Fondamenti di informatica T-1 (A – K)
Esercitazione 9: array

AA 2018/2019

Tutor

Lorenzo Rosa

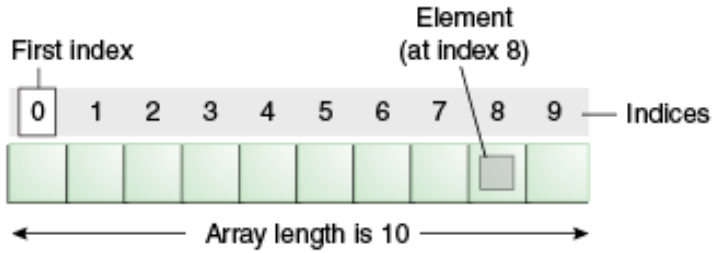
lorenzo.rosa@unibo.it

Esercitazione 9

- Introduzione al calcolatore e Java
- Linguaggio Java, basi e controllo del flusso
- Eclipse ed esercizi di consolidamento
- **Stringhe ed array**
- Metodi, classi, oggetti
- Ereditarietà e polimorfismo
- Collezioni Java e interfacce
- **Esercizi d'esame**

Array

- Gli array sono dei contenitori che ospitano un numero prestabilito di oggetti dello stesso tipo.



- Il codice che segue, ad esempio, crea un array capace di contenere al più 10 numeri interi:

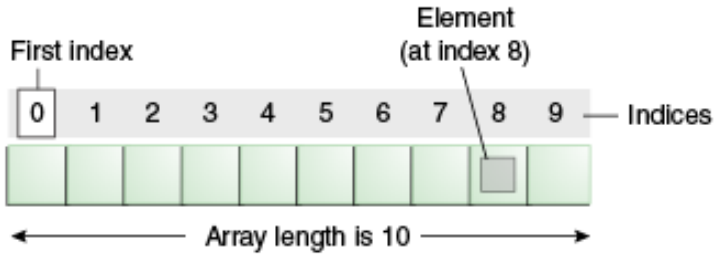
```
int[] arrayInteri;

arrayInteri = new int[10];
```

Array: dimensione fisica

- La **dimensione fisica** dell'array indica quanti elementi al massimo esso può contenere. Essa è definita all'atto della inizializzazione (con la new) e memorizzata nell'attributo `length`.

```
int dimFisica = arrayInteri.length; //10
```

- Una volta creato, non se ne può estendere la dimensione.



Array: dimensione logica

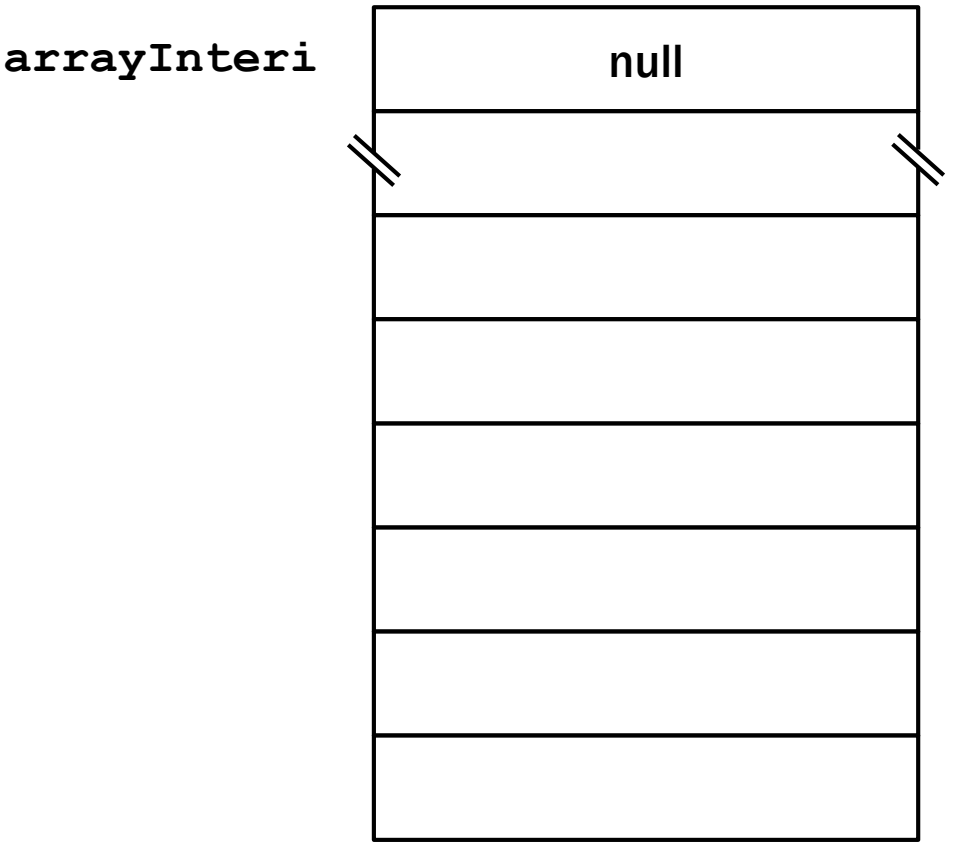
- La **dimensione logica** dell'array è il numero di elementi che abbiamo inserito noi nell'array.

```
int[] arrayInteri;
arrayInteri = new int[10];
arrayInteri[0] = 1;
```

