

Static

Metodi static

- **Esempio**

```
public class Financial
{
    public static double percentOf(double p, double a)
    {
        return (p / 100) * a;
    }
    // . . .
}
```

Metodi static

- Altri esempi ben noti
 - `main()`
 - i metodi della classe `Math`

<code>Math.sqrt(x)</code>	Radice quadrata
<code>Math.pow(x, y)</code>	Potenza x^y
<code>Math.exp(x)</code>	<i>Esponenziate e^x</i>
<code>Math.log(x)</code>	Logaritmo naturale
<code>Math.sin(x)</code> , <code>Math.cos(x)</code> , <code>Math.tan(x)</code>	Funzioni trigonometriche (x in radianti)
<code>Math.round(x)</code>	Attotondamento
<code>Math.min(x, y)</code> , <code>Math.max(x, y)</code>	Minimo, massimo

Metodi static

- **Sono definiti all'interno di classe, ma ...**
 - non sono associati ad oggetti
 - non sono invocati su oggetti
- **Quando definire un metodo static?**
 - quando il metodo opera solamente sui propri parametri

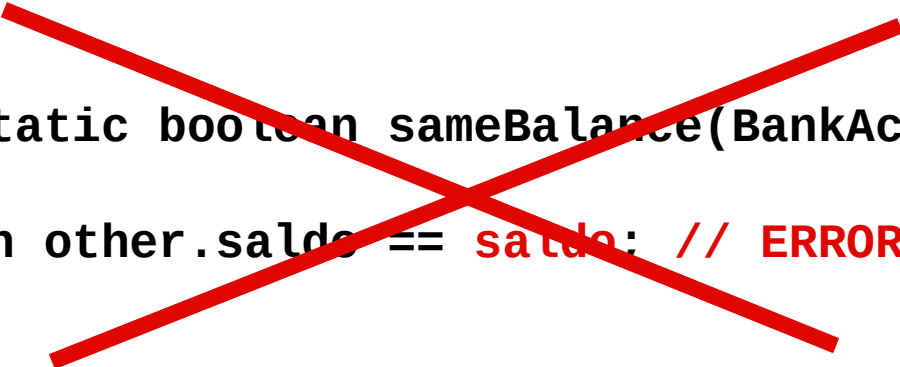
Metodi static

- Poichè appartengono alla classe e non sono riferibili ad alcuna istanza, all'interno di questi metodi il parametro implicito **this** è indefinito
- Quindi all'interno di questi metodi non si può far riferimento ad alcun campo dell'oggetto corrente

Metodi static

- Attenzione agli accessi

```
public class BankAccount
{
    . . .
    public static boolean sameBalance(BankAccount other)
    {
        return other.saldo == saldo: // ERRORE!
    }
    . . .
}
```



Metodi static

- Correggiamo così

```
public class BankAccount
{
    . . .
    public static boolean sameBalance(BankAccount other)
    {
        return other.saldo == saldo; // OK
    }
    . . .
}
```

Metodi static

- Oppure così

```
public class BankAccount
{
    . . .
    public static boolean sameBalance(BankAccount b1,
                                      BankAccount b2)
    {
        return b1.saldo == b2.saldo; // OK
    }
    . . .
}
```


Chiamata di metodi statici

- **All'interno della classe in cui sono definiti**
 - Stessa sintassi utilizzata per i metodi non-static
- **All'esterno della classe di definizione**
 - Utilizziamo **il nome della classe** invece del riferimento ad un oggetto:

```
double tax = Financial.percentOf(taxRate, total);
```

Domande

- È corretto invocare `x.pow(y)` per calcolare x^y ?
- Quale delle due istruzioni è più efficiente tra `x*x` e `Math.pow(x, 2)`

Risposte

- **No: x è un numero, non un oggetto, e non è possibile invocare un metodo su un numero**
- **$x*x$ la seconda espressione coinvolge una chiamata di metodo, passaggio di parametri ... etc, tutte operazioni costose**

