

## Programmazione ad Oggetti: JAVA

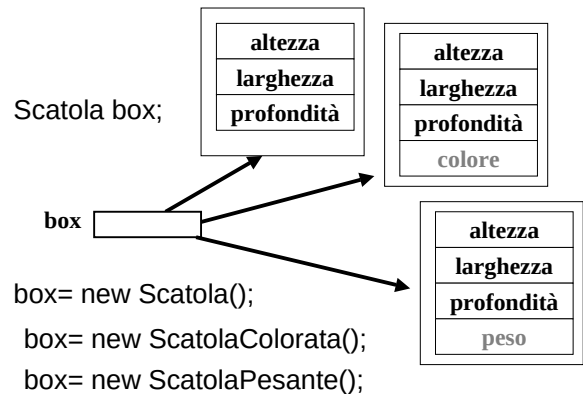
Esempi proposti nella  
lezione 26/5/06 e precedenti

Docente

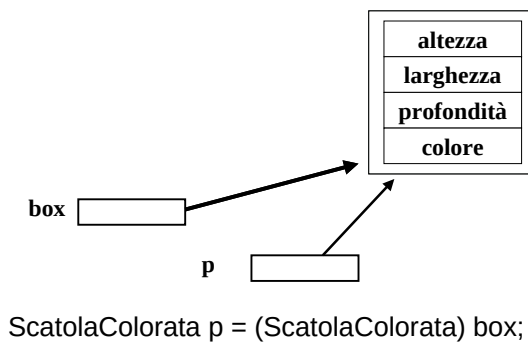
Ing. Mariateresa Celardo

[mariateresa.celardo@uniroma2.it](mailto:mariateresa.celardo@uniroma2.it)

### Ereditarietà e polimorfismo



### Ereditarietà e polimorfismo



### Ereditarietà e polimorfismo

Se costruisco un metodo che prende come  
parametro un riferimento a Scatola  
Tale metodo potrà essere applicato anche a  
ScatolaColorata e ScatolaPesante

```
void aumentaDimensioni(Scatola s, int x){
    s.altezza+=x;
    s.larghezza+=x;
    s.profondità+=x;
}
```

### Polimorfismo

Potrei ridefinire il metodo in una sottoclasse e  
specializzarlo:

```
void aumentaDimensioni(Scatola s, int x){
    super.aumentaDimensioni(s,x);
    if (s instanceof ScatolaColorata) {
        ScatolaColorata c = (ScatolaColorata) s;
        c.colore="bianco";
    }
}
```

### Documentazione

- Si possono inserire commenti particolari (commenti di documentazione) che vengono usati dal programma javadoc per produrre pagine HTML che descrivono la classe

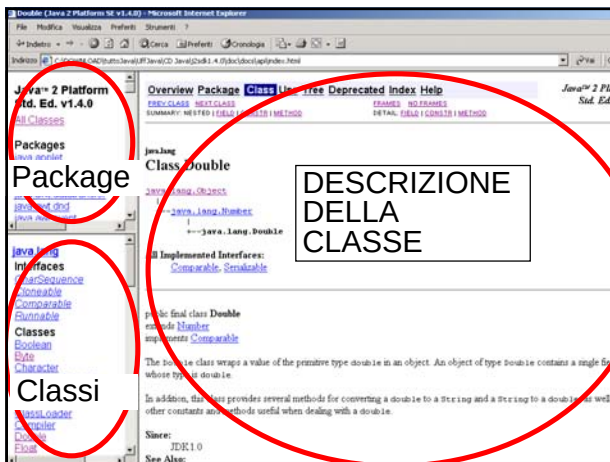
```
/**
 *
 */
```
- **@ param** <nome parametro> <spiegazione>
- **@ return** <spiegazione>

## Documentazione

- Per produrre la documentazione occorre digitare  
javadoc NomeClasse.java
- Nella cartella corrente verranno creati file html

```
C:\esercizi>javadoc InterfacciaConsole.java
Loading source file InterfacciaConsole.java...
Constructing Javadoc information...
Standard Doclet version 1.4.1
```

```
Generating constant-values.html...
Building tree for all the packages and classes...
Building index for all the packages and classes...
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating allclasses-noframe.html...
Generating index.html...
Generating packages.html...
Generating InterfacciaConsole.html...
Generating package-list...
Generating help-doc.html...
Generating stylesheet.css...
```