- E gli altri elementi? Java inizializza tutti gli elementi al valore di default:
  - valori numerici → 0
  - valori booleani → false
  - tipi definiti da classi → *null*

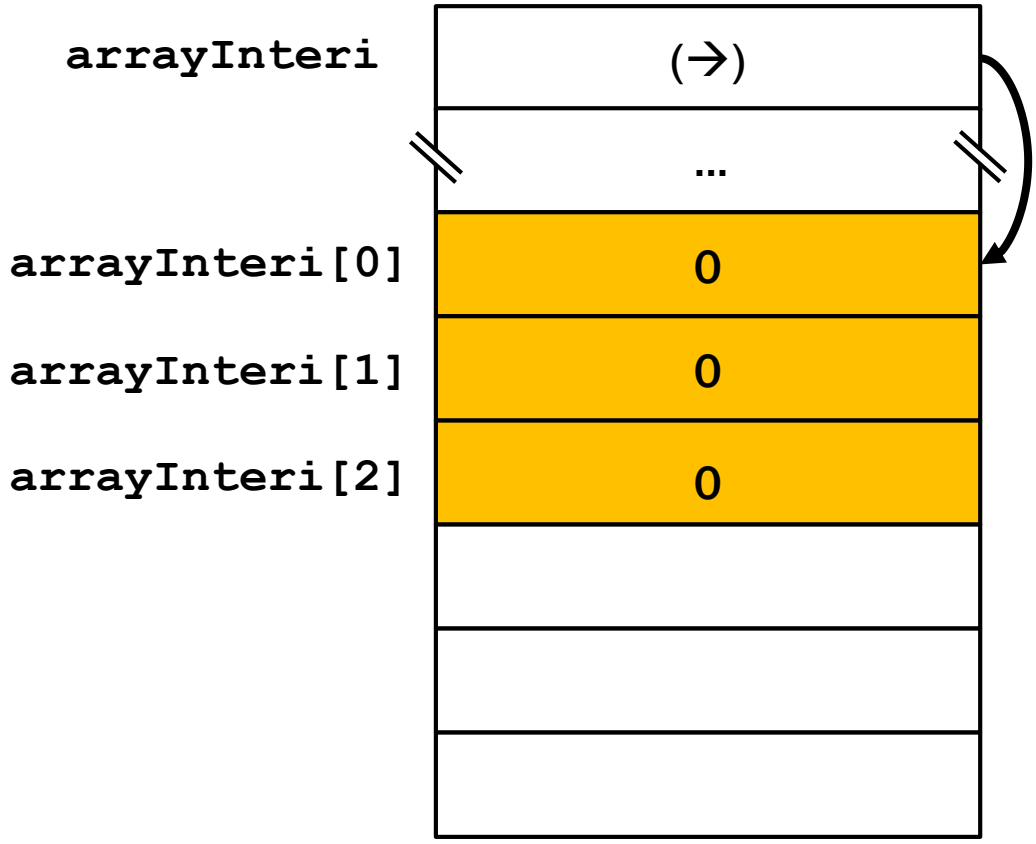
Array

```
int[] arrayInteri;
