

# OOP in JAVA



Programmazione  
orientata agli Oggetti

Giulio Angiani  
I.I.S. "Blaise Pascal" - Reggio Emilia





# OOP

## Object Oriented Programming



# Un esempio completo: Numero Razionale

Vogliamo realizzare una classe Java che modelli il concetto matematico di **numero razionale**

Di cosa abbiamo bisogno? Come è fatto un **numero razionale**

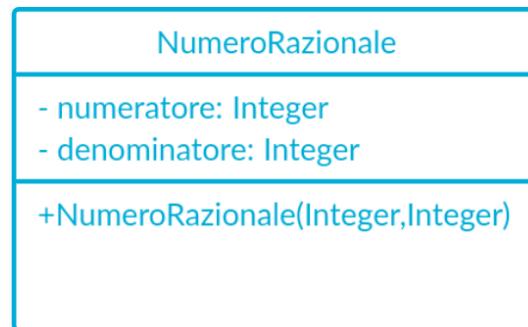
Abbiamo bisogno di due numeri **interi**

- un numeratore
- un denominatore (che sia diverso da zero)



Possiamo graficamente scriverlo con il formalismo **UML**<sup>1</sup> nel modo seguente

Nome della classe ==>



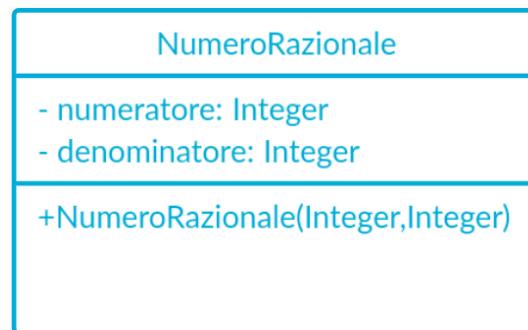
Attributi ==>

Metodi ==>



# Un esempio completo: Numero Razionale (2)

Nel nostro caso, inizialmente, abbiamo un **costruttore** che riceve due parametri interi che rappresentano il numeratore ed il denominatore del nostro numero razionale



```
public class NumeroRazionale {  
  
    private int numeratore;  
    private int denominatore;  
  
    public NumeroRazionale(int n, int d) {  
  
    }  
  
}
```

JAVA



# Un esempio completo: Numero Razionale (3)

Inseriamo nel costruttore le istruzioni per **inizializzare** gli attributi privati

```
public class NumeroRazionale {  
  
    private int numeratore;  
    private int denominatore;  
  
    public NumeroRazionale(int n, int d) {  
        this.numeratore = n;  
        this.denominatore = d;  
    }  
}
```



Nel **main** proviamo subito se funziona l'instanziazione di un oggetto della nuova classe

```
NumeroRazionale nr = new NumeroRazionale(2,3);  
System.out.println("nr = " + nr);
```

JAVA

```
nr = matematica.NumeroRazionale@2ff4acd0 // <== rappresentazione standard di un oggetto qualsiasi  
// package.<tipo>@<hashcode> (hashcode è in esadecimale)
```

OUTPUT



# Un esempio completo: Numero Razionale (4)

E' possibile che un denominatore sia **zero** ?

NO!

Dobbiamo controllare l'input del costruttore!

Se il denominatore è uguale a zero NON devo permettere la creazione dell'oggetto!

Se succede cacciamo fuori una **eccezione** (errore run-time)

```
public NumeroRazionale(int n, int d) throws Exception { // <= potrebbe restituire una eccezione
    this.numeratore = n;

    if (d == 0) {
        throw new Exception("Il denominatore non può essere zero!"); // <= caccio fuori una eccezione con descrizione
    }
    else {
        // tutto ok, assegniamo questo valore all'attributo privato
        this.denominatore = d;
    }
}
```

JAVA



# Un esempio completo: Numero Razionale (5)

Proviamo il nuovo costruttore nel `main`

Prima potevamo scrivere

```
NumeroRazionale nr2 = new NumeroRazionale(2,0);  
System.out.println("nr2 = " + nr2);
```

JAVA

```
nr2 = matematica.NumeroRazionale@54bedef2
```

OUTPUT (PRIMA)