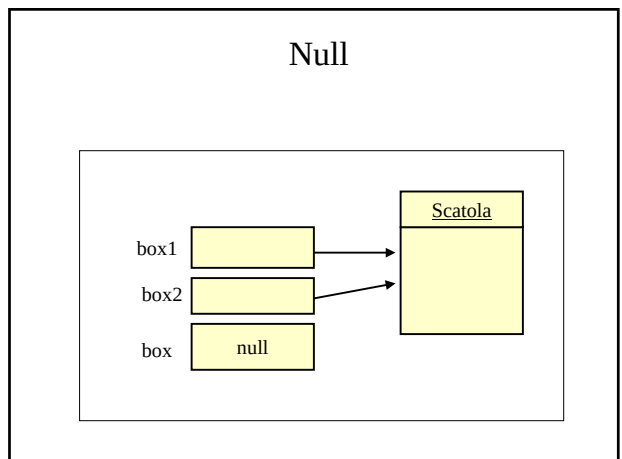
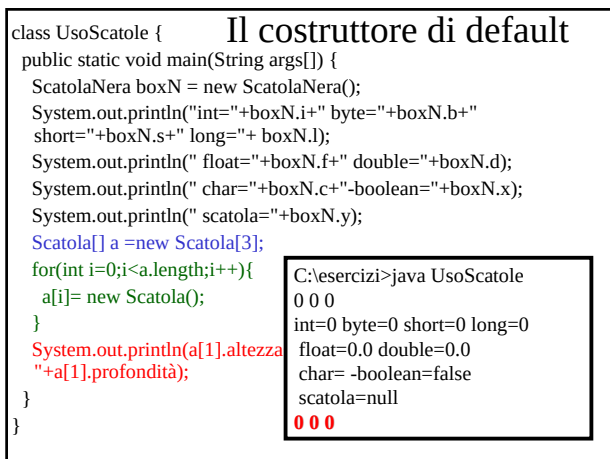
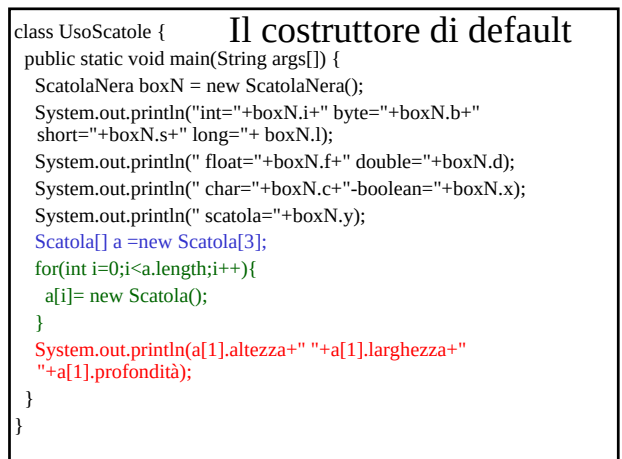
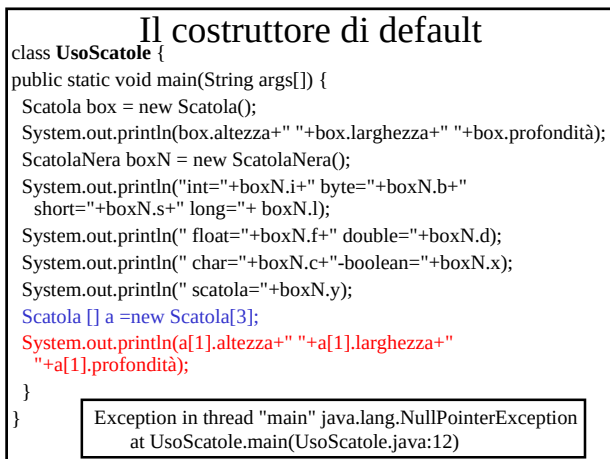
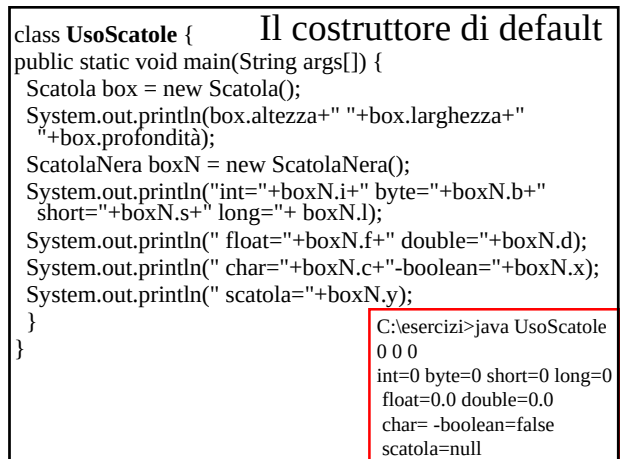
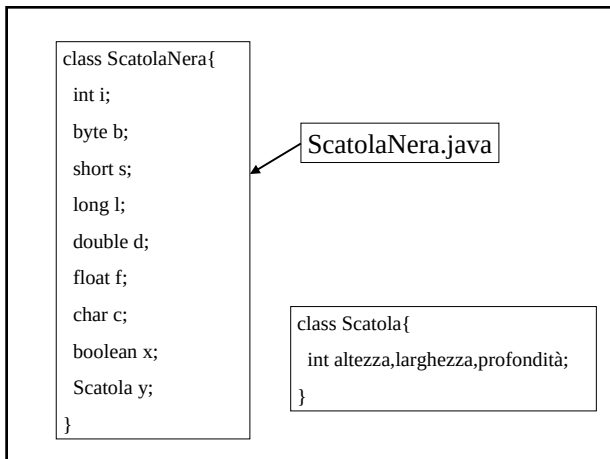


