DD2385
Programutvecklingsteknik
Några bilder till föreläsning 6
22/4 2014

Innehåll

- Ramverk
- Javas objektsamlingar
  med mönstret Iterator
- Generiska typer i Java
- Omslagsklasserna

Ramverk är en uppsättning

- Återanvändbara
- Generella
- Samarbetande

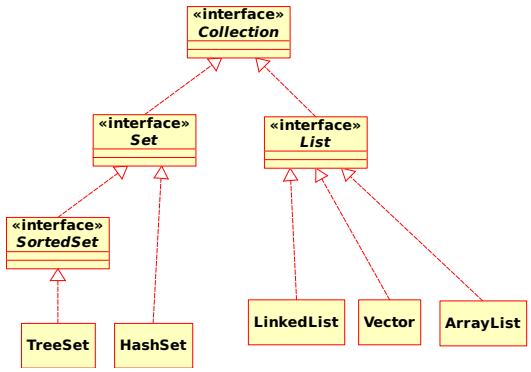
klasser för en kategori av tillämpningar

Ramverk i Java-API:n (exempel)

`awt` `swing` `Collections` `Maps`

Collection

Javas bibliotheksklasser för objektsamlingar,
här visas *en del av* klasshierarkin:



Några metoder i `Collection`.

Alla implementerande klasser har dessa

```
add(obj)
addAll(coll)
iterator()
remove(obj)
removeAll(coll)
retainAll(coll)
toArray()
isEmpty()
size()
```

List

- Numrerad samling objekt
- Dynamisk
  (kan växa och krympa, jfr `[]`-array)
- Insättning och borttagning görs var som helst i listan

Ett objekt kan finnas på flera ställen i en lista

Det gäller `[]`-arrayer också!

ArrayList, Vector

- Lagras m.h.a. `[]`-arrayer
- Bäst för snabb åtkomst/ändring av listelement.

LinkedList

- Lagras länkat
- Bäst för snabb insättning/borttagning av listelement

Generiska typer

En objektsamling kan skapas utan att elementtypen anges:

```
ArrayList alist = new ArrayList();
```

Element av typ `Object` antas men varningsmeddelande ges då listan används.

Collection-klasserna kan ges typparameter

`ClassName<ElementType>`

Generiska typer, forts.

Större säkerhet och enklare hantering med

```
CollType<ElementTyp> coll =
        new CollType<ElementTyp>();
```

Till exempel:

```
ArrayList<String> alist =
        new ArrayList<String>();
```

Endast `String`-objekt kan sättas in i listan

Några metoder i `Collection`.

Alla implementerande klasser har dessa

```
add(obj)
addAll(coll)
iterator()
removeAll(coll)
retainAll(coll)
toArray()
remove(obj)
isEmpty()
size()
```

Mönstret Iterator

- Ger elementen ur en komplicerad struktur
  i sekvens
- Avslöjar ej detaljer om lagringen
- Implementerar ett `Iterator` – gränssnitt, t.ex.

```
interface Iterator {
  Object first();
  Object next();
  boolean isDone();
  Object current();
}
```

Iteratorn känner till den underliggande strukturen
och "vet" hur långt den kommit i iterationen

Iteratorn i Java-API:n

```
interface Iterator<T> {
  T next();
  boolean hasNext();
  void remove();
}
```

Alla objektsamlingar har Iterator !!

Några metoder i `List`.

add(obj)	sätt in sist
add(k,obj)	sätt in på plats k
get(k)	referens till objektet på plats k returtyp Object eller T T = typparametern till objektsamlingen
remove(k)	tag bort elementet på plats k

Set

- Ett objekt kan bara finnas med en gång.
- Använder `equals()` från `Object`

HashSet

- Snabb åtkomst
- Använder `hashCode()` från `Object`

SortedSet

- Hålls sorterad
- Elementen måste implementera interfacet `Comparable<T>` eller `Comparable`

`Comparable` utan typparameter

```
public interface Comparable {
    public int compareTo (Object obj);
}
```

`Comparable` med typparameter

```
public interface Comparable<T> {
    public int compareTo (T obj);
}
```

`compareTo()` returnerar <0, 0 eller >0 beroende på ordningen mellan elementen.

`t1.compareTo(t2)` <0 om `t1` kommer före `t2`.

Omslagsklasser

- Ibland behövs objekt av primitiva data
- T.ex. för en objektsamling av primitiva data
- I Javabiblioteket finns en klass för varje primitiv typ

`Byte`	`Boolean`
`Short`	`Character`
`Integer`	`Float`
`Long`	`Double`

Omslagsklasser, forts

- Omslagsklasserna innehåller många metoder, t.ex. för omvandling mellan `String` och aktuell typ. Se dokumentationen!
- Autoboxing = automatisk konvertering mellan primitiv typ och dess omslagstyp.

Exempel

`Integer.MAXVALUE` är största möjliga `int` – värde.

`Integer.parseInt(astring)` omvandlar textsträng till `int`, t.ex. vid textinmatning.

```
int finatalet = Integer.parseInt("1729");
Integer x = new Integer(27);
Integer y = 38;      // autoboxing
int z = x*y;         // autoboxing igen
```

Gör inte autoboxing i onödan!

Exempel med Set

Generera
goal st olika tal från intervallet `1..range`.

tries räknar antalet försök.

range $\geq$ goal nödvändigt.

Använd subklassen HashSet.

- Slumpa tal, lägg i en mängd
- När mängden har goal st tal är det klart (vi vet ju att det är olika tal)

```
HashSet<Integer> hset =
  new HashSet<Integer>();

int tries = 0;

while (hset.size() < goal){
  int next =
    (int)(Math.random()*range+1);
  hset.add(next);    // autoboxing
  tries++;
}
```

Skriv ut talen ur hset

Med for-sats:

```
for (int i : hset)
    System.out.print("␣" + i);
System.out.println();
```

Med iterator:

```
Iterator<Integer> iter = hset.iterator();

while (iter.hasNext())
    System.out.print("␣" + iter.next());
System.out.println();
```

Varning för osäker operation:

Att ta bort element ur en lista medan man loopar över listan.

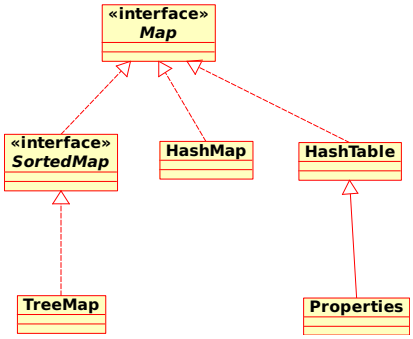
Via `iterator()` i Java-collections går det bra:

Tag bort alla 5-multipler ur hset via iter:

```
while (iter.hasNext()){
  int x = iter.next(); // autoboxing
  if (x%5 == 0)
    iter.remove();
}
```

Map

Bibliotheksklasser för avbildningstabeller, en del av klasshierarkin



Map

Objekt lagras tillsammans med en nyckel som också är ett objekt.

Map liknar dictionaries i Python

Några metoder i Map

`put(key,val)`	lägg in val med nyckel key
`remove(key)`	tag bort
`get(key)`	ger referens till objektet
`keySet()`	alla nycklar
`size()`	antal element