```



Array

```
int[] arrayInteri;

arrayInteri = new int[3];
```



Dimensione fisica = 3

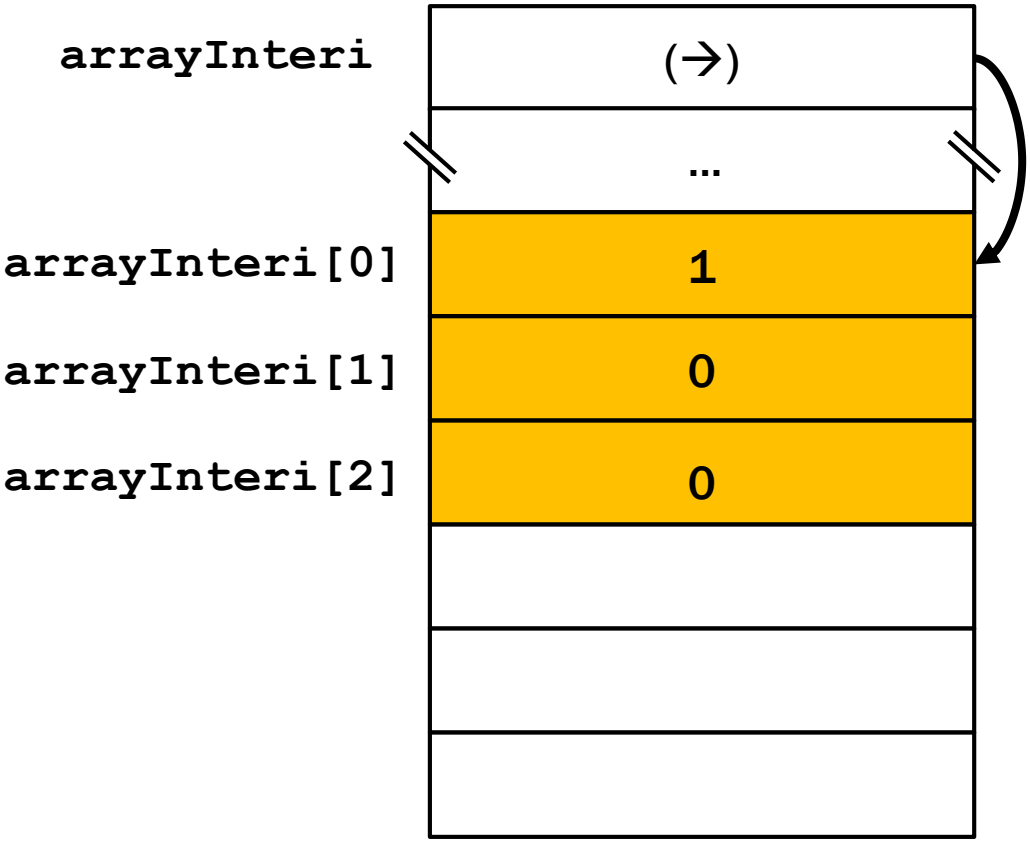
Dimensione logica = 0

Array

```
int[] arrayInteri;

arrayInteri = new int[3];

arrayInteri[0] = 1;
```