## Alcuni esempi e considerazioni

## Costruttore di default

## Esercizio sul costruttore di default

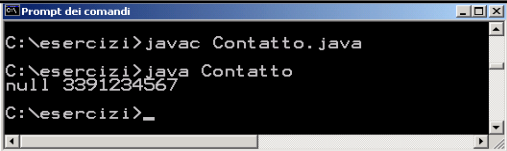
- Scrivere una classe ScatolaNera avente 9 attributi di cui:
  - 8 attributi sono di ciascun tipo primitivo
  - un attributo è una variabile di riferimento ad un oggetto scatola
- Scrivere una classe UsoScatole che:
  - istanzia un oggetto di tipo Scatola e stampa i suoi tre attributi,
  - istanzia un oggetto di tipo ScatolaNera e stampa i suoi 9 attributi
- Aggiungere alla classe UsoScatole le istruzioni per istanziare un array di tipo Scatola di 3 elementi e stampare tutti gli attributi del secondo elemento dell'array



## Costruttore e this

## Esempio

```
class Contatto {
    String nome;
    String numero;
    Contatto(String nome, String numero){
        nome= nome;
        this.numero=numero;
    }
    public static void main(String args[]){
        Contatto c= new Contatto("mario","3391234567");
        System.out.println(c.nome+" "+c.numero);
    }
}
```



```
class Contatto {
    String nome;
    String numero;
    Contatto(String nome, String numero){
        nome= nome;
        this.numero=numero;
    }
    public static void main(String args[]){
        Contatto c= new Contatto("mario","3391234567");
        System.out.println(c.nome+" "+c.numero);
    }
}
```

## Ereditarietà

|                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>class A{     int i,j;     void stampaij(){         System.out.println(             "i e j = "+i+" "+j);     } }</pre>                                                    | <pre>class ProvaAeB{     public static void main(String         args[]) {         A c_madre= new A();         B c_figlia= new B();         c_madre.i=3; c_madre.j=6;         c_figlia.i=1; c_figlia.j=2;         c_figlia.k=9;         c_madre.stampaij();         c_figlia.stampaij();         c_figlia.stampak();         c_figlia.somma();     } }</pre> |
| <pre>class B extends A{     int k;     void stampak(){         System.out.println("k = "+k);     }     void somma(){         System.out.println("r= "+(i+j+k));     } }</pre> | <pre>i e j = 3 6 i e j = 1 2 k= 9 r= 12</pre>                                                                                                                                                                                                                                                                                                               |

|                                                                                                                                       |                                                                                                                                                                           |
|---------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>class A{     int i;     <b>private</b> int j;     void stampaij(){         System.out.println("i e j = "+i+" "+j);     } }</pre> | <pre>class B extends A{     int k;     void stampak(){         System.out.println("k = "+k);     }     void somma(){         System.out.println("r= "+(i+j+k)); } }</pre> |
|---------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

