

Singleton pattern

È un design pattern

Vogliamo fare in modo che di una classe possa esistere un'unica istanza

```
public class MiaClasse {
    private static MiaClasse m;
    private MiaClasse() {}

    public static MiaClasse getInstance() {
        if (m == null)
            m = new MiaClasse();
        return m;
    }
}
```

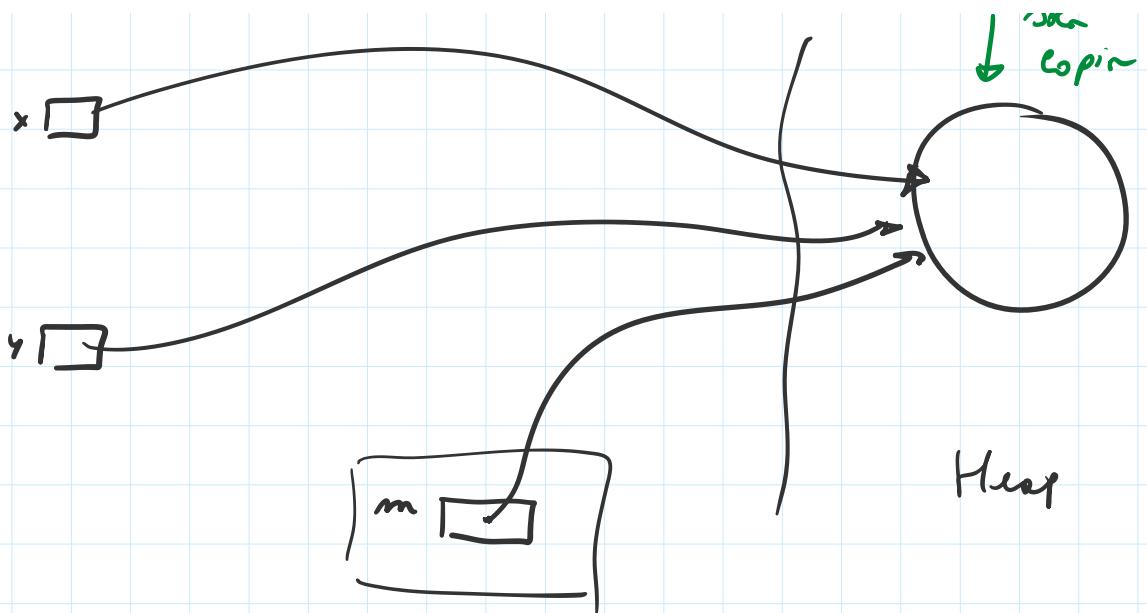
non può essere creata da fuori
esiste a prescindere dall'esistenza di oggetti di tipo MiaClasse
non può essere chiamata da fuori MiaClasse

In un'altra classe :

```
:
MiaClasse x = MiaClasse.getInstance();
:
```

```
MiaClasse y = MiaClasse.getInstance();
```

una sola copia



MiaClasse z = new MiaClasse(); // errore

Il costruttore di default è public

Class e file

Regole generali: ogni classe è definita in un proprio file e il file ha lo stesso nome della classe

Cosa succede quando compiliamo con java c una classe che ne usa altre:

```
public class MiaClasse { } → MiaClasse.java
```

```

public class Principale {
    ...
    MiaClasse x = new MiaClasse();
    ...
}

```

] → Principale.java

Compiliamo Principale

\$ javac Principale.java

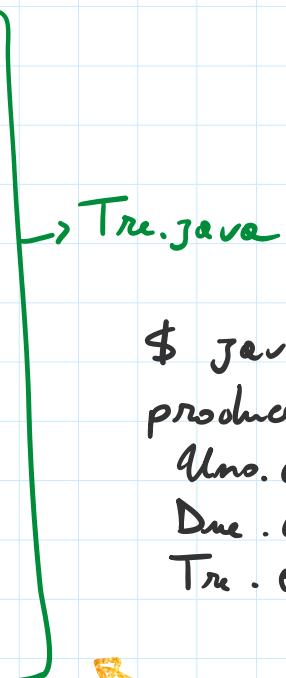
- 1) il compilatore si accorge che Principale usa MiaClasse
- 2) il compilatore cerca MiaClasse.class
 - a) se MiaClasse.class non c'è javac compila MiaClasse.java
 - b) se MiaClasse.class è presente javac controlla la data di file MiaClasse.class e MiaClasse.java, nel caso in cui il file sorgente sia più recente lo ricompila.