Domanda

- Come utilizzereste la classe **MyMath** qui di seguito per calcolare la radice quadrata di 5?

```
public class MyMath
{
    public double sqrt(double x)
    {
        return Math.sqrt(x);
    }
}
```

Risposta

```
(new MyMath()).sqrt(5);
```

Campi static (class variables)

- Un campo static appartiene alla classe, non ad alcuna delle istanze della classe

```
public class BankAccount
{
    . . .
    private double saldo;
    private int accountNumber;
    private static int lastAssignedNumber = 1000;
}
```

- Una sola copia di `lastAssignedNumber`, accessibile solo all'interno della classe

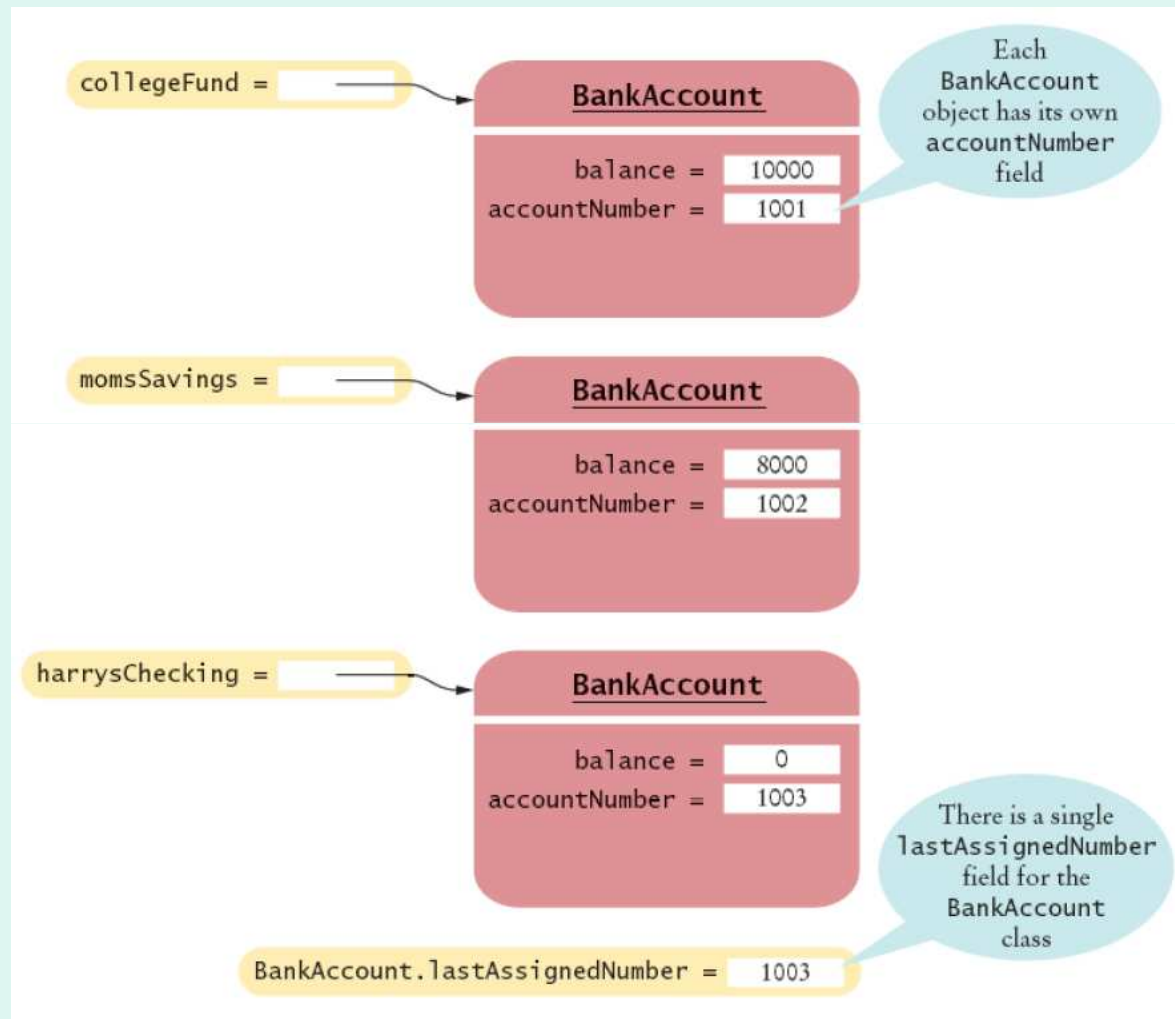
Campi static

- La nuova definizione del costruttore

```
/**
    Genera il prossimo numero di conto da assegnare
 */
public BankAccount()
{
    lastAssignedNumber++;
    accountNumber = lastAssignedNumber;
}
```

- Questo è un uso tipico dei campi static
 - Altro caso di utilizzo: definizione di costanti (campi static e final)

Campi e campi static



Campi static

Tre modalità di inizializzazione:

1. Automatica: inizializzati ai valori di default.
2. Mediante un iniziatore esplicito

```
public class BankAccount
{
    . . .
    private static int lastAssignedNumber = 1000;
        // Eseguito una volta sola,
        // quando la classe viene caricata
}
```

3. Mediante un blocco di inizializzazione static

Continua...

Campi static

- **Allocazione/Deallocazione:**
 - vengono creati quando viene caricata la classe,
 - continuano ad esistere finchè esiste la classe, quindi durante tutta l'esecuzione.

Campi static

- Come tutti i campi, preferibilmente dichiarati **private**
- Eccezioni: costanti

```
public class BankAccount
{
    . . .
    public static final double OVERDRAFT_FEE = 5;
        // riferita mediante il nome qualificato
        // BankAccount.OVERDRAFT_FEE
}
```

Accesso ai campi static

- **All'interno della classe in cui sono definiti**
 - Stessa sintassi utilizzata per i campi non-static
- **Da una classe diversa da quella di definizione**