ERRORE j non è visibile a B

## Ridefinizione di metodi

- Si ha quando, in una gerarchia di classi, un metodo di una sottoclasse ha lo stesso nome e la stessa lista di parametri di un metodo della superclasse.
- In tal caso si dice che il metodo della sottoclasse ridefinisce il metodo della superclasse.
- Quando il metodo ridefinito viene chiamato da una sottoclasse verrà eseguito il codice definito nel metodo della sottoclasse.

```
class A{
    int i,j;
    void stampa (){
        System.out.println(" i e j = "+i
        +" "+j);
    }
}

class B extends A{
    int k;
    void stampa(){
        System.out.println(
        " i e j = "+i +" "+j);
        System.out.println("k = "+k);
    }
}
```

```
class ProvaAeB{
    public static void
    main(String args[]) {
        A c_madre= new A();
        B c_figlia= new B();
        c_madre.i=3;
        c_madre.j=6;
        c_figlia.i=1; c_figlia.j=2;
        c_figlia.k=9;
        c_madre.stampa();
        c_figlia.stampa();
    }
}
```

```
i e j = 3 6
i e j = 1 2
k= 9
```

## Ereditarietà e costruttori

- si utilizza la parola chiave super se nel definire il costruttore della sottoclasse si chiama il costruttore della superclasse:

```
super([parametri]);
```

**Questa deve essere la prima istruzione del costruttore della sottoclasse, dopo della quale si possono scrivere le istruzioni specifiche per il costruttore della sottoclasse.**

Esempio:

```
public ScatolaPesante(int a,int l,int p, double ps){
    super(a,l,p);
    peso= ps;
}
```

```
class A{
    int i,j;
    A(int a, int b){
        i=a;
        j=b;
    }
    void stampa (){
        System.out.println(" i e j = "+i
        +" "+j);
    }
}
```

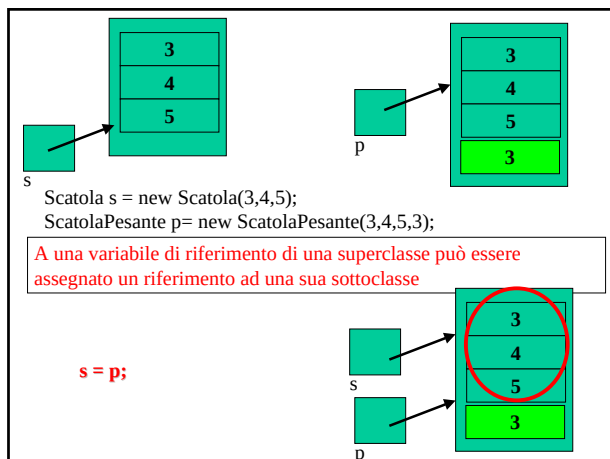
```
class B extends A{
    int k;
    B(int a, int b, int c){
        super(a,b);
        k=c;
    }
    void stampa(){
        super.stampa();
        System.out.println("k = "+k);
    }
}
```

```
class ProvaAeB{
    public static void main(String
    args[]) {
        A c_madre= new A(3,6);
        B c_figlia= new B(1,2,9);
        c_madre.stampa();
        c_figlia.stampa();
    }
}
```

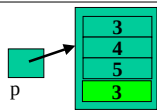
```
i e j = 3 6
i e j = 1 2
k= 9
```

## Osservazioni

- Ad una variabile di riferimento di una superclasse può essere assegnato un riferimento ad una sua sottoclasse
- In una gerarchia di classi i costruttori sono invocati in ordine di derivazione dalla superclasse alla sottoclasse
- Java esegue dinamicamente i metodi (binding dinamico)

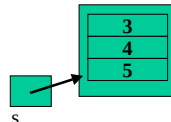


A una variabile di riferimento di una superclasse può essere assegnato un riferimento ad una sua sottoclasse