Dimensione fisica = 3

Dimensione logica = 1



Array di oggetti

- Fino ad ora abbiamo solo utilizzato array di *tipi primitivi.*

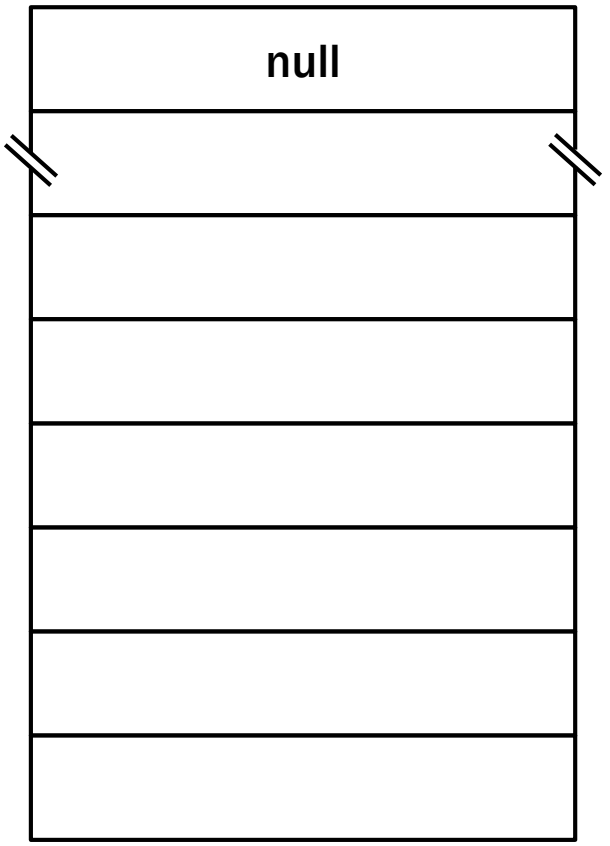
  Poichè **una classe definisce un tipo**, possiamo costruire array di oggetti del tipo della classe che abbiamo definito:

```
String[] stringhe = new String[3];
Libro[] libri = new Libro[3];
```

- Gli array di oggetti vengono inizializzati a *null.*

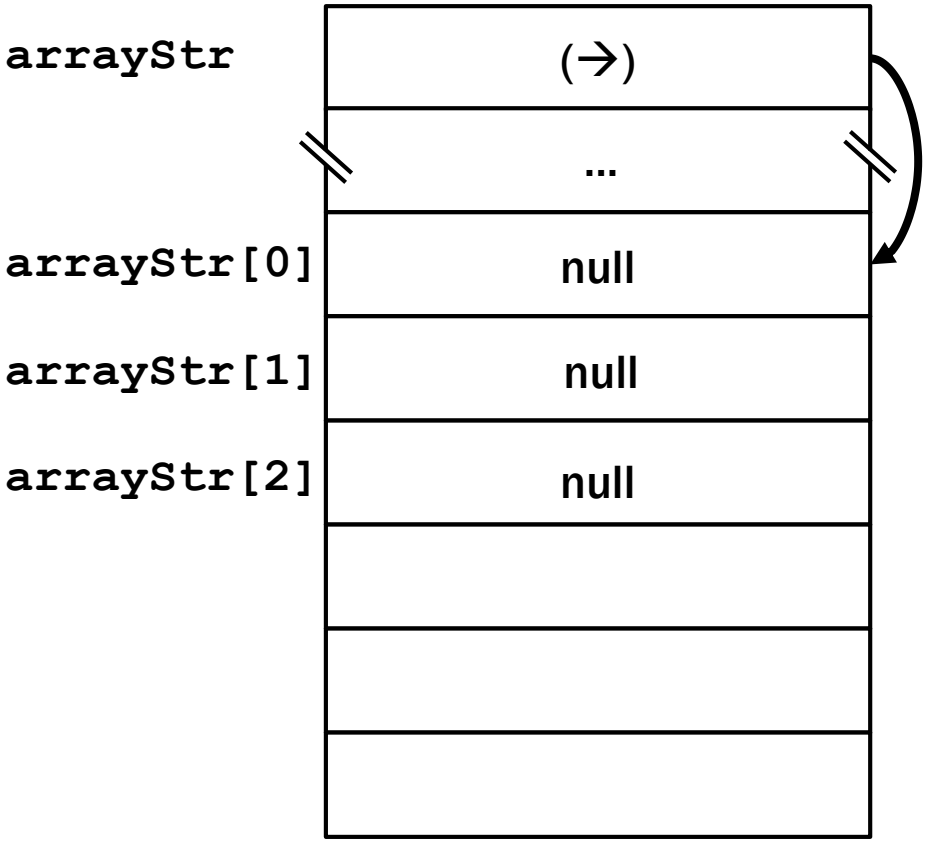
Array di stringhe

```
Stringhe[] arrayStr;
```



