**HashMap**

Een map is een **verzameling van koppels = key – value – pairs - objectenparen**.   
Elk koppel bestaat uit een **sleutelobject** en uit een bijhorend **waardeobject**.  
Voorbeeld.  
Key = studentnummer  
Value = naam van de student.

OF

Key = studentnummer  
Value = het volledige object van de student. Een instantie van de klasse Student die alle private variabelen van de student bevat.

Net als een ArrayList kan een map een flexibel aantal records bevatten.   
Het soort element bij een HashMap is wel anders.   
De elementen van een HashMap zijn objectenparen.  
Een paar = een sleutelobject en een waardeobject.   
I.p.v. te zoeken op basis van een indexnummer, gebruiken we het sleutelobject om het waardeobject te zoeken.   
Bv. een telefoonboek:   
elk record bestaat uit een naam en een telefoonnummer.   
Een map wordt meestal zo geordend dat ze heel geschikt is voor zoekopdrachten in één richting – van naam naar nummer in ons telefoonboek. Omgekeerd zoeken kan ook, maar kost meer tijd.  
De twee belangrijkste methodes:   
**put** : voegt een record toe.   
**get** : haalt de waarde voor een sleutel op.

Uitgewerkt voorbeeld : telefoonboekUiteraard raadplegen we de Oracle documentatie.  
 **STAP1**

Maak een class Telefoonboek met een HashMap voor private contacten.  
Omdat we een eenvoudig voorbeeld willen uitwerken, gaan we er van uit dat elke naam maar 1 keer in het telefoonboek staat. Het gaat hier immers over een persoonlijk contactenboek, geen echt telefoonboek zoals die vroeger verspreid werden.  
   
Plaats er de volgende objectenparen in.

|  |  |
| --- | --- |
| **KEY** | **VALUE** |
| Jan | 123 |
| Jos | 456 |
| Jozef | 789 |

private HashMap<String, String> contacts = new HashMap<>();  
*//eerste waarde = sleutel (naam),   
//tweede waarde = waarde (telefoonnummer)   
//objecten-paren!*public Telefoonboek() {  
 contacts.put("Jan", "123");  
 contacts.put("Jos", "456");  
 contacts.put("Jozef", "789");  
}

**STAP2**

Maak een functie die controleert hoeveel records een map heeft.

public int telContacten() {  
 return contacts.size();  
}

**STAP3**

Maak een functie waarmee je op basis van de naam (=key) het overeenkomstige telefoonnummer (=value) opzoekt.

public String zoekTelefoonnummerOp(String naam) {  
 return contacts.get(naam);  
}

TO DO : Breid de functie zodat de functie “unknown” returnt indien de naam niet voorkomt in het telefoonboek.  
TO DO : Maak een functie waarmee je een objectenpaar in het telefoonboek kan plaatsen. Deze functie heeft twee parameters : naam en telefoonnummer.  
TO DO : maak een functie die true returnt indien een naam voorkomt in het telefoonboek en false indien niet.  
Welke functie kan je hiervoor gebruiken?

**STAP4**

Maak een functie die alle sleutels (namen) afdrukt.

public void alleNamen() {  
 for (String naam : contacts.keySet()) {  
 System.*out*.println(naam);  
 }  
}

TO DO : maak een functie die niet alleen de namen uitprint, maar ook de bijhorende telefoonnummers.

Oefeningen Hashmap

**Oefening 1 : wachtrij**

Maak een wachtrij van mensen. Gebruik hiervoor een HashMap.  
Key : naam.  
Value : WACHTEND of BEDIEND. **Werk hiervoor met enum.**Met enum hebben we nog niet gewerkt.  
**Wat is enum?**<https://www.w3schools.com/java/java_enums.asp>

Maak functies voor  
-persoon toevoegen aan de wachtrij.  
-status van WACHTEND naar BEDIEND veranderen.  
-een overzicht maken van de wachtende en de bediende personen.  
-tellen hoeveel wachtende en hoeveel bediende personen.

-een functie maken die de bediende personen uit de wachtrij schrapt. Deze code krijg je en we leren hier meteen weer iets nieuws, namelijk werken met een **Iterator-object**.  
**Een Iterator-object is een object waarmee alle elementen van een collectie één voor één kunnen bekeken worden.**

public void verwijderBediend() {  
  
 Iterator it = wachtrij.entrySet().iterator();  
 while (it.hasNext()) {  
 Map.Entry pair = (Map.Entry) it.next();  
 *//System.out.println(pair.getKey() + " = " + pair.getValue());* if (pair.getValue().equals(Toestand.*BEDIEND*.name())) {  
 it.remove();  
 }  
  
 }

}

Anne2  
WACHTEND

Anne1  
WACHTEND

Anne3  
BEDIEND

Anne4  
BEDIEND

Anne5  
BEDIEND

it

**Hoe werkt bovenstaande code?**  
STAP1 :  
Maak een variabele it van **de interface Iterator** aan.   
Doel : Deze Iterator-variabele kan elk paar van de HashMap apart aanwijzen.  
Een Iterator object wordt aangemaakt door **de functie iterator().**  
De aanroep van deze methode iterator() gebeurt 1 keer voor de HashMap.  
Je moet wel expliciet vertellen dat je over de **paren** wil itereren en dat doe je door de functie entrySet() te gebruiken.   
Lees de documentatie van de functie entrySet().   
  