ScatolaPesante p= new ScatolaPesante(3,4,5,3);  
Scatola s = new Scatola(p);  
Supponendo di avere il seguente costruttore:

```
Scatola(Scatola a){
    altezza=a.altezza;
    larghezza=a.larghezza;
    profondita=a.profondita;
}
```



In una gerarchia di classi i costruttori sono invocati in ordine di derivazione dalla superclasse alla sottoclasse

```
class A {
    A(){
        System.out.println("classe A");
    }
}
```

```
class B extends A{
    B(){
        System.out.println("classe B");
    }
}
```

```
class C extends B{
    C(){
        System.out.println("classe C");
    }
}
```

```
class UsoC {
    public static void main (String args[])
    {
        C c= new C();
    }
}
```

Risultato esecuzione

```
C:\esercizi>javac UsoC.java
C:\esercizi>java UsoC
classe A
classe B
classe C
C:\esercizi>
```

## Cast

- **Conversione di un riferimento ad un oggetto da una classe a un'altra**
- **il tipo di destinazione deve essere racchiuso tra parentesi e collocato prima del riferimento di oggetto su cui eseguire il cast**

*Scatola box= new ScatolaPesante(2,3,4,10);*

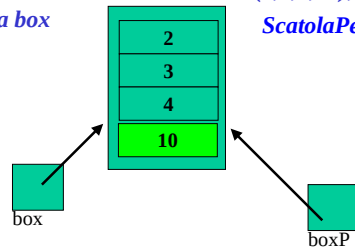
...

**ScatolaPesante boxP = (ScatolaPesante)box;**

*Scatola box= new ScatolaPesante(2,3,4,10);*

*Scatola box*

*ScatolaPesante boxP*



box.altezza  
box.larghezza  
box.profondità  
~~box.peso~~

**boxP = (ScatolaPesante)box;**

boxP.altezza  
boxP.larghezza  
boxP.profondità  
boxP.peso

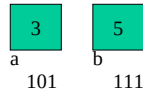
## Passaggio di parametri sovraccarico di metodi

## Passaggio parametri

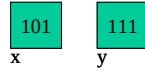
- **Passaggio per valore**
  - Le variabili così passate non vengono modificate dall'esecuzione del metodo
- **Passaggio per riferimento**
  - Le variabili così passate vengono modificate dall'esecuzione del metodo

### Generico Programma passaggio per riferimento

```
a=3; b=5;
funzione(a,b);
```

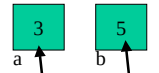


```
funzione(int x,int y){
  x=x+3;
  y=y+5
}
```



### Generico Programma passaggio per riferimento

```
a=3; b=5;
funzione(a,b);
```

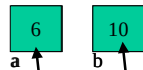


```
funzione(int x,int y){
  x=x+3;
  y=y+5
}
```



### Generico Programma passaggio per riferimento

```
a=3; b=5;
funzione(a,b);
```

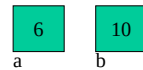


```
funzione(int x,int y){
  x=x+3;
  y=y+5
}
```



### Generico Programma passaggio per riferimento

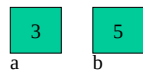
```
a=3; b=5;
funzione(a,b);
stampa(a,b);
```



Stamperà 6 e 10

### Generico Programma passaggio per valore

```
a=3; b=5;
funzione(a,b);
```

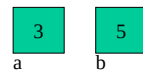


```
funzione(int x,int y){
  x=x+3;
  y=y+5
}
```



### Generico Programma passaggio per valore

```
a=3; b=5;
funzione(a,b);
stampa(a,b);
```



Stamperà 3 e 5

## Passaggio parametri

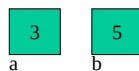
- **Passaggio per valore**
  - Le variabili così passate non vengono modificate dall'esecuzione del metodo
- **Passaggio per riferimento**
  - Le variabili così passate vengono modificate dall'esecuzione del metodo

## Passaggio parametri

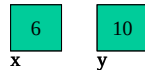
- **In Java i parametri sono sempre passati per valore**
  - **Se il parametro è un tipo primitivo è passato per valore**
  - Se il parametro è un oggetto: la variabile di riferimento è passato per valore
    - Tramite il riferimento all'oggetto si può accedere agli attributi e metodi dell'oggetto e modificare lo stato dell'oggetto