Regole: all'interno di un unico file è possibile memorizzare la definizione di più classi; al più una classe può essere indicata come "public"

Regola: se c'è una classe "public" allora il file sorgente dove avere lo stesso nome della classe

Esempio

```
class Uno {  
    :  
}  
class Due {  
    :  
}  
public class Tre {  
    :  
}
```



```
$ javac Tre.java  
produce 3 file .class:  
Uno.class  
Due.class  
Tre.class
```

Quando scegliere questo approccio?

solo quando le classi Uno e Due
(quelle non public) vengono usate
solo ed esclusivamente da Tre

Perché?

Supponiamo di avere una classe

Quattro che usa Uno

cercò di compilare Quattro

```
$ javac Quattro.java
```

javac cerca Uno.class

se non c'è, cerca Uno.java

se non c'è, cerca `Uno.java`
che però non esiste (Uno è definito in `Tre.java`)
la compilazione si ferma con una segnalazione
di errore.

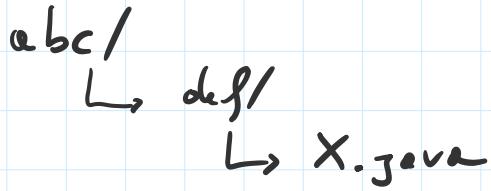
Se invece `Tre.java` è stato compilato in
precedenza allora `Uno.class` viene trovato.

Class, file e filesystem (directory)

Se definisco una classe che appartiene a un
package il file sorgente devo metterlo
in una cartella che ha lo stesso nome
del package.

Supponiamo di definire `X.java`
e X appartiene al package `abc.def`

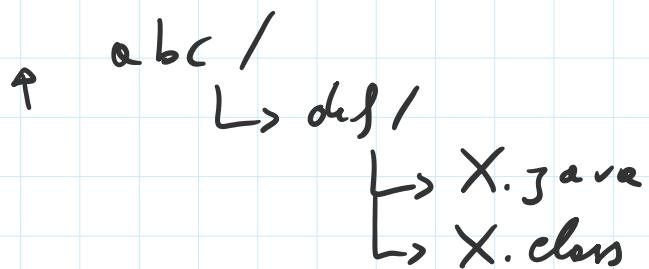
`abc/`



per compilare mi dovrò "posizionare" nella cartella che contiene la cartella "abc"

\$ javac abc/def/X.java

produce X.class nella cartella def



Se X contiene un main, per mandarlo in esecuzione mi dovrò "posizionare" nella cartella che contiene la cartella "abc" e dare il comando

\$ java abc.def.X

le JVM cerca il file X.class nella cartella "def" contenuta nella cartella "abc"

Regola: posizionarsi, se per compilazione che per esecuzione, nella cartella che contiene la radice del package

a b c /
↳ abc/
Qui ↳ abc/
↳ X.java
↳ X.class

c1 /
↳ c2 /
↳ c3 /
↳ Y.java
↳ Y.class

Scegliere che appartenne al package c1.c2.c3
una X (che appartiene a abc.def)

con il comando

\$ javac c1/c2/c3/Y.java

vieni prodotto.

Y.class

X.class

classpath

classpath: un insieme di cartelle* a partire dalle quali il compilatore e la jvm cercano le classi che servono loro

* possono essere non solo cartelle ma anche file .jar (archivi)



