

Esercitazione n° 2

Obiettivi:

- Utilizzo di **costruttori**
- Utilizzo della classe `java.lang.reflect.Array`
- Semplici **algoritmi di ordinamento**
 - Naive sort
 - Bubble sort, Merge sort, ... (???)
- **Ricerca binaria** su un array ordinato
- Dal punto di vista operativo, sviluppo di un **progetto** organizzato **su più classi e file (package)**

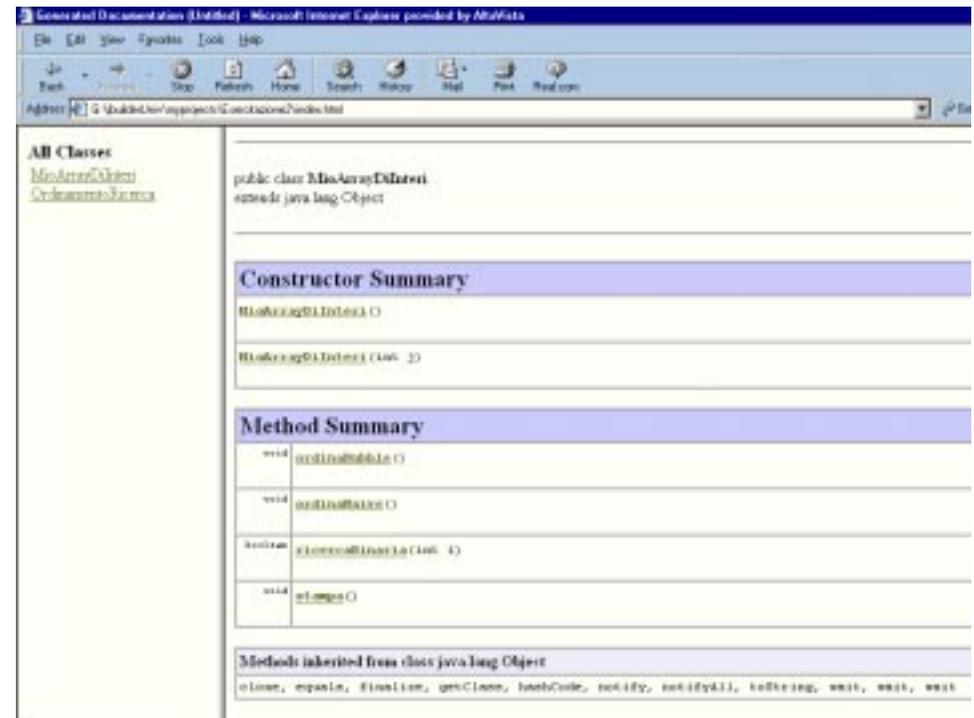
Le classi OrdinamentoRicerca e MioArrayDiInteri (1)

- Progetto strutturato su **due classi**:
 - `OrdinamentoRicerca` è la classe principale dell'applicazione

```
public static void main(String[] args) {
    MioArrayDiInteri a1 = new MioArrayDiInteri(10);
    MioArrayDiInteri a2 = new MioArrayDiInteri();
    a1.stampa();
    a2.stampa();
    a2.ordinaNaive();
    a2.stampa();
    boolean esiste = a2.ricercaBinaria(x);
}
```

Le classi OrdinamentoRicerca e MioArrayDiInteri (2)

- `MioArrayDiInteri` include:
 - **definizione/inizializzazione** degli oggetti (differenti costruttori);
 - implementa i metodi di
 - **stampa**
 - **ordinamento**
 - **ricerca**



La classe MioArrayDiInteri

- Ogni oggetto della classe MioArrayDiInteri deve poter memorizzare ELEM elementi che sono **numeri interi**
- Un oggetto della classe può essere:
 - inizializzato ad avere tutti gli elementi con **valore uguale** ad un parametro - **costruttore1**
 - inizializzato a contenere **valori casuali** nell'intervallo [0, 99] - **costruttore2**

MioArrayDiInteri: i costruttori

```
public MioArrayDiInteri() { // Costruttore1
    numbers = new int[ELEM];
    for (int i=0; i<ELEM; i++) // Assegnamento di valori
        // casuali in [0,99]
        numbers[i]=(int) (ELEM * Math.random());
}

public MioArrayDiInteri(int j) { // Costruttore2
    numbers = new int[ELEM];
    for (int i=0; i<ELEM; i++) // Assegnamento di j
        // a tutti gli elementi
        numbers[i] = j;
}
```

MioArrayDiInteri: stampa degli elementi

```
public void stampa() {

    System.out.println("I numeri interi contenuti
        nell'array sono:\n");

    for (int i=0; i<ELEM/10; i++)
        for (int j=0; j<ELEM/10; j++)
            System.out.print(numbers[i*10+j] + "\t");
}
```

Algoritmi di ordinamento: Naive sort

```
public void ordinaNaive() {
    int j, i, posmin, min;

    for (j=0; j < ELEM; j++) {
        posmin = j; min = numbers[j];
        for (i = j+1; i < ELEM; i++)
            if (numbers[i] < min) {
                min = numbers[i]; posmin = i; }
        if (posmin != j) { // scambio valori
            min = numbers[posmin];
            numbers[posmin] = numbers[j];
            numbers[j] = min; }
    }
}
```


Da fare in laboratorio...

- Si scriva una procedura di **ricerca binaria**
- Si scriva una procedura di **ricerca binaria ricorsiva**
Algoritmo:
 - Se l'elemento cercato i è uguale a $V[\text{med}]$ si termina (caso base)
 - Altrimenti se $i < V[\text{med}]$ si effettua una ricerca binaria sulla prima metà del vettore
 - Altrimenti se $i > V[\text{med}]$ si effettua una ricerca binaria sulla seconda metà del vettore
- (facoltativo) Si scriva un metodo alternativo per l'**ordinamento** dell'array (***bubble sort, merge sort, ...***)