## Programma Java

```
...
a=3; b=5;
...
obj.funzione(a,b);
```



```
...
void funzione(int x,int y){
    x=x+3;
    y=y+5
}
...
```

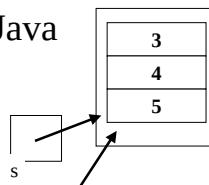


## Passaggio parametri

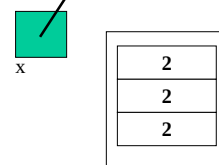
- **In Java i parametri sono sempre passati per valore**
  - Se il parametro è un tipo primitivo è passato per valore
  - **Se il parametro è un oggetto: la variabile di riferimento è passato per valore**
    - Tramite il riferimento all'oggetto si può accedere agli attributi ed i metodi dell'oggetto e modificare lo stato dell'oggetto

## Programma Java

```
...
a=3; b=5;
Scatola s =new Scatola(3,4,5);
obj.funzione(s);
```

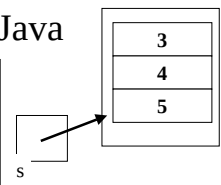


```
...
void funzione(Scatola x){
    x=new Scatola(2,2,2);
}
...
```

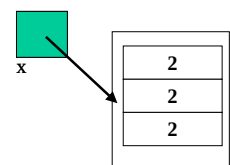


## Programma Java

```
...
a=3; b=5;
Scatola s =new Scatola(3,4,5);
obj.funzione(s);
```

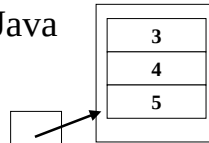


```
...
void funzione(Scatola x){
    x=new Scatola(2,2,2);
}
...
```



### Programma Java

```
...
a=3; b=5;
Scatola s =new Scatola(3,4,5);
obj.funzione(s);
...
```



```
System.out.println(s.altezza+" "+s.larghezza+" "+s.profondità);
```

Stamperà: 3 4 5

### Passaggio parametri

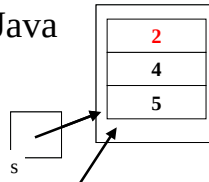
#### • In Java i parametri sono sempre passati per valore

- Se il parametro è un tipo primitivo è passato per valore
- Se il parametro è un oggetto: la variabile di riferimento è passato per valore

• **Tramite il riferimento all'oggetto si può accedere agli attributi ed ai metodi dell'oggetto e modificare lo stato dell'oggetto**

### Programma Java

```
...
a=3; b=5;
Scatola s =new Scatola(3,4,5);
obj.funzione(s);
```

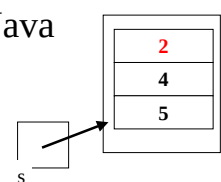


```
...
void funzione(Scatola x){
    x.altezza =2;
}
...
```



### Programma Java

```
...
a=3; b=5;
Scatola s =new Scatola(3,4,5);
obj.funzione(s);
...
```

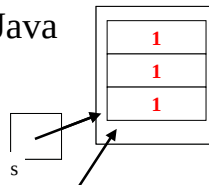


```
System.out.println(s.altezza+" "+s.larghezza+" "+s.profondità);
```

Stamperà: **2** 4 5

### Programma Java

```
...
a=3; b=5;
Scatola s =new Scatola(3,4,5);
obj.funzione(s);
```

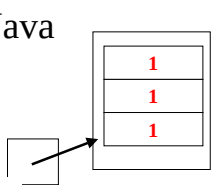


```
...
void funzione(Scatola x){
    x.setDimensioni(1,1,1);
}
...
```



### Programma Java

```
...
a=3; b=5;
Scatola s =new Scatola(3,4,5);
obj.funzione(s);
...
```



```
System.out.println(s.altezza+" "+s.larghezza+" "+s.profondità);
```

Stamperà: **1 1 1**

## Package

### Come il compilatore trova i package

- Tutti i file di un package devono essere situati in una subdirectory (rispetto ad una directory nominata nella variabile di sistema CLASSPATH) corrispondente al nome completo del package

Esempio:

il package **scatole** si deve trovare in una sottodirectory **scatole**

### Come il compilatore trova i package

```
package scatole;  
class Scatola{  
    int altezza, larghezza, profondit ;  
}
```

```
import scatole.Scatola;  
class UsoScatola{  
    public static void main(String args[]) {  
        ....  
    }  
}
```