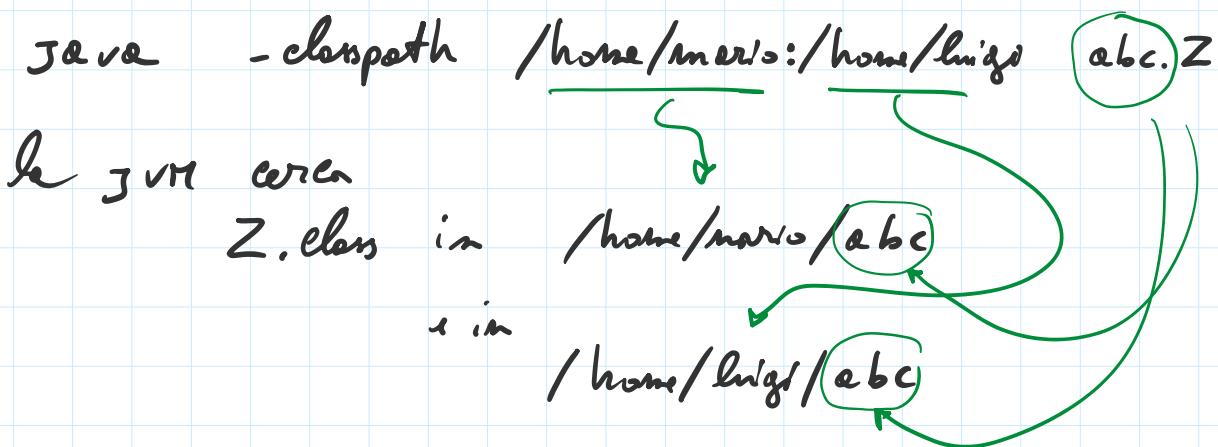
Se il classpath non viene specificato allora

Se il classpath non viene specificato allora
è uguale a ":"

Possiamo specificarlo con le opzioni:

-cp oppure -classpath

i vari punti di ricerca possono essere
separati con : in Unix e ; in Win



Lo stesso vale per il compilatore

Java -classpath .:/ver/lib abc.Z

Le classi del sistema vengono trovate
automaticamente

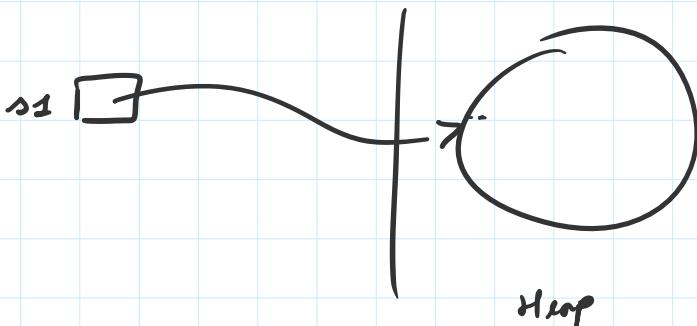
Gestione della memoria

Tutti gli oggetti vengono allocati nello heap

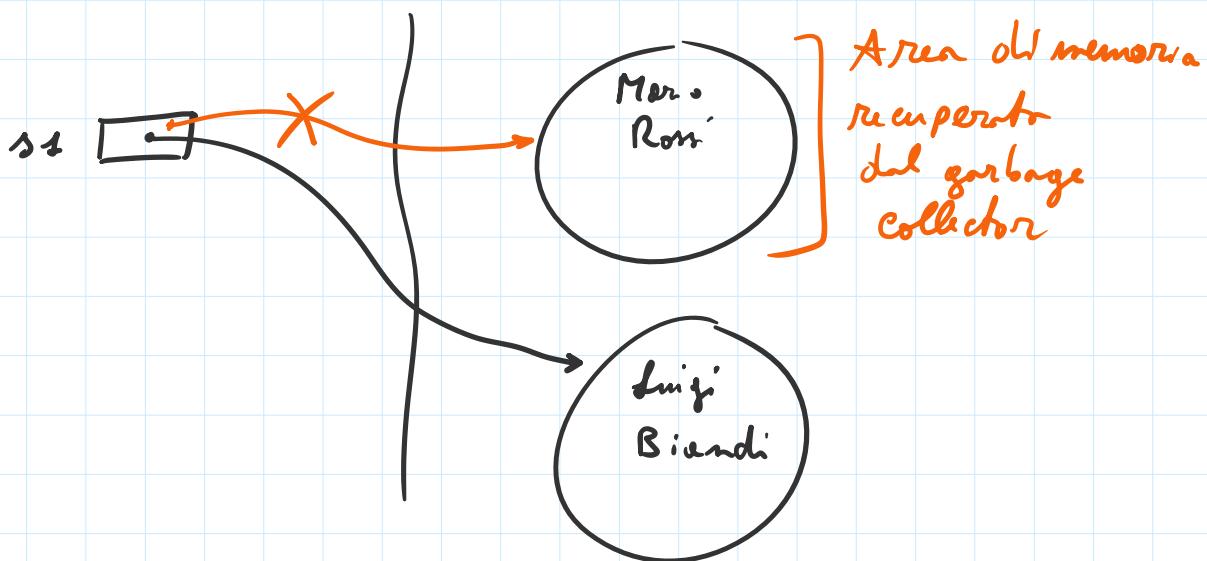
La deallocazione degli oggetti non è responsabilità
del programmatore

