Datalogi C
Henning Christiansen 15/11-2004

Opgaver til stoffet gennemgået ved forelæsningen 16/11-2004

Opgave 2 er en afleveringsopgave, afleveringsfrist 29/11 kl. 8.00, men vi forventes at begynde på den ved øvelserne. Af hensyn til evalueringen 30/11 skal afleveringsfristen overholdes (men det er ikke velset, at man bliver væk fra øvelserne 26/11 fordi man løser afleveringsopgave).

Opgave 1
Bogens opgave 20.5 a, b og 20.6 side 710.

Opgave 2 (Afleveringsopgave)

Java.util's HashMap er beregnet til at vedligeholde afbildinger, hvor der er en entydig sammenhæng fra et nøgle-objekt til en værdi-objekt. I database-terminologi svarer det til én-til-mange; det sidste »-mange« antyder at et givet objekt godt kan være værdi for forskellige nøgler. (Indenfor databaser taler man sjældent om »afbildninger«, men om »relationer«, men lad det nu være.)

Denne opgave handler om at konstruere en general klasse for mange-til-mange afbildninger, dvs. hvor der til givet nøgle kan høre mere end én værdi.

Som eksempel kan vi benytte en forenklet Dansk-Engelsk-ordbog, hvor man har udeladt enhver grammatisk opysning, og blot afbilder hvert ord på dansk over i ét eller flere engelske ord, der kan bruges som oversættelse, f.eks.

```
ben    -->   bone, leg
knogle -->   bone
```

Opgaven går ud på at konstruere en klasse HashManyMany, som opfører sig som en hash-tabel, men repræsenterer mange-til-mange afbildninger. Når vi siger »opfører sig som ...« betyder det, at metodekaldende i så høj grad som mulig skal ligne dem fra HashMap. Det nemmeste er at definere den som en specialisering af class HashMap, hvor man så afbilder nøglerne over i en eller anden form for Collection med et eller flere objekter i.

Klassen må altså defineres noget i stil med dette her:

```
class HashManyMany extends HashMap {

public Object put(Object key, Object value) { . . . }

public Object get(Object key){ . . . }
```

```
    ...     }
```

Nogle metoder fra HashMap fungerer godt nok når de nedarves til HashManyMany, men andre så som put og get må ændres.

Når put(n,v) kaldes, må den checke, om der findes en Collection allerede for n, og hvis ikke, må der skabes en, hvorefter v kan tilføjes til den aktuelle Collection.

En teknisk bemærkning: Collection er et interface, så vi må internt vælge en bestemt repræsentation (dvs. en ikke-abstrakt klasse som implementerer Collection), og her kan man f.eks. benytte ArrayList.

En teknisk bemærkning mere: Bemærk at der står »public Object get(Object key)« i skitsen ovenfor, og som svarer til det, man nedarver fra HashMap. Umiddelbart ville man nok forvente at »public *Collection* get(Object key)«, men Java tillader af en eller anden grund ikke at man ændrer resultattypen på denne måde.

En af de metoder, som (måske lidt overraskende) faktisk fungerer ganske udmærket uden at blive omdefineret er »toString«. I en test-implemention med følgende main:

```
public static void main(String [] args){
   HashManyMany h = new HashManyMany();
   h.put("ben","leg");
   h.put("ben","bone");
   h.put("knogle","bone");
   System.out.println(h);    }
```

give anledning til følgende udskrift

```
{ben=[leg, bone], knogle=[bone]}
```

Spørgsmål 1
Implementér klassen HashManyMany i Java som skitseret overfor med metoderne put og get med den sædvanlige betydning (tilpasset fra HashMap!) samt følgende metoder:

public boolean remove(Object key, Object value)

> Associationen mellem angivet key og value fjernes, men hvis der er andre values for key, skal de selvfølgelig bevares. Der returneres »true« hvis key-value fandtes allerede, »false« ellers.
> NB: Vær påpasselig med at hvis du sletter den sidste value for den givne key, bør du ikke efterlade en tom Collection under key, men fjerne den helt.
> NB: Der nedarves en remove-metode med ét argument fra HashMap, som stadig giver god mening, men den du skal lave her tager to argumenter.

public Iterator iterator(Object key)

Returnerer en Iterator, som vil kunne gennemløbe de elementer (values) som key er bundet til.

Spørgsmål 2
Gå listen af metoder, som nedarves fra HashMap (se http://java.sun.com/j2se/1.4.2/docs/api/java/util/HashMap.html), igennem, og for hver enkelt skal du kommentere hvorvidt det giver god mening at genbruge disse i HashManyMany, eller hvordan de burde ændres (du behøver ikke implementere dem, men du må gerne).

Spørgsmål 3.
Tilføj en ny metode til klassen

```
public HashManyMany inverse(){ . . . }
```

som returnerer en ny HashManyMany-afbildning som repræsenterer den omvendte afbildning af den aktuelle. Dvs. hvis den aktuelle indeholder (bl.a.) Søren --> Karen, skal den nye indeholde Karen --> Søren. Når test-implementationen omtalt ovenfor blev udvidet med følgende linier i sin main

```
HashManyMany hInverse = h.inverse();
System.out.println(hInverse);
```

fremkom følgende udskrift:

```
{bone=[ben, knogle], leg=[ben]}
```

(Dvs. i dette eksempel svarer »inverse« til at lave Dansk-Engelsk-ordbogen om til en Engelsk-Dansk-ordbog.)

Spørgsmål 4
Konstruér en lidt større HashManyMany-afbildning og demonstrér ved test-udskrifter, at løsningerne på spørgsmål 1 og 3 fungerer som de skal.

Gode råd:
Opgaven kræver ikke mange kodelinjer, og man behøver slet ikke tænke over detaljerne omkring hvordan hashtabeller fungerer internt. Det kan oplyses at din lærers test-implementation fylder 34 linjer, excl. kommentarer og main-metode. Det svære i opgaven ligger i at navigere rundt i Java's API og at få diverse type-castings o.lign. til at makke ret.