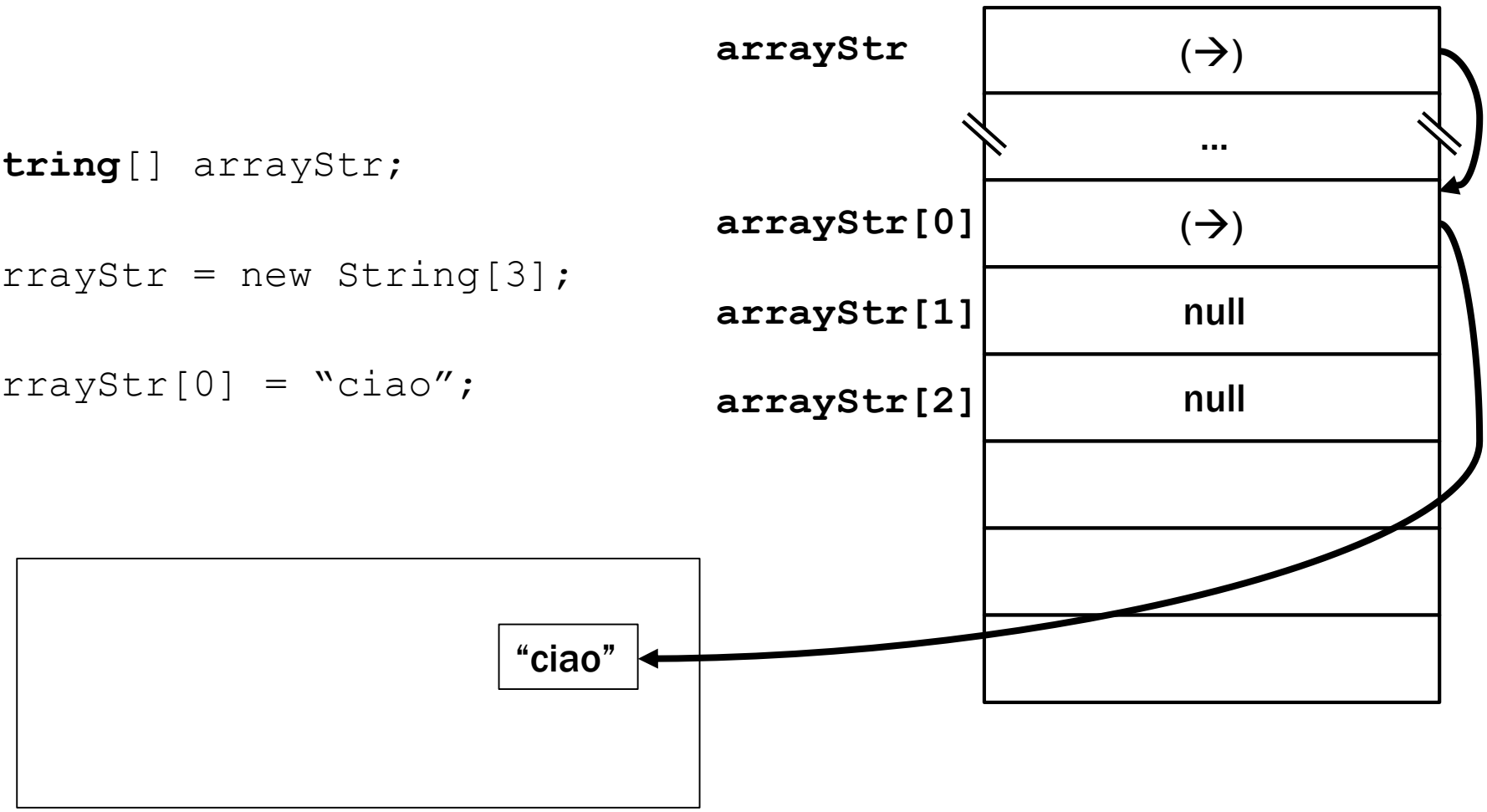
Array di stringhe

```
String[] arrayStr;

arrayStr = new String[3];
```