La Java dispone di un sistema per il recupero della memoria non più utilizzata:
il garbage collector

Studente s1 = new Studente("Mario", "Ross");



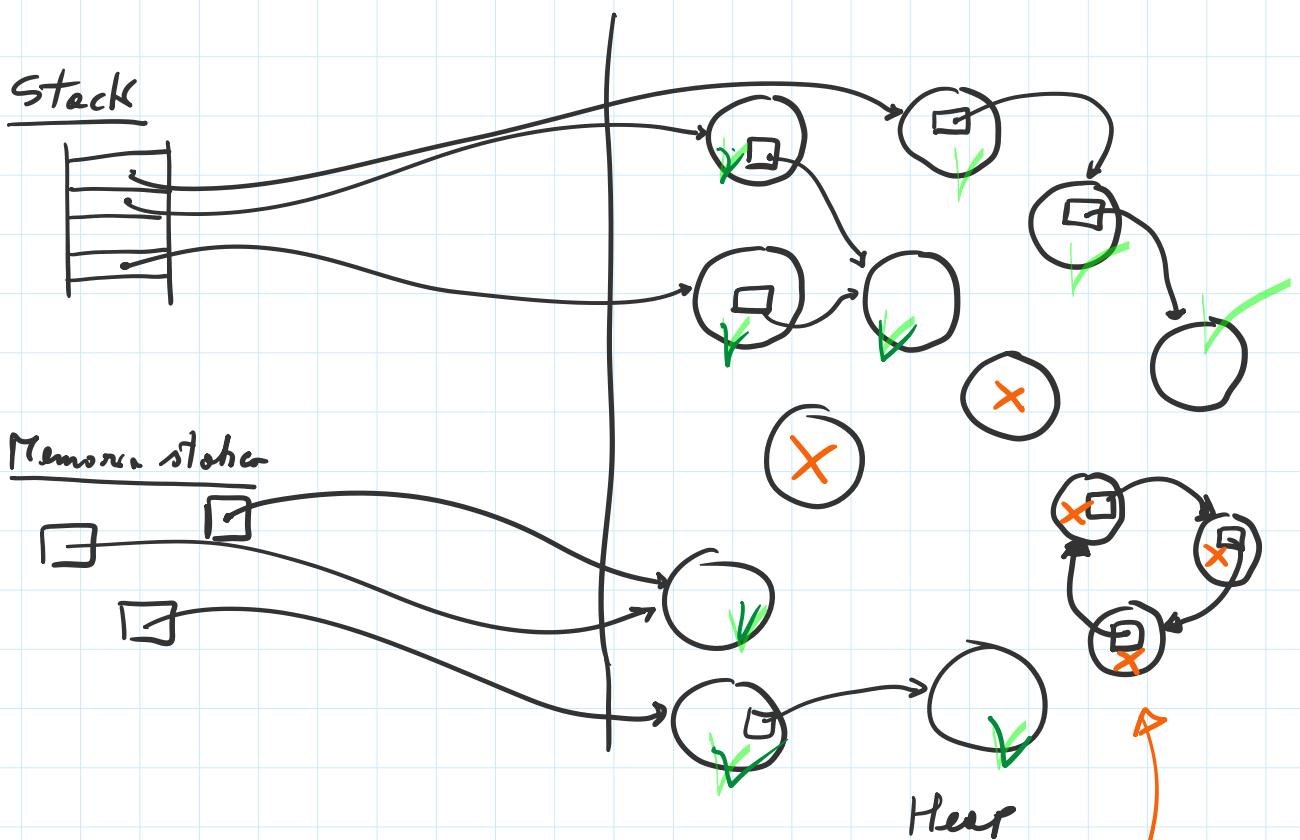
s1 = new Studente("Luigi", "Biagini");



