TDDC77 Objektorienterad Programmering

Föreläsning 11

Sahand Sadjadee
IDA, Linköpings Universitet

Outline

- Collections(Kollektioner)
- Hashing
- Instanceof/Casting(demo)

Collections(Kollektioner)

Att lagra värden

- En variabel: Data av en sort
- Problem: om det blir många variabler
- En array: En rad av data av samma sort
- Problem: Måste kunna storleken i förväg, även svårt att bearbeta data i arrayen (söka, sortera osv)

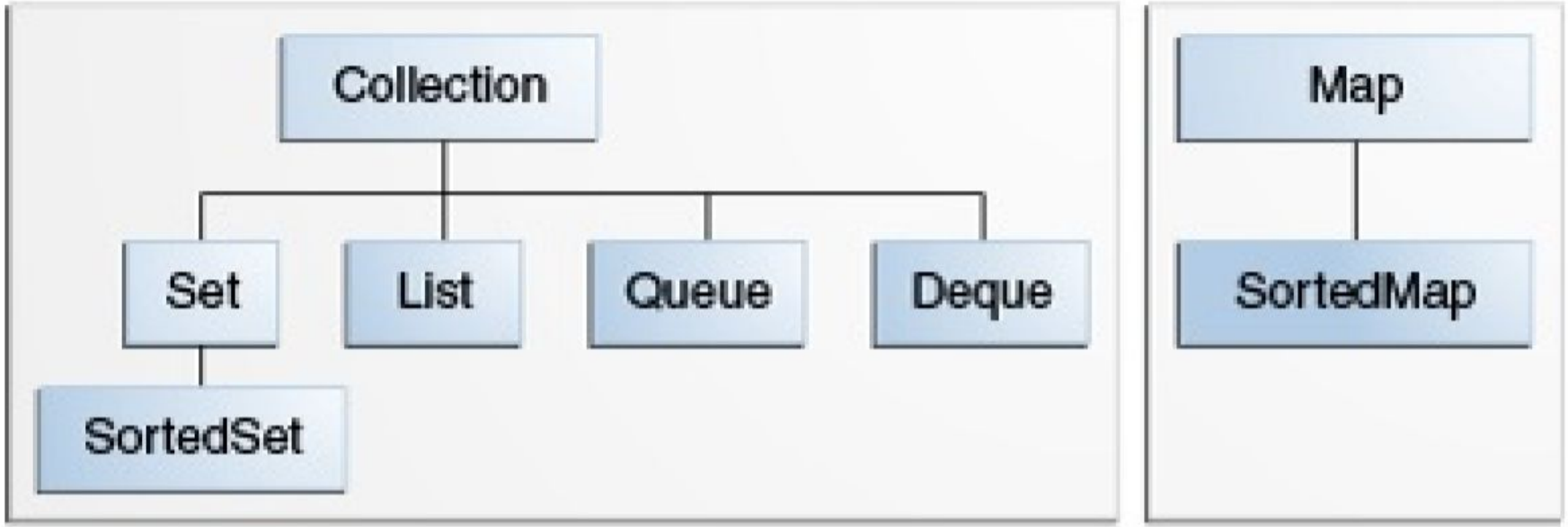
Kollektion (Collection)

- är ett objekt som grupperar flera element i en enda enhet.
- används för att lagra, hämta, manipulera, och kommunicera aggregerad data.
- bildar en naturlig grupp såsom en en post mapp (en samling av brev) eller en telefonkatalog (en kartläggning av namn till telefonnummer).