Array di stringhe

```
String[] arrayStr;

arrayStr = new String[3];

arrayStr[0] = "ciao";
```



Array di oggetti

- Gli oggetti dell'array sono inizializzati a *null.* Per poter effettivamente utilizzare l'array, dobbiamo **costruire un oggetto per ogni cella dell'array**.

```
Contatore[] contatori = new Contatore[3];

for(int i = 0; i<contatori.length; i++)
    contatori[i] = new Contatore(0);

//Solo successivamente si possono invocare i
metodi sugli oggetti (prima non esistevano...)

contatori[1].incrementa();
```

Libro (uguale a esercitazione 8)

Il rettore ha deciso di informatizzare la gestione dei libri dell'ateneo. Per ogni libro occorre memorizzare:

1. l'autore
2. il titolo
3. la casa editrice
4. l'anno di pubblicazione.

Si scriva una classe `Libro` che:

1. Possieda un opportuno **costruttore** con parametri.
2. Presenti opportuni **metodi** che permettano di **accedere** alle variabili di istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca la descrizione del libro (`String`).
4. Presenti il metodo `uguale` che, dato un `Libro` in ingresso, restituisca un booleano: `true` se tutti i campi del Libro in ingresso sono uguali ai propri, `false` altrimenti.

Cominciamo a ragionare...

La classe Biblioteca è un po' troppo rigida:

- perché solo tre libri?
- perché tutte le biblioteche lo stesso numero di libri?

Utilizzando gli array come contenitore, vogliamo riscrivere la classe Biblioteca per superare questi due problemi.

Esercizio 1/2 - Biblioteca

Si scriva una classe `Biblioteca` che memorizzi le informazioni relative ai libri contenuti all'interno della biblioteca. Occorre memorizzare il **nome** della biblioteca, l'**indirizzo**, un **codice** identificativo univoco (intero). La classe `Biblioteca` deve inoltre:

1. presentare un opportuno costruttore **che prenda come argomento anche il numero massimo di libri che tale biblioteca può contenere.**
2. presentare opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto (non si considerino i libri)
3. possedere un metodo `aggiungi` che, dato un oggetto `Libro`, **lo inserisca nella biblioteca**, purché vi sia posto, e restituisca *true;* altrimenti, restituisca *false.*
4. presentare un metodo `cerca` che, dato il nome di un autore, restituisca **tutti i libri** di tale autore all'interno della biblioteca (*null* se non esistono).
5. possedere un metodo `cancella` che, dato il nome di un autore, rimuova dalla biblioteca tutti i libri di tale autore (se presenti).
6. possedere il metodo `toString` che restituisca una stringa che fornisca una descrizione della biblioteca, compreso il numero di libri effettivamente presenti.

Biblioteca: primo cambiamento

Si scriva una classe `Biblioteca` che memorizzi le informazioni relative ai libri contenuti all'interno della biblioteca. Occorre memorizzare il **nome** della biblioteca, l'**indirizzo**, un **codice** identificativo univoco (intero).

La classe `Biblioteca` deve inoltre:

1. presentare un opportuno costruttore **che prenda come argomento anche il numero massimo di libri che tale biblioteca può contenere**.