Il garbage collector recupera la memoria quando vuole lui

Potrei mettere in esecuzione il garbage collector con il metodo `gc()` della classe `System`

`System.gc();`

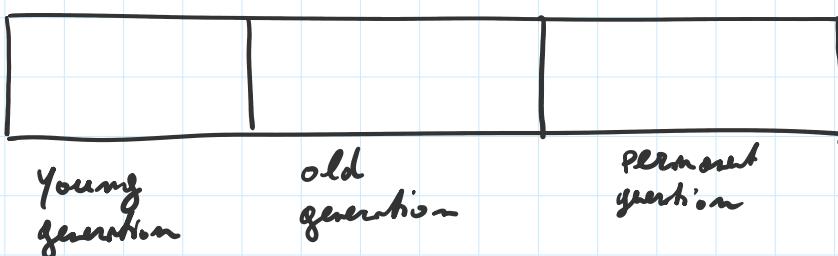


il g.c.
è in grado di
catturare da non
sono raggiungibili

Gli oggetti possono essere divisi in due categorie:

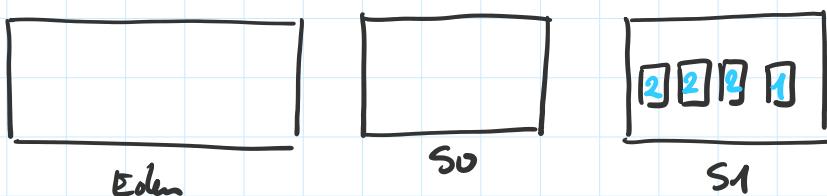
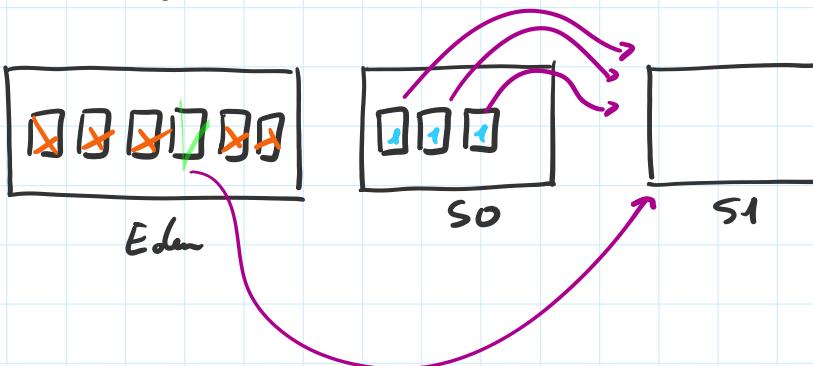
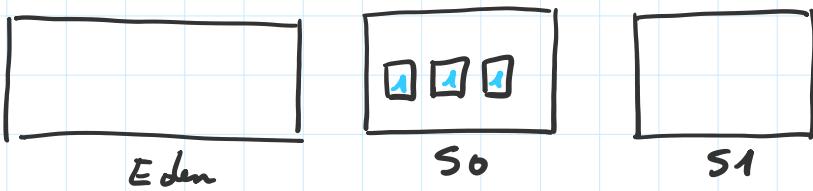
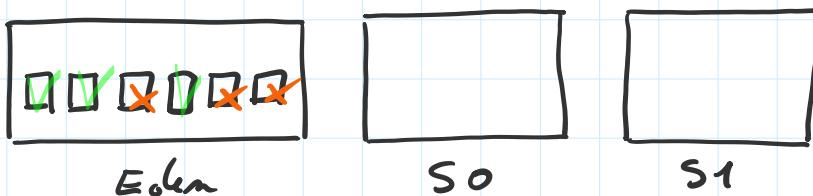
- quelli che vivono poco
- quelli che vivono molto