Adesso otteniamo:

```
Exception in thread "main" java.lang.Exception: Il denominatore non può essere zero!  
at matematica.NumeroRazionale.<init>(NumeroRazionale.java:15)  
at matematica.MainNumeroRazionale.main(MainNumeroRazionale.java:16)
```

OUTPUT (DOPO)



# Un esempio completo: Numero Razionale (6)

NOTA: Il programma di test è cambiato nell'intestazione del metodo **main** che diventa

```
public static void main(String[ ] args) throws Exception { // <== QUI  
    NumeroRazionale nr = new NumeroRazionale(2,3);  
    System.out.println("nr = " + nr);  
  
    NumeroRazionale nr2 = new NumeroRazionale(2,0);  
    System.out.println("nr2 = " + nr2);  
}
```

JAVA

Java si è accorto che l'instanziazione della classe **NumeroRazionale** potrebbe restituire una eccezione!

Ci obbliga a gestirla (come per i file di testo)!

Possiamo mettere un **try-catch** intorno all'istruzione di creazione dell'oggetto oppure, come in questo caso, **propagare** l'errore a run-time

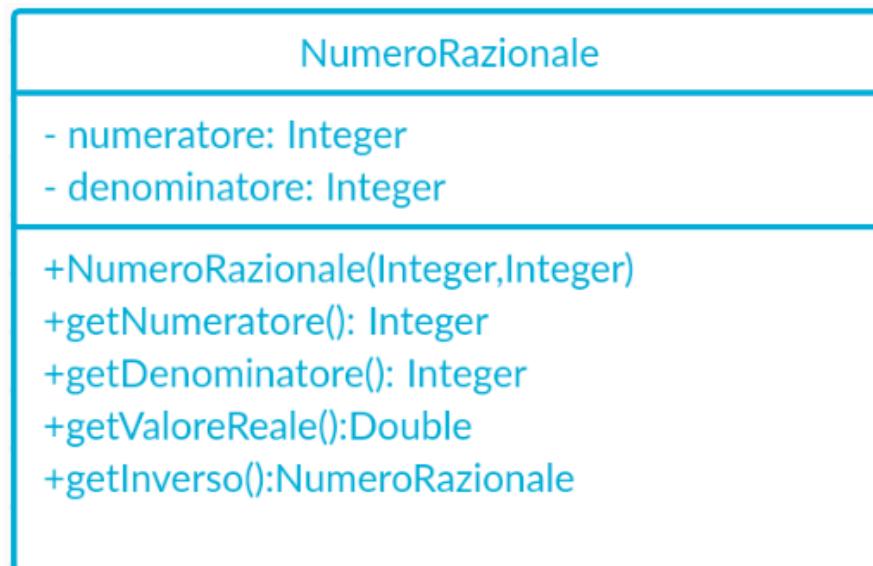
Essendo questa la procedura di **main**, se c'è un errore, l'applicazione si rompe.



# Un esempio completo: Numero Razionale (7)

Esercizi:

Completare la classe NumeroRazionale implementando tutti i metodi previsti dal modello UML in figura



# Un esempio completo: Numero Razionale (8)

Soluzioni:

```
public int getNumeratore() {  
    return this.numeratore;  
}  
  
public int getDenominatore() {  
    return this.denominatore;  
}  
  
public double getValoreReale() {  
    // prima trasformo in double altrimenti verrebbe sempre zero  
    return (double)(this.numeratore)/this.denominatore;  
}  
  
public NumeroRazionale getInverso() throws Exception {  
    // istanzio al volo un altro oggetto con numeratore e denominatore invertiti  
    // NOTA: potrebbe dare una eccezione (che va dichiarata) perché  
    // potrei avere l'inverso di un numero con numeratore 0 : per esempio 0/3  
    return new NumeroRazionale(this.denominatore, this.numeratore);  
}
```

JAVA



# Un esempio completo: Numero Razionale (8)

Esercizi:

Vogliamo creare una rappresentazione leggibile dell'oggetto NumeroRazionale che mostri ad esempio il numero con numeratore 3 e denominatore 4 con il formato **[3/4]**

Dobbiamo ridefinire il metodo **toString()**

Il metodo è sempre presente in ogni oggetto e va riscritto se si vuole modificarlo

```
public String toString() {  
    return "[" + this.numeratore + "/" + this.denominatore + "]";  
}
```

