Uso di metodi statici

Walter Didimo

Metodi di istanza

Fino ad ora abbiamo imparato a creare oggetti e ad invocare metodi su tali oggetti

- i metodi venivano eseguiti dagli oggetti
- un metodo invocato su un oggetto ed eseguito dall'oggetto è chiamato metodo di istanza
- il termine "metodo di istanza" indica proprio che tale metodo va invocato sull'istanza di una classe (cioè un oggetto)

Metodi statici

In Java anche le classi possono offrire direttamente dei servizi:

- una classe può cioè definire dei metodi che essa stessa (non un suo oggetto) eseguirà
- tali metodi si chiamano metodi statici, o anche metodi di classe
- un metodo statico va invocato sulla classe – l'esecuzione del metodo non prevede la presenza di un oggetto

Riconoscere metodi statici

Quando si deve usare una classe scritta da qualcuno, come si riconoscono i metodi statici da quelli di istanza?

- chi scrive una classe definisce i metodi statici usando una parola chiave del linguaggio Java, la parola static
- chi ha scritto la classe deve documentare che un metodo è statico mettendo la parola *static* davanti al prototipo del metodo

Esempio di metodi statici

La classe *String* ha alcuni metodi statici, oltre ai metodi di istanza che abbiamo studiato

Ad esempio, il seguente metodo

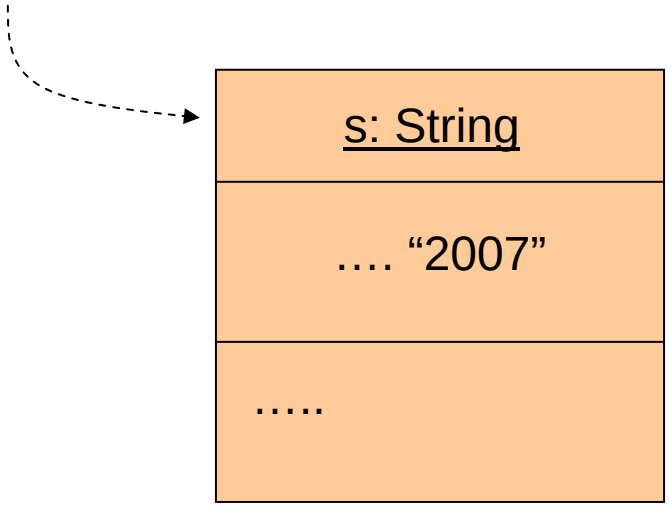
static String valueOf (int a)

restituisce una stringa che rappresenta il numero intero *a*

Esempio di uso di valueOf

Poiché *valueOf* è un metodo statico, esso va invocato sulla classe (la sua esecuzione non è richiesta ad uno specifico oggetto)

String s = String.valueOf (2007); // s vale “2007”



Varianti di valueOf

Il metodo *valueOf* è sovraccarico, nel senso che per ogni tipo primitivo di dato *tipo*, esiste il metodo

static String valueOf (tipo a);

valueOf restituisce sempre la stringa che rappresenta letteralmente il valore primitivo contenuto in *a*

Altre classi con metodi statici

Nella API di Java esistono molte classi che offrono metodi statici

Una di queste classi particolarmente utile è la classe *Math* (nel package *java.lang*)

- la classe *Math* ha soltanto metodi statici
- ciascuno di tali metodi svolge una funzione matematica di uso frequente

Metodi della classe Math

Ecco alcuni metodi della classe *Math*:

```
static double abs (double a)          // ritorna il valore assoluto di a
static double pow (double a, double b) // ritorna a^b
static double sqrt (double a)         // ritorna la radice quadrata di a
static double sin (double a)          // ritorna il seno di a
static double cos (double a)          // ritorna il coseno di a
static double tan (double a)          // ritorna la tangente di a
static double log (double a)          // ritorna il log. di a in base e
static double log10 (double a)        // ritorna il log. di a in base 10
…..
```

Costanti nella classe Math

La classe *Math* definisce anche due costanti di uso comune, usando la massima precisione possibile

Math.PI	la costante π = 3.14159…
Math.E	la costante *e* = 2.7182 …

Esempio di uso di Math

Il seguente programma fa inserire all'utente il raggio di un cerchio e ne visualizza l'area sullo standard output

```
class AreaCerchio{
  public static void main (String[] args){
    InputWindow in = new InputWindow ();
    double r = in.readDouble ("Raggio");
    double a = Math.pow(r,2)*Math.PI;
    System.out.println ("raggio = " + a);
  }
}
```

.....

Un problema ricorrente

Supponiamo che, dato un valore *String* che rappresenta un numero intero, si voglia ottenere il valore intero corrispondente

Ad esempio, supponiamo che *s* è una stringa che rappresenta una data nella forma gg/mm/aa (es. “07/11/2006”) e supponiamo di voler memorizzare l’anno in una variabile intera

L’anno è rappresentato dalla sottostringa *s.substring(6)*, ma come possiamo ottenere il valore intero corrispondente?