 - Utilizziamo il nome della classe invece del riferimento ad un oggetto:

```
double fee = BankAccount.OVERDRAFT_FEE;
```

```
Printstream ps = System.out;
```

Domanda

- L'istruzione `System.out.println(4)` denota una chiamata di metodo static?

Risposta

- **No: il metodo `println()` è invocato qui sull'oggetto `System.out`**

Costanti: `final`

- Una variabile `final` è una costante
- Una volta inizializzata, non è possibile modificarne il valore
- Convenzione: i nomi delle variabili usano solo caratteri MAIUSCOLI

```
final double QUARTER = 0.25;  
final double DIME = 0.1;  
final double NICKEL = 0.05;  
final double PENNY = 0.01;
```

Costanti: `static final`

- Costanti utilizzate da più di un metodo possono essere dichiarate come campi, e qualificate come `static` e `final`
- Le costanti `static final` possono inoltre essere dichiarate `public` per renderle disponibili ad altre classi

```
public class Math
{
    . . .
    public static final double E = 2.7182818284590452354;
    public static final double PI = 3.14159265358979323846;
}

double circonferenza = Math.PI * diametro;
```


La classe Pila - 1

```
class Pila {  
    private int size;  
    private int defaultGrowthSize;  
    private int marker;  
    private contenuto[];  
    final int INITSIZE=3;  
  
    public Pila() {  
        size=initialSize;  
        defaultGrowthSize=INITSIZE;  
        marker=0;  
        contenuto=new int[size];  
    }
```

La classe Pila - 2

```
private void cresci(int dim){  
    size+=dim;  
  
    int temp[ ]=new int[size];  
    for (int k=0;k<marker;k++)  
        temp[k]=contenuto[k];  
  
    contenuto=temp;  
}
```

La Classe Pila - 3

```
public void inserisci(int k) {  
    if (marker==size){  
        cresci(defaultGrowthSize;)}  
    contenuto[marker]=k;  
    marker++;  
}
```

La Classe Pila - 4

```
public int estrai() {  
    if (marker==0) {  
        System.out.println(  
            "Non posso estrarre da una pila vuota");  
        System.exit(1);  
    }  
    return contenuto[--marker];  
}
```