "happy");
Entry e3 = new Entry("eka", "echo");
Entry e4 = new Entry("boll", "ball");
boolean lika1 = e1.equals( e2 ); // lika1 tilldelas värde false eftersom nycklarna ej är lika
boolean lika2 = e1.equals( e3 ); // lika2 tilldelas värde true eftersom nycklarna är lika
```

Exempel 3 – Klassen Entry, söka efter Entry-objekt i lista

```
LinkedList list = new LinkedList();
Entry e1 = new Entry("eka", "boat");
Entry e2 = new Entry("glad", "happy");
Entry e4 = new Entry("boll", "ball");
list.add(e1);
list.add(e2);
list.add(e4);


String key = "boll";
Entry keyValue = new Entry( key, null);   // Söka efter key i list
int res = list.indexOf( keyValue );       // res tilldelas värde 2 eftersom ett Entry-objekt
                                          //med  nyckeln ”boll” finns i position 2.
```

Uppgift 2

I denna uppgift ska du använda filen **capital.txt**. Filen innehåller ett antal rader med två namn i varje rad, nämligen namnet på ett land och namnet på landets huvudstad. Ett par rader i filen kan se ut så här:

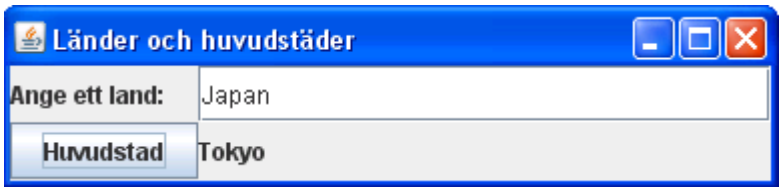
```
Sverige,Stockholm
Norge,Oslo
```

Som du ser separeras namnen av ett komma-tecken, precis som orden separerades i Laboration9.java. Det innebär att du kan använda en liknande metod som den i Laboration9 (readFromFile) för att läsa namnen i capital.txt. Samtliga inlästa (land-huvudstad)-par ska placeras i ett **Hashtabell**-objekt. Landet ska vara nyckel och huvudstaden värde.

Din uppgift är att skriva ett program vilket låter en användare mata in namnet på ett land och sedan skriver ut vad landets huvudstad heter. Om landet inte finns lagrad Hashtabell-objektet så ska användaren meddelas att landet är okänt.

Du får själv bestämma hur användargränssnittet i ditt program ska se ut. Det viktiga är att användaren får mata in namnet på ett land och att sedan programmet svarar med vad landets huvudstad heter. Och naturligtvis ska användaren få göra så många inmatningar av länder som hon önskar innan programmet avbryts.

Två tänkbara användargränssnitt är att använda dialogruter ungefär som figurerna ovan, eller att använda ett fönster med några komponenter:



I fönstret ovan är det två JLabel-komponenter (”Ange ett land:” och ”Tokyo”), en JTextField(”Japan”) och en JButton(”Huvudstad”).