G.C. generazionale



Young generation

S: Survivor

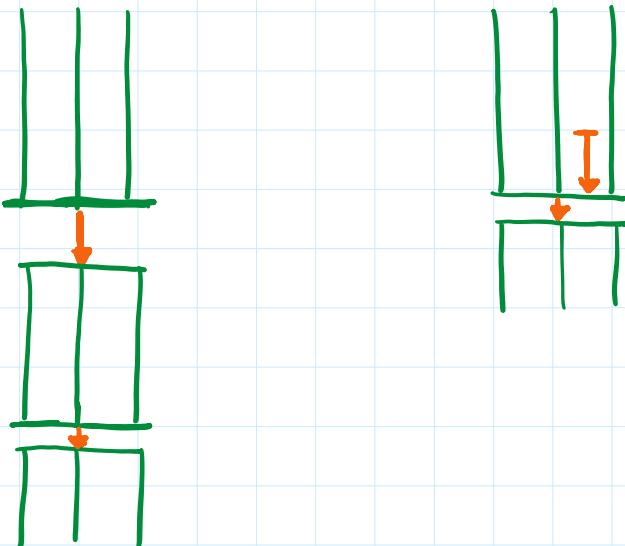


Quando il costruttore supera un certo valore l'oggetto viene spostato nella old generation

Minor garbage collection (solo sulla young generation)

Major garbage collection (quando la old generation è piena)

Molte effutte del G.C. sono del tipo "stop-the-world"



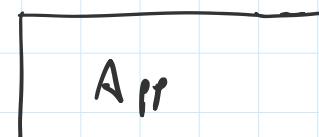
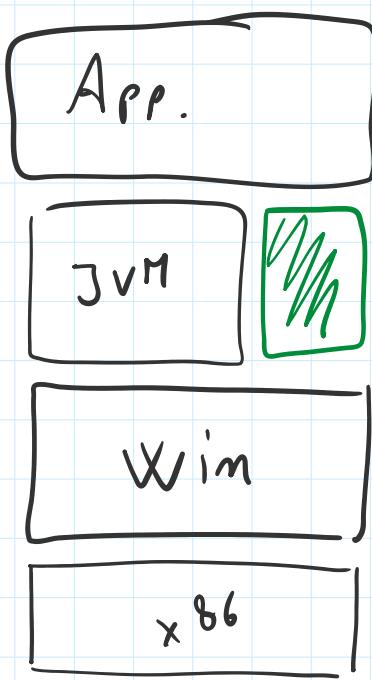
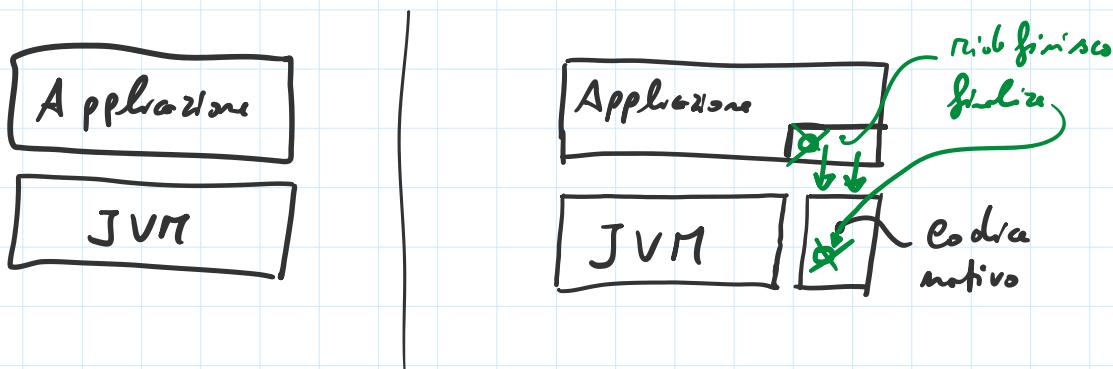
java -classpath /usr/lib/oracle.jar: ~

Per ricevere "notifiche" dal g.c. quando un oggetto sta per essere rimosso

ri definire il metodo

protected void finalize()

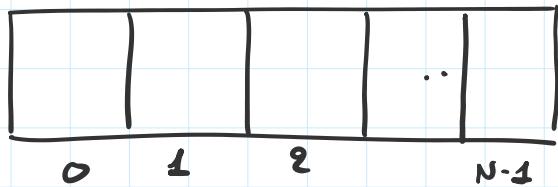
utile soprattutto se il codice Java è
collegato a codice non Java (codice nativo)