Iterator it = wachtrij.entrySet().iterator();  
  
STAP2 :   
Van de Iterator klasse gebruiken we 2 functies : **next()** en **hasNext().**

De manier waarop we een Iterator gebruiken, kunnen we als volgt beschrijven in **pseudo-code.**  
  
 public void gebruikIterator() {  
 Iterator it = wachtrij.entrySet().iterator();  
 *//de conditie test of er nog een volgend element is in de collectie   
 //waarnaar de iterator kan wijzen  
 //deze conditie is onwaar als alle elementen van de collectie overlopen   
 //zijn door de iterator.* while (it.hasNext()) {  
 *//wijs met de iterator naar het volgende element in de collectie  
 //de eerste keer zal de iterator naar het eerste element wijzen,   
 //de tweede keer naar het volgende element, enz...* it.next();  
 *//doe iets met dat object* }  
}

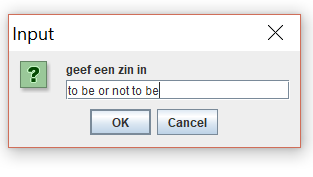
while (it.hasNext()) {  
 Map.Entry pair = (Map.Entry) it.next();  
}

Map.Entry  
Dit is het datatype van 1 key-value-pair uit de HashMap.

STAP3:

if (pair.getValue().equals(Toestand.*BEDIEND*.name())) {  
 it.remove();  
 }

Oefening 2 : woorden tellen

Gebruik een Hashmap om te tellen hoeveel keer een woord voorkomt in een opgegeven zin.   
Vereenvoudiging : geen hoofdletters en leestekens.  
  
Voorbeeld  


not: 1

be: 2

or: 1

to: 2

Definieer de volgende HashMap.

private HashMap<String, Integer> woordenTel = new HashMap<>();

Maak een constructor die de zin inleest.  
  
Maak een functie die de woorden uit de zin haalt en in de HashMap zet. Nakijken of het woord al voorkomt. Indien wel, dan wordt de teller verhoogd.  
  
Maak een functie die de inhoud van de HashMap toont.

Maak een functie die de woord(en) toont die het meest voorkomen. Plaats de woorden die het meest voorkomen in een arraylist. Verwijder deze entries (woorden) uit de HashMap.

Oefening 3 : Hashmaps samen voegen

Voeg twee hashmaps samen volgens de volgende regels.

public class Samenvoegen {

private HashMap<String, String> map1 = new HashMap<>();

private HashMap<String, String> map2 = new HashMap<>();

private HashMap<String, String> samen = new HashMap<>();

public Samenvoegen() {

map1.put("a", "b");

map2.put("b", "c");

//wordt in samen ("a", "c")

//

map1.put("d", "e");

map2.put("e", "f");

//wordt in samen ("d", "f")

map1.put("g", "h");

map2.put("i", "j");

//wordt geen paar in samen want "h" is niet gelijk aan "i"

}

Oefening 4

Om boodschappen te coderen en te decoderen, maken we gebruik van een coderingstabel. We laten bijvoorbeeld met elke letter van het alfabet, met elk leesteken (voor de gemakkelijkheid, enkel met een . werken) en met een spatie een getal overeenkomen. Vervolledig de constructor en de methodes van de klasse Sleutel.

**public class Sleutel**   
{   
 private HashMap<Character, Integer> secretMap = ?;

}

**private void fillsecretMap()**

{

secretMap.put(' ',0); // spatie

secretMap.put('A',1);

// nog verder aan te vullen

secretMap.put('Z',26);

secretMap.put ('.',27); // punt

..

}

//Tekens en bijbehorende codes worden samen op het scherm getoond

**public void printSleutel ()**

{ // nog verder aan te vullen }

}

//Voorbeeld: de boodschap " Er was eens ... " wordt gecodeerd als:   
// "5 18 0 23 1 19 0 5 5 14 19 0 27 27 27"   
//let op: tussen elke cijfer staat een spatie, anders kan je niet meer decoderen.

**public String codeerBoodschap(String boodschap)**

{ // nog verder aan te vullen }

// Voorbeeld: de boodschap in cijfercode "5 18 0 23 1 19 0 5 5 14 19 0 27 27 27"   
// wordt gedecodeerd als: "ER WAS EENS ..."

Bekijk de documentatie van Set, HashSet, List en ArrayList.  
Probeer de overeenkomsten en verschillen tussen een HashSet en ArrayList te weten te komen.  
Een set is een collectie waarin elk afzonderlijk element slechts éénmaal kan worden opgeslagen.   
De set houdt geen bepaalde volgorde in stand.   
Items toevoegen werkt zoals bij een ArrayList.   
HashSet mySet = new HashSet();  
mySet.add(" een " );   
mySet.add(" twee " );   
  
Ook elementen doorzoeken is erg gelijkaardig.

for(String item : mySet)   
{…}  
   
In een set worden de items niet in een bepaalde volgorde bewaard.

Extra oefeningen  
value = hele student en de key is een studentnummer.

Bot ontwerpen.