Biblioteca: primo cambiamento

```
public class Biblioteca {
        String nome;
        String indirizzo;
        int codice;
        Libro[] libri;
        public Biblioteca(String nome, String indirizzo,
                                        int codice, int maxLibri) {
                this.nome = nome;
                this.indirizzo = indirizzo;
                this.codice = codice;
                this.libri = new Libro[maxLibri];
        }
...
```

Biblioteca: secondo cambiamento

3. possedere un metodo `aggiungi` che, dato un oggetto `Libro`, **lo inserisca nella biblioteca**, purché vi sia posto, e restituisca *true;* altrimenti, restituisca *false.*

Biblioteca: terzo cambiamento

4. presentare un metodo cerca che, dato il nome di un autore, restituisca **tutti i libri** di tale autore all'interno della biblioteca (null se non esistono).

Problemi:

1. Come restituiamo una collezione di `Libro`?
2. Di che dimensione deve essere la collezione da restituire?
   a. Potrei restituire una collezione di dimensione fissa, ma chi la riceve non sa quanti elementi effettivamente ci sono.
   b. Oppure, prima conto gli elementi da inserire, poi creo una collezione della giusta dimensione! → meglio così!

Biblioteca: terzo cambiamento

```
public Libro[] cerca(String autore) {
        int count = 0;
        for(int i = 0; i < libri.length; i++)
                if(libri[i] != null && libri[i].getAutore().equals(autore))
                        count++;
        if(count == 0)
                return null;
        Libro[] res = new Libro[count];
        int j = 0; //Indice per l'array "res"
        for(int i = 0; i < libri.length; i++)
                if(libri[i] != null && libri[i].getAutore().equals(autore)) {
                        res[j] = libri[i];
                        j++;
        }
        return res;
}
```

Biblioteca: quarto e quinto cambiamento

5. possedere un metodo `cancella` che, dato il nome di un autore, rimuova dalla biblioteca tutti i libri di tale autore (se presenti).

6. possedere il metodo `toString` che restituisca una stringa che fornisca una descrizione della biblioteca, compreso il numero di libri effettivamente presenti.

Biblioteca: quarto e quinto cambiamento

```
public void cancella(String autore) {
        for(int i = 0; i < libri.length; i++)
                if(libri[i] != null && libri[i].getAutore().equals(autore))
                        libri[i] = null;
}

public String toString() {
        String libridisponibili = "";
        for(int i = 0; i < libri.length; i++)
                if(libri[i] != null)
                        libridisponibili += libri[i].toString() + " ";
        return "Biblioteca " + nome + "(" + codice + ") - "
                        + indirizzo + " con libri " + libridisponibili;
}
```

Esercizio 2/3 - Applicazione

Si scriva un'applicazione per l'ateneo, che:

1. Crei due oggetti `Biblioteca` (nome e altri dati a piacere);
2. Crei un oggetto `Libro`, lette da tastiera le informazioni necessarie.
3. Trovi la prima biblioteca che non contiene alcun libro dell'autore del libro di cui al punto 2.
4. Inserisca il libro di cui al punto 2 nella biblioteca di cui al punto 3 (se tale biblioteca esiste).
5. **Letto da tastiera il nome di un autore, provveda a stampare a video tutti i libri di tale autore presenti in entrambe le biblioteche.**

Esercizio 2/3 - Applicazione

Si scriva un'applicazione per l'ateneo, che:

1. Crei due oggetti `Biblioteca` (nome e altri dati a piacere);

   Perché funzioni, bisogna aggiungere ai costruttori il parametro che abbiamo inserito, che indica il massimo numero di libri che la biblioteca possiede.

Esercizio 2/3 – punto numero 4

```
System.out.print("Inserire autore: ");
autore = tastiera.nextLine();

Libro[] res = biblio1.cerca(autore);
if(res != null) {
    System.out.println("Biblioteca 1: ");
    for(int i = 0; i < res.length; i++)
    System.out.println(res[i]);
}

res = biblio2.cerca(autore);
if(res != null) {
    System.out.println("Biblioteca 2: ");
    for(int i = 0; i < res.length; i++)
       System.out.println(res[i]);
}
```