# MAPPE E DIZIONARI

## DEFINIZIONE DI MAPPA

- ADT che contiene dati come coppie chiave/valore
- Ogni chiave deve essere **UNICA**
  - viene utilizzata come identificatore
  - utilizzata per effettuare ricerca/rimozione elementi

## DEFINIZIONE DI DIZIONARIO

- Simile ad una mappa
- **NON** si richiede che le chiavi siano uniche
  - possono essere associate a **più valori**
- Si distinguono in:
  - Dizionari ordinati
    - chiavi ordinate
  - Dizionari non ordinati
    - chiavi non ordinate
- La trattazione si soffermerà su dizionari a chiave unica (= mappe)

## INTERFACCIA

```
public interface Dictionary extends Container {
    void insert(Comparable key, Object value);
    void remove(Comparable key);
    Object find(Comparable key);
}
```

- Definiamo una nuova **eccezione**
  - `class DictionaryItemNotFoundException extends RuntimeException`

## IMPLEMENTAZIONE

- Struttura dati: array riempito solo in parte
- Il dizionario contiene oggetti di tipo `Pair`
  - coppie chiave/valore

- È possibile scegliere se:
  - mantenere chiavi **ordinate**
    - performante nella lettura (si può usare binary search)
  - lasciare chiavi **non ordinati**

## LA CLASSE PAIR

- Contiene elementi formati da **coppie chiave/valore**
- Due **campi di esemplare**
  - key (Comparable)
  - value (Object)
- Metodi di accesso e modificatori per questi campi di esemplare

## IMPLEMENTAZIONE DIZIONARIO NON ORDINATO

```java
public class ArrayDictionary implements Dictionary {
    protected Pair[] array;
    protected int arraySize;
    protected final int INIT_SIZE = 100;

    public ArrayDictionary() {
        array = new Pair[100];
        makeEmpty();
    }

    public boolean isEmpty() {
        return arraySize == 0;
    }

    public void makeEmpty() {
        arraySize = 0;
    }

    public String toString() {
        String s = "";
        for (int i = 0; i < arraySize; i++) {
            s = s + array[i] + "\n";
        }
        return s;
    }

    public void insert(Comparable key, Object value) {
        if (key == null)
            throw new IllegalArgumentException();
        try {
            remove(key);
        } catch (DictionaryItemNotFoundException e) {
        }

        if (arraySize == array.length)
            array = resize(2 * arraySize);
        array[arraySize++] = new Pair(key, value);
    }

    protected Pair[] resize(int newLength) // solita tecnica
    {
        if (newLength < array.length)
            throw new IllegalArgumentException();
        Pair[] newArray = new Pair[newLength];
        System.arraycopy(array, 0, newArray, 0, array.length);
        return newArray;
    }

    public void remove(Comparable key) {
        array[linearSearch(key)] = array[--arraySize];
    }

    public Object find(Comparable key) {
        return array[linearSearch(key)].getValue();
    }

    private int linearSearch(Comparable key) {
        for (int i = 0; i < arraySize; i++) {
            if (array[i].getKey().compareTo(key) == 0) {
                return i;
            }
        }
        throw new DictionaryItemNotFoundException();
    }

    protected class Pair // classe interna ad ArrayDictionary
    {
        ...
    }

}
```

## IMPLEMENTAZIONE DIZIONARIO ORDINATO

- L'array deve sempre essere mantenuto ordinato con Insertion Sort
  - ogni volta che si inserisce o toglie un elemento
- La ricerca deve avvenire attraverso **binary search** in quanto l'array è ordinato

## PERFORMANCE DIZIONARIO

### DIZIONARIO CON ARRAY NON ORDINATO

- **Ricerca**
  - prestazione $O(n)$
  - ricerca lineare
- **Rimozione**
  - prestazione $O(n)$
  - bisogna effettuare ricerca lineare e poi spostare nella posizione trovata l'ultimo element
- **Inserimento**
  - prestazione $O(n)$
  - Bisogna rimuovere (sovrascrivere) un elemento con la stessa chiave, se c'è, e poi inser nuovo elemento nella ultima posizione dell'array (l'ordinamento non interessa)

### DIZIONARIO CON ARRAY ORDINATO

- **Ricerca**
  - prestazione $O(\log n)$
  - binary search
- **Rimozione**
  - prestazione $O(n)$
  - bisogna effettuare una ricerca e poi spostare mediamente $\frac{n}{2}$ elementi

- **Inserimento**
  - prestazione $O(n)$
  - o con algoritmo diverso $O(n \log n)$
  - si può usare insertion sort in un array ordinato

| Dizionario | array ordinato | array non ordinato | |
|---|---|---|---|
| ricerca | O(lg n) | O(n) | |
| inserimento | O(n) | O(n) | [O(1) per chiavi non uniche |
| rimozione | O(n) | O(n) | |

**Prestazioni in sintesi:**

- find: $O(\log n)$ (ordinato) - $O(n)$ (non ordinato)
- insert: $O(n)$
- remove: $O(n)$