C:\esercizi\scatole>javac Scatola.java

C:\esercizi>javac UsoScatola.java

C:\esercizi>java UsoScatola

### Come il compilatore trova i package

```
package scatole;  
class Scatola{  
    int altezza, larghezza, profondit ;  
    ...  
    public static void main(String args[]) {  
        ....  
    }  
}
```

C:\esercizi\scatole>javac Scatola.java

C:\esercizi>java scatole.Scatola

## Eccezioni

```
class Libro{  
    String titolo;  
    int numero_pagine;  
    double costo;  
    int pagina_corrente;  
    Libro(String s, int a){  
        titolo=s;  
        numero_pagine=a;  
    }  
    void setCosto(double x){  
        costo=x;  
    }  
    void apriLibro(int p){  
        pagina_corrente=p;  
    }  
    int getPagina(){  
        return pagina_corrente;  
    }  
}
```

```

class UsoLibro0{
public static void main (String args[]){
    Libro a = new Libro("pippo", 45);
    a.setCosto(22.4);
    a.apriLibro(-3);
    System.out.println("\nlibro pg="+a.getPagina());
}
}

```

```

C:\esercizi>java UsoLibro0
libro pg=-3
C:\esercizi>

```

```

class Libro1{
    String titolo; int numero_pagine,pagina_corrente;
    double costo;
    Libro1(String s, int a){
        titolo=s;
        numero_pagine=a; }
    void setCosto(double x){
        costo=x;}

    void apriLibro(int p) throws MyException{
        if ((p<0)||(p>numero_pagine)) throw new
        MyException("pagina non compresa nel libro");
        else pagina_corrente=p;
    }

    int getPagina(){return pagina_corrente; }}

```

```

class MyException extends Exception{
    private String message;
    MyException(String message) {
        this.message = message;
    }
    public String toString() {
        return "La mia eccezione : "+ message+" ";
    }
}

```

```

class UsoLibro1{
    public static void main (String args[]){
        Libro1 a = new Libro1("pippo", 45);
        a.setCosto(22.4);
        a.apriLibro(-3);
        System.out.println("libro1 pg="+a.getPagina());
    }
}

```

```

C:\esercizi>javac UsoLibro1.java
UsoLibro1.java:7: unreported exception MyException; must be
caught or declared to be thrown
    a.apriLibro(-3);
    ^
1 error

```

```

class UsoLibro1{
public static void main (String args[]){
    Libro1 a = new Libro1("pippo", 45);
    a.setCosto(22.4);
    try{
        a.apriLibro(-3);
    }catch(MyException e){
        System.out.println("errore in apriLibro "+e);
    }
    System.out.println("\nlibro1 pg="+a.getPagina());
}
}

```

```

C:\esercizi>java UsoLibro1
Errore in apriLibro La mia eccezione : pagina non compresa nel libro
libro1 pg=0
C:\esercizi>

```

```

class Libro{
    String titolo;
    int numero_pagine;
    double costo;
    int pagina_corrente;
    Libro(String s, int a){
        titolo=s;
        numero_pagine=a;
    }
    void setCosto(double x){
        costo=x;
    }
    void apriLibro(int p){
        pagina_corrente=p;
    }
    int getPagina(){
        return pagina_corrente;
    }
}

```

|                                                                                                                                                                                                                                                                                                   |                                                                                                                               |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <pre> class <b>Libro1a</b> extends Libro{     Libro1a(String s, int a){         super(s,a);     }     void apriLibro(int p) throws MyException{         if ((p&lt;0)  p&gt;numero_pagine)) throw new MyException("pagina non compresa nel libro");         else pagina_corrente=p;     } } </pre> | <p><b>Non è possibile ridefinire un metodo facendo lanciare una eccezione se nella superclasse non lanciava eccezioni</b></p> |
| <pre> C:\esercizi&gt;javac Libro1a.java Libro1a.java:5: apriLibro(int) in Libro1a cannot override apriLibro(int) in Libro; overridden method does not throw MyException     void apriLibro(int p) throws MyException{         ^ 1 error </pre>                                                    |                                                                                                                               |