JAVA

```
NumeroRazionale n = new NumeroRazionale(3,4);  
System.out.println("n = " + n);
```

JAVA

n = [3/4]

OUTPUT



# Un esempio completo: Numero Razionale (9)

Esercizi:

Vogliamo introdurre un metodo **riduciAiMinimiTermini()** che modifichi lo stato interno di un oggetto in modo che si abbia sempre la rappresentazione minima di un numero razionale

Esempio: si vuole quindi che il numero [4/6] venga trasformato in [2/3] con l'applicazione del metodo indicato

```
NumeroRazionale n = new NumeroRazionale(4,6);
System.out.println("(prima) n = " + n);
n.riduciAiMinimiTermini();
System.out.println("(dopo)  n = " + n);
```

JAVA

deve produrre

```
(prima) n = [4/6]
(dopo)  n = [2/3]
```

OUTPUT

NOTA: per sapere se un numero **x** è divisore di **y** basta controllare tutte le divisioni fino a **sqrt(y)**



# Un esempio completo: Numero Razionale (10)

Soluzione:

```
// metodo privato di appoggio che calcola il massimo comune divisore fra due numeri
private int getMassimoDivisore(int N, int D) {
    for (int i = Math.min(N,D); i > 1; i--) {
        if ((N%i == 0) && (D%i==0)) return i;
    }
    return 1;
}

public void riduciAiMinimiTermini() {
    int mcd = this.getMassimoDivisore(this.numeratore, this.denominatore);
    this.numeratore /= mcd; // come scrivere ==> this.numeratore = this.numeratore / mcd;
    this.denominatore /= mcd;
}
```

JAVA



# Un esempio completo: Numero Razionale (10-bis)

Altra Soluzione :

```
public void riduciAiMinimiTermini() {  
    while (true) {  
        boolean finito = true;  
        for (int d = 2; d < Math.sqrt(this.numeratore); d++) {  
            if (this.numeratore % d == 0) {  
                System.out.println(d + " è divisore di " + this.numeratore);  
                if (this.denominatore % d == 0) {  
                    System.out.println(d + " è divisore anche di " + this.denominatore);  
                    // divido e ricomincio  
                    this.numeratore = this.numeratore / d;  
                    this.denominatore = this.denominatore / d;  
                    finito = false;  
                }  
            }  
        }  
        if (finito) break;  
    }  
}
```

JAVA



# Un esempio completo: Numero Razionale (11)

Esercizi:

Vogliamo introdurre un metodo **add(NumerRazionale)** che aggiunge un numero razionale all'oggetto istanziato

Le seguenti istruzioni

```
NumerRazionale n1 = new NumerRazionale(2,3);
NumerRazionale n2 = new NumerRazionale(3,4);
n1.add(n2);
System.out.println("Ora n1 vale : " + n1);
```

JAVA

devono produrre il seguente output

```
Ora n1 vale : [17/12]
```

OUTPUT



# Un esempio completo: Numero Razionale (12)

Esercizi:

Vogliamo introdurre un metodo **mult(NumerRazionale)** che moltiplica un numero razionale all'oggetto istanziato

Le seguenti istruzioni

```
NumerRazionale n1 = new NumerRazionale(2,3);
NumerRazionale n2 = new NumerRazionale(3,4);
n1.mult(n2);
System.out.println("Ora n1 vale : " + n1);
```

JAVA

devono produrre il seguente output

```
Ora n1 vale : [1/2]
```

OUTPUT



# Un esempio completo: Numero Razionale (13)

Soluzioni:

JAVA

```
public void add(NumerRazionale other) {  
    int comunedenominatore = other.getDenominatore()*this.denominatore;  
    int numeratore_risultato = other.getDenominatore()*this.numeratore + this.denominatore*other.getNumeratore();  
    this.numeratore = numeratore_risultato;  
    this.denominatore = comunedenominatore;  
    this.riduciAiMinimiTermini(); // dopo l'operazione invoco il metodo di riduzione  
}  
  
public void mult(NumerRazionale other) {  
    this.numeratore = this.numeratore*other.getNumeratore();  
    this.denominatore = this.denominatore*other.getDenominatore();  
    this.riduciAiMinimiTermini(); // dopo l'operazione invoco il metodo di riduzione  
}
```





Giulio Angiani  
I.I.S. "Blaise Pascal" - Reggio Emilia