ADT preslikava

Operacije definirane za **ADT MAPPING**:

- **MAKENULL(M)** – inicializira prazno preslikavo
- **ASSIGN(M, d, r)** – definira, da je M(d) = r
- **COMPUTE(M, d)** – vrne vrednost M(d), če je definirana, sicer **null**

Pri implementaciji v javi lahko definiramo vmesnik:

```
public interface Mapping {
   public abstract void makenull() ;
   public abstract void assign(Object d, Object r) ;
   public abstract Object compute(Object d) ;
}
```

Metode vseh zbirk

```
public interface Collection
{
        // osnovne operacije
        int size();
        boolean isEmpty();
        boolean contains(Object element);
        boolean add(Object element);        // opcijsko
        boolean remove(Object element);     // opcijsko
        Iterator iterator();

        // množične operacije
        boolean containsAll(Collection c);
        boolean addAll(Collection c);       // opcijsko
        boolean removeAll(Collection c);    // opcijsko
        boolean retainAll(Collection c);    // opcijsko
        void clear();                       // opcijsko

        // pretvorba v polje
        Object[] toArray();
        Object[] toArray(Object a[]);
}
```

Iteratorji

Iteratorji omogočajo sprehod prek vseh elementov zbirke. Vmesnik:

```
Interface Iterator {
    boolean hasNext(); // preveri, ali je na voljo še kak element
    Object next(); // vrne element ter se prestavi za mesto naprej
    void remove(); // odstrani element
}
```

Primer:

```
public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<String>();
    list.add("Primer");
    list.add("elementov");
    list.add("seznama");

    Iterator<String> itr = list.iterator();
    while (itr.hasNext()) {
        System.out.println(itr.next());
    }
}
```

Seznami

Implementacija **s poljem** (**ArrayList**) in **s kazalci** (**LinkedList**).

POLJE: seznam se malo spreminja, pogosto dostop preko indeksov

KAZALCI: seznam se pogosto spreminja, redko dostop preko indeksov

Pomembnejše podedovane funkcije od vmesnika **Collection**:

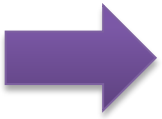
- `add(Object)` – doda nov element na konec seznama
- `remove(Object)` – odstrani prvo pojavitev elementa v seznamu

Metode za **dostop do elementov** preko indeksov (pričnejo se z 0):

- `add(int, Object)`
- `get(int)`
- `set(int, Object)`
- `remove(int)`

Funkcije za **iskanje elementov**:

- `indexOf(Object)`
- `lastIndexOf(Object)`



Seznam najpomembnejših operacij za razreda ArrayList in LinkedList se nahaja v učbeniku, str. 125 in 126

Množice

Neurejena je implementirana z **zaprto zgoščeno tabelo s ponovnim zgoščanjem** (**HashSet**)
Urejena množica je implementirana z **rdeče-črnim drevesom** (**TreeSet**).

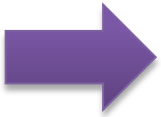
Metode za **dodajanje, iskanje in brisanje elementov**:	HashSet	TreeSet
• `add(Object e)`	`O(1)`	`O(log n)`
• `Contains(Object e)`		
• `remove(Object e)`		

Metodi za **velikost množice**:

- `isEmpty()`
- `size()`

Metode za **operacije glede urejenosti** (samo **TreeSet**):

- `first()` // vrne prvi (min element
- `last()` // vrne zadnji (max) element
- `headSet(Object to)` // vrne podmnožico elementov manjših od to
- `tailSet(Object from)` // vrne podmnožico elementov večjih od from
- `subSet(Object f, Object t)` // vrne podmnožico elementov med f in t



Tabeli najpomembnejših operacij za razreda HashSet in TreeSet se nahajata v učbeniku, strani 127 in 128

Množice

Metode za **množične operacije** omogočajo izvedbo algebre množic:

- **unija**: `s1.addAll(s2);`
- **presek**: `s1.retainAll(s2);`
- **razlika**: `s1.removeAll(s2);`
- **Ugotavljanje podmnožice**: `s1.containsAll(s2);`

Preslikave

Preslikava ***brez urejenosti*** *po ključih* je implementirana z
zaprto zgoščeno tabelo s ponovnim zgoščanjem (**HashMap**),
Preslikava z ***urejenimi pari*** *po ključih* je implementirana z **rdeče-črnim drevesom** (**TreeMap**).

Metode za **dodajanje, računanje in brisanje parov**:

- `put(Object key, Object value)`
- `remove(Object key)`

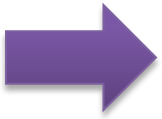
Metoda za **iskanje vrednosti za dani ključ**:

- `get(Object key)`

	HashMap	TreeMap
	O(1)	O(log n)

Metode za poglede na preslikavo skozi zbirke

- `keySet()` // vrne množico vseh ključev preslikave
- `values()` // vrne zbirko vseh vrednosti
- `entrySet()` // vrne množico parov preslikav ključa v vrednost



Tabeli najpomembnejših operacij za razreda HashMap in TreeMap se nahajata v učbeniku, strani 130 in 131