Lösningen heter Collections

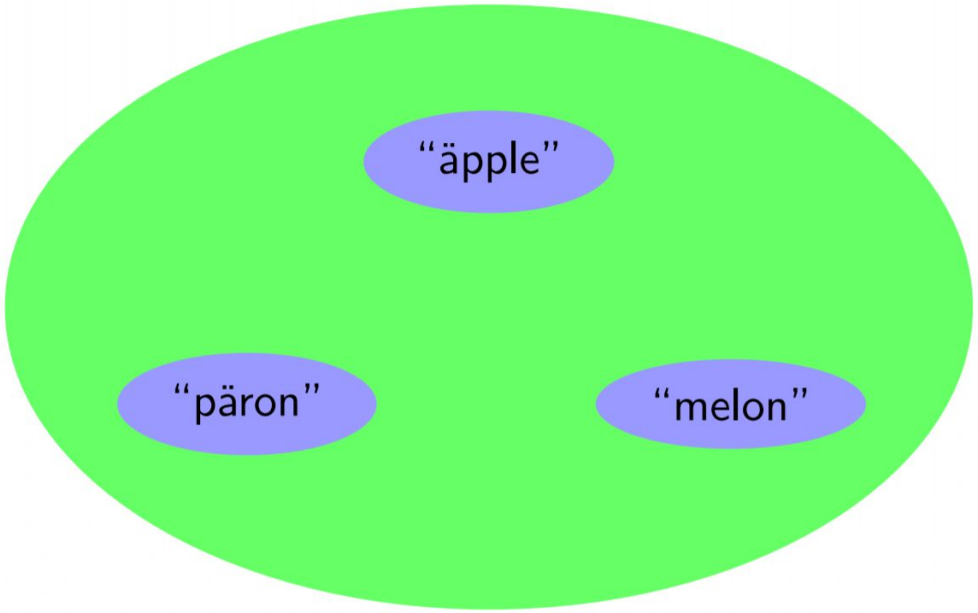
- Samling av klasser för effektiv och avancerad lagring av data
- Finns i paketet `java.util`
- Är uppdelade i interface och klasser
- Sök efter “Collection” i API:et
- https://docs.oracle.com/en/java/javase/11/docs/api/index.html
- Finns även bra tutorials, ex http://docs.oracle.com/javase/tutorial/collections

Lösningen heter Collection



Set - Mängd

- osorterad mängd av data utan kopior
- Interfacet heter `Set`
- Implementationer: exempelvis `HashSet, SortedSet`



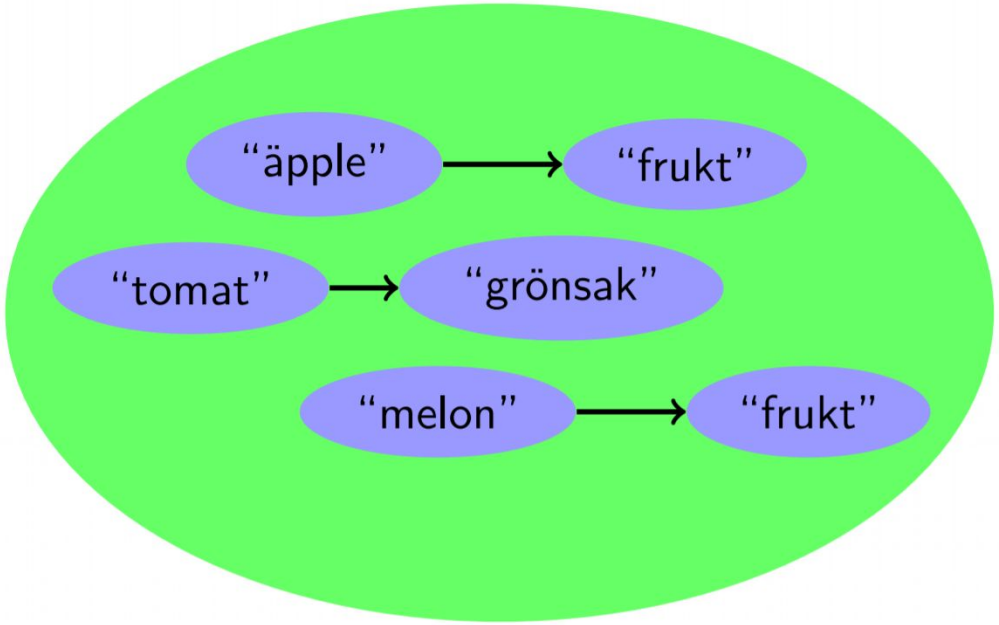
Set

```
import java.util.*;

/* The program FindDuplicates prints messages for each duplicated
 * argument it is passed.
 * example: java FindDuplicates i came i saw i left
 */
class FindDuplicates {
    public static void main(String[] args) {
     Set<String> minSet = new HashSet<String>();
     for (int i=0; i<args.length; i++)
            if (minSet.add(args[i]) == false)
                System.out.println("Duplicate detected: " + args[i]);

        System.out.println(minSet.size() + " distinct words: " + minSet);

      }
}

//      for(String str: args)
```

LINKÖPINGS UNIVERSITET

Map

- par av data, key -> value.
- Interfacet heter `Map`
- Implementationer: exempelvis `HashMap`