Array

Container che valori dello stesso tipo
elementi identificati da un indice

indici partono da zero



R con N dimensioni
nell'array

Dimensione obbligatoria al tempo di esecuzione

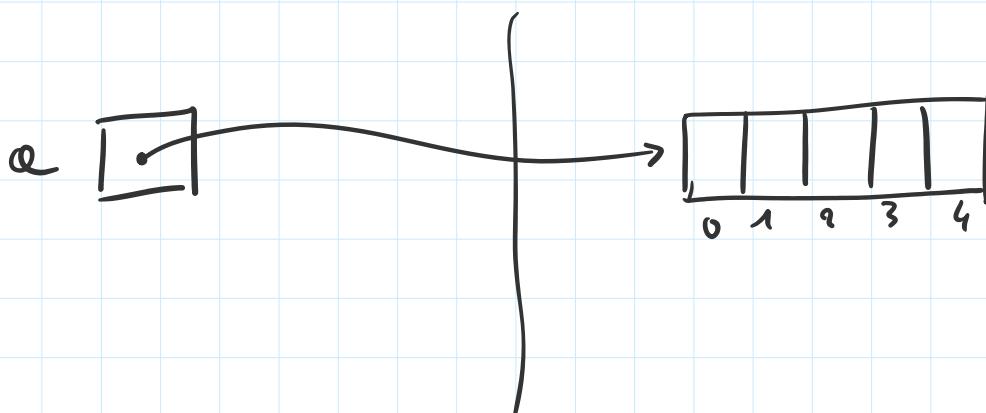
int [] a;

Ho creato un riferimento ad array di int
per ora non c'è nessun array

a = new int[5];

adesso ho creato un array di 5 int
memorizzato nello heap

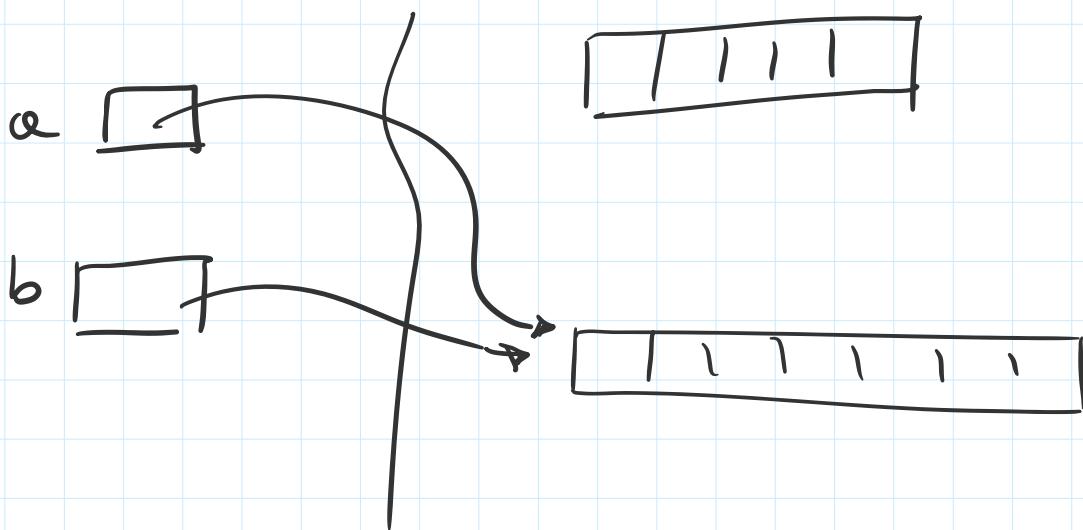
lo posso raggiungere con il riferimento a



`int [] b = new int [10];`



`a = b; //OK`

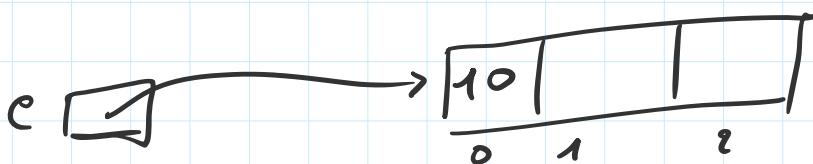


per accedere , selezione con indice

`int [] c = new int [3];`

`c[0] = 10;`

`int k = c[0];`



In generale

Typo[] nome = new Typo[dimensione];

variable members length che ne indica la lunghezza

int[] q1 = new int[5];

int l = q1.length; // length: lunghezza
// l vale 5

float[] q2 = new float[23];

l = q2.length; // l vale 23