La classe Integer

L'API di Java fornisce una classe che permette di risolvere il problema, la classe *Integer*

La classe *Integer* offre un metodo statico che restituisce il valore intero associato ad una stringa

static int parseInt(String s)

Si può ad esempio scrivere

```
int anno = Integer.parseInt(s.substring(6));
```

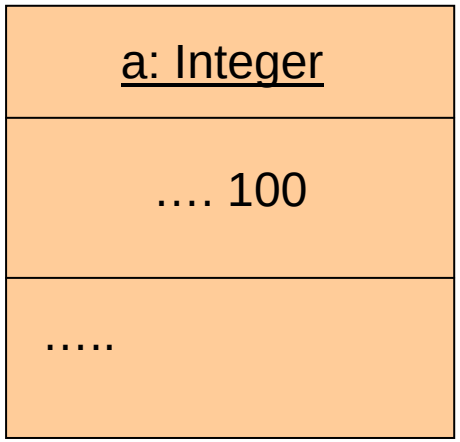
Classi wrapper

La classe *Integer* è chiamata classe wrapper ("wrapper" significa "incartatrice")

In effetti, la classe *Integer* permette di creare oggetti

- un oggetto *Integer* rappresenta un valore *int*
- lo stato dell'oggetto è il valore *int* rappresentato

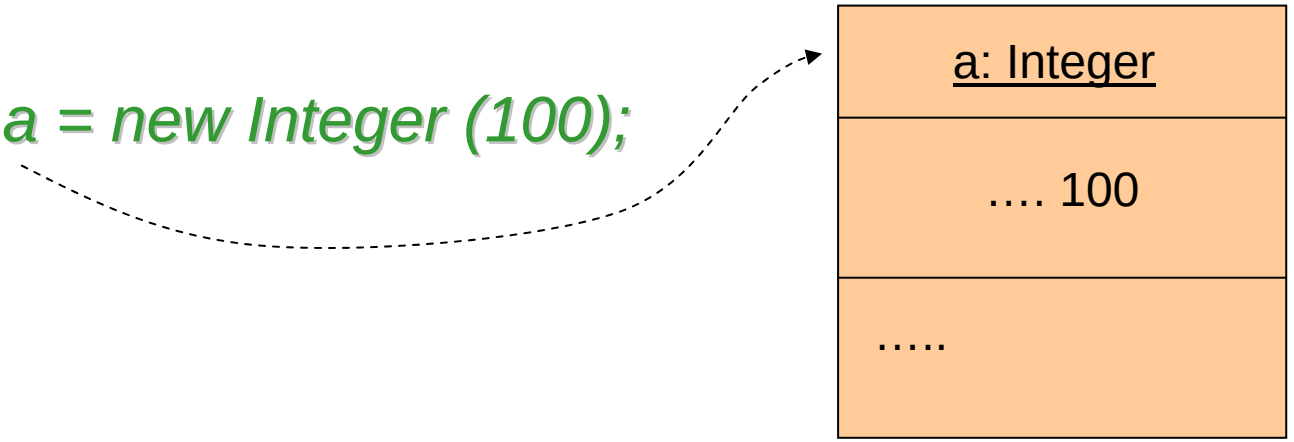
Integer a = new Integer (100);



Gli oggetti Integer

E’ possibile creare un oggetto *Integer*, usando un apposito costruttore, che specifica l’intero rappresentato

Integer a = new Integer (100);



Il metodo di istanza *intValue* chiede ad un oggetto *Integer* di restituire il valore *int* che esso rappresenta

int b = a.intValue( );

Altre classi wrapper

L'API di Java offre una classe wrapper per ogni tipo di dato primitivo:

Double	classe wrapper per i *double*
Character	classe wrapper per i *char*
Boolean	classe wrapper per i *boolean*

….

Ogni classe wrapper *X* (tranne *Character*) ha un metodo statico di nome *parseX;* ogni classe wrapper ha inoltre un metodo di istanza per ottenere il valore primitivo incartato (*doubleValue(), charValue(), …*)

Il metodo speciale main

Il metodo speciale *main*, che serve ad avviare l'esecuzione di un programma, è un metodo statico

Se X è una classe che contiene il metodo speciale *main*, il programma può essere avviato chiamando (indirettamente) il metodo *main* sulla classe X

- non esistono oggetti di X prima che il programma inizi la sua esecuzione

Glossario dei termini principali

Termine	Significato
Metodo di istanza	Metodi che vengono invocati sulle istanza (oggetti) di una classe – i metodi visti fin qui sono metodi di istanza
Metodo statico (o di classe)	Metodo che viene invocato sulla classe – è la classe che esegue il metodo non un suo oggetto
static	Parola chiave del linguaggio Java usata per definire metodi statici
Math	Classe dell'API di Java che offre solo metodi statici, utili nei calcoli matematici
Classi wrapper	Classi dell'API di Java che "incartano" valori primitivi. Esiste una classe wrapper per ogni tipo primitivo (es. Integer, Double, Character, …)