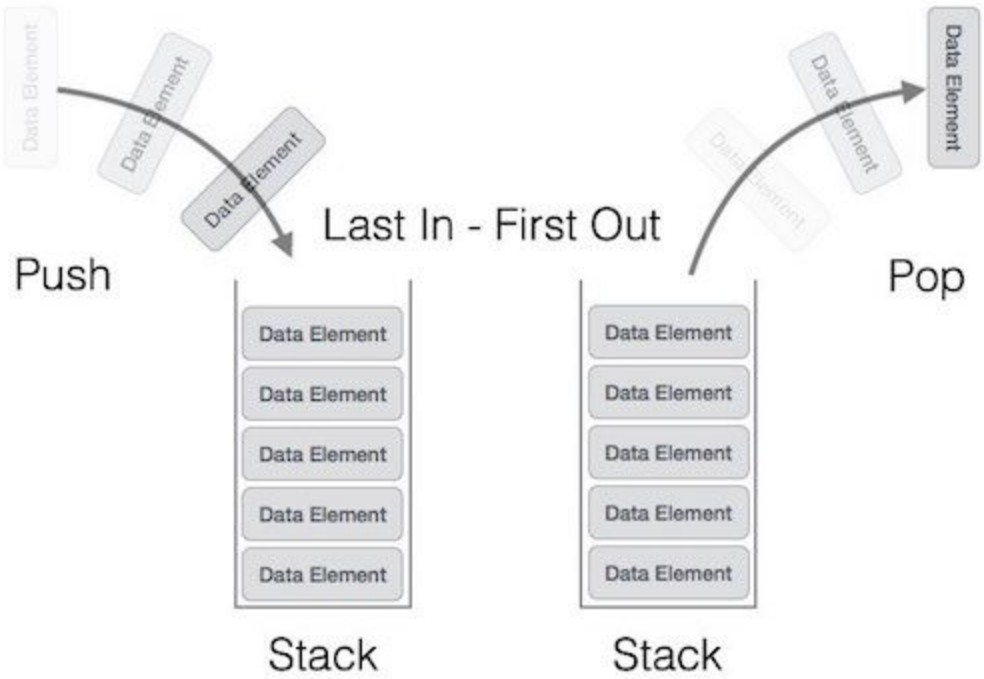
Map

```
import java.util.*;

public class Freq {
    public static void main(String[] args) {
        Map<String, Integer> m = new HashMap<String, Integer>();

        // Initialize frequency table from command line
        for (String a : args) {
            Integer freq = m.get(a);
            m.put(a, (freq == null) ? 1 : freq + 1);
        }

        System.out.println(m.size() + " distinct words:");
        System.out.println(m);
    }
}

//java Freq if it is to be it is up to me to delegate
```

Klassen Collections

- Klassen Collections har statiska metoder för att manipulera listor, mängder osv.
  - `frequency()`
  - `min(), max()`
  - `reverse()`
  - `sort(), shuffle()`

Stack


Data Element
Data Element
Data Element
Last In - First Out
Push
Pop
Data Element
Data Element
Data Element
Data Element
Data Element
Stack
Data Element
Data Element
Data Element
Data Element
Data Element
Stack

Stack Operations:

- **push()**: Insert a new element into the stack i.e just insert a new element at the beginning of the linked list.
- **pop()**: Return the top element of the Stack i.e simply delete the first element from the linked list.
- **peek()**: Return the top element.

Klassen ***`LinkedList`*** innehåller alla metoder som behövs för att kunna använda den som en stack. Klassen ***`LinkedList`*** kan också användas som en vanlig lista.

https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

Queue


In
Data
Data
Data
Data
Data
Data
Out
Last In Last Out
First In First Out
Queue

Queue Operations:

