



---

# Spccifica di Java API: Un esempio

---



# Specifica di un tipo “primitivo”

---

- Le specifiche sono ovviamente utili **anche** per capire e usare correttamente i tipi di dato “primitivi” di Java
- Vedremo, come esempio, il caso dei vettori
  - Vector
  - array dinamici che possono crescere e ridursi
  - sono definiti nel package `java.util`



# Vector 1

---

```
public class Vector<E> {
    // OVERVIEW: un Vector è un array modificabile
    // di dimensione variabile i cui elementi sono
    // di tipo E: indici tra 0 e size - 1
    // costruttore
    public Vector( )
        // @effects: inizializza this a vuoto
    // metodi
    public void add(E x)
        // @modifies this
        // @effects: aggiunge una nuova posizione a this
        // inserendovi x di tipo E
    public int size( )
        // @effects: ritorna il numero di elementi di this
    ...
}
```



# Vector 2

---

```
public E get(int n) throws IndexOutOfBoundsException
    // @throws se  $n < 0$  o  $n \geq this.size$  solleva
    // IndexOutOfBoundsException,
    // @returns: l'oggetto in posizione  $n$  in this
public void set(int n, E x) throws
    IndexOutOfBoundsException
    // @modifies this
    // @throws se  $n < 0$  o  $n \geq this.size$  solleva
    // IndexOutOfBoundsException
    // @effects modifica  $this[n]$  inserendo l'oggetto  $x$ 
```



# Vector 3

---

```
public void remove(int n) throws IndexOutOfBoundsException
    // @motifies this
    // @throws se n<0 o n>= this.size solleva
    // IndexOutOfBoundsException,
    // effects elimina l'oggetto this[n] &&
    // Sposta tutti gli elementi successivi a sinistra
    // sottraendo uno dai loro indici) &&
    // diminuisce di uno il valore di this.size
```



# Vector: analisi

---

```
public class Vector {  
    // OVERVIEW: un Vector è un array modificabile  
    // di dimensione variabile i cui elementi sono  
    // di tipo E: indici tra 0 e size - 1  
    ...  
}
```

- Gli oggetti della classe sono descritti nella specifica in termini di concetti noti: gli array
- Gli stessi concetti sono anche usati nella specifica dei metodi
  - indice, elemento identificato dall'indice
- Il tipo è modificabile come l'array
- Notare che gli elementi sono di tipo E



# Vector: analisi

---

```
public class Vector {  
    // OVERVIEW: un Vector è un array modificabile  
    // di dimensione variabile i cui elementi sono  
    // di tipo E: indici tra 0 e size - 1  
    // costruttore  
    public Vector( )  
        // EFFECTS: inizializza this a vuoto  
    ...  
}
```

- Un solo costruttore (senza parametri)
  - inizializza this (l'oggetto nuovo) a un "array" vuoto



# Vector: analisi

---

```
public void add(E x)
```

```
public void set(int n, E x) throws  
    IndexOutOfBoundsException
```

```
public void remove (int n) throws  
    IndexOutOfBoundsException
```

## Sono modificatori

- modificano lo stato del proprio oggetto
- set e remove possono sollevare un'eccezione unchecked





# Vector: analisi

---

```
public int size( )
```

```
public E get(int n) throws IndexOutOfBoundsException
```

```
public E lastElement()
```

- Sono osservatori
  - non modificano lo stato del proprio oggetto
  - get può sollevare un'eccezione primitiva unchecked



# Aspetti metodologici

---

- Per prima cosa si definisce la specifica
  - “scheletro” formato da header, overview, pre- e post-condizioni di tutti i metodi
  - mancano la rappresentazione degli oggetti e il codice dei corpi dei metodi...
    - ✓ che possono essere sviluppati in un momento successivo e indipendentemente dallo sviluppo dei “moduli” che usano il nuovo tipo di dato
    - ✓ ed è molto importante riuscire a differire le scelte relative alla rappresentazione



# Aspetti metodologici

---

- Se aggiungiamo il codice dei metodi ben tipati alla specifica dei metodi
  - la specifica può essere compilata
  - possono essere compilate implementazioni di moduli che la utilizzano (errori rilevati subito dall'analisi statica)
  - possono essere progettati i test di analisi
  - esempio: Junit è basato su questa idea



# E ora?

---

- Quale è la proprietà significativa della classe Vector?
  - ***La dimensione effettiva di un oggetto di tipo Vector è inferiore o uguale alla sua capacità.***
- Ragionando sulla specifica (solamente) si dimostra che la proprietà è preservata da tutti i metodi della classe
  - I metodi non finali possono essere riscritti quindi è essenziale ragionare solamente in termini della specifica e non delle strutture di implementazione
- Maggiori dettagli:
  - A case study in class library verification: Java's vector class (*International Journal on Software Tools for Technology Transfer*, Springer 2001)
  - Marieke Huisman, Bart Jacobs, Joachim van den Berg