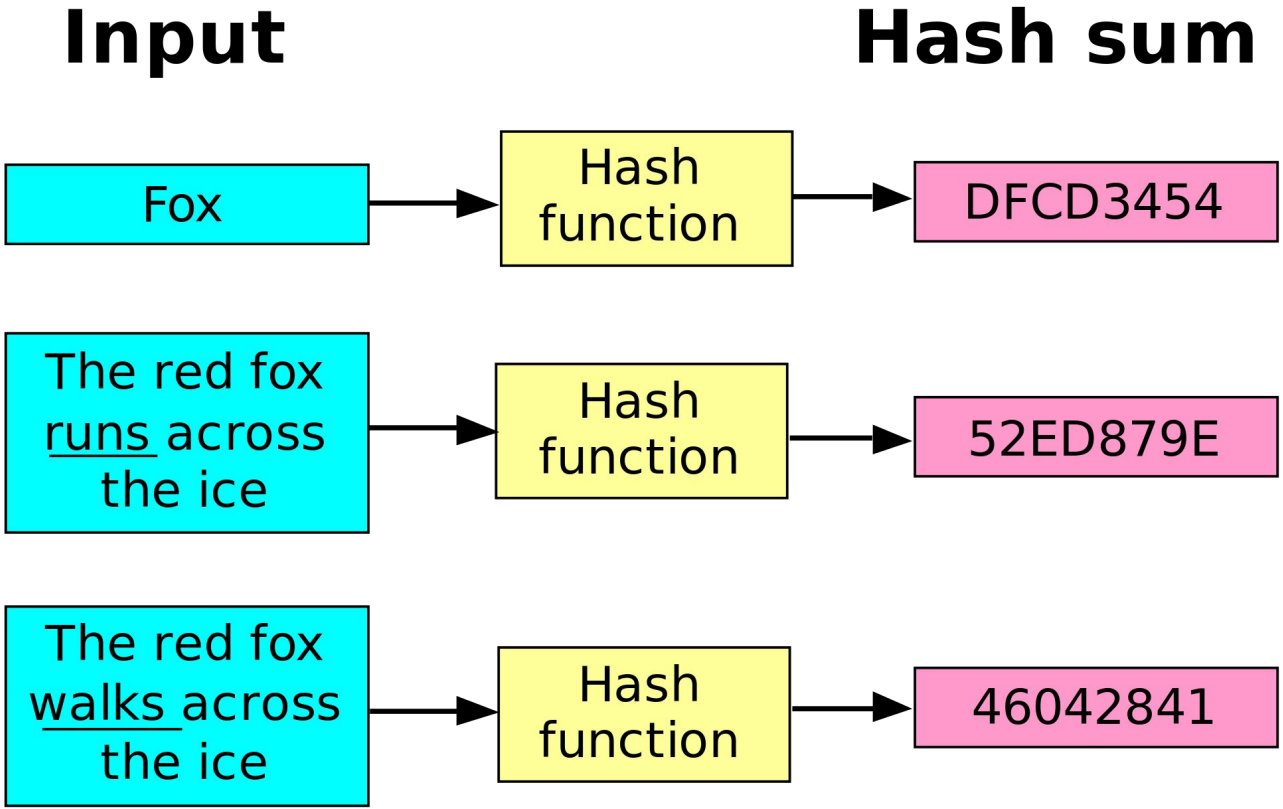
- enqueue() – add (store) an item to the queue.
- dequeue() – remove (access) an item from the queue.
- peek() – Gets the element at the front of the queue without removing it.
- isfull() – Checks if the queue is full.
- isempty() – Checks if the queue is empty.

Klassen ***`LinkedList`*** innehåller alla metoder som behövs för att kunna använda den som en queue.

https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

Hashing

Hashing



LINKÖPINGS UNIVERSITET

Hashing

- Idéen är att beräkna ett så kallad hash för varje objekt .
- Hashfunktionen måste alltid returnera samma värde för samma objekt, och för alla andra objekt där `equals()` returnerar true.
- Olika objekt får ha samma hashkod men det kan påverka HashMaps prestation negativt.
- För er ordlistlaboration så räcker det bra att returnera värdet för textsrängens `haschCode()`.

Hashing

```
class Fruit {
    private String name;

    public Fruit(String name){this.setName(name);}

    public String getName() {return name;}

    public void setName(String name) {this.name = name;}

    public String toString(){return name;}

    public int hashCode(){return name.hashCode();}

    public boolean equals(Object other){
     if(other==null || !(other instanceof Fruit))
         return false;
     return name.equals(((Fruit)other).name);
    }
}
```

LINKÖPINGS UNIVERSITET

Hashing

- Om man inte åsidosätter `equals` i egen klass då är “alla” objekt olika enligt `equals` default implementation.
- Om man inte åsidosätter `hashcode` i egen klass då “alla” objekt av klassen får samma hashkod även om de inte är lika enligt `equals` default implementation.

Hashing

- I Alla klasser som innehåller Hash i namnet använder något som heter hashing för att öka hastigheten på vissa operationer, exempelvis när man ska göra snabba sökningar i en lista.
- Det ingår inte i kursen att förstå exakt hur det fungerar, men ni måste förstå tillräckligt för att kunna använda det.

Hashing

```
import java.util.*;

//Basket päron melon apelsin melon
public class Basket {

    public static void main(String[] args) {
     Set<Fruit> set = new HashSet<Fruit>();
     for(String in: args)
         set.add(new Fruit(in));

     System.out.println("Mängden av olika input är: " + set);
    }
}
```

Thanks for listening!

LINKÖPINGS UNIVERSITET