Classi e array

Viene ora affrontato un problema di definizione di una classe in cui una variabile d'istanza è di tipo array

Si vuole definire una classe **Polinomio** per la rappresentazione di polinomi a coefficienti reali

- un oggetto **Polinomio** rappresenta un polinomio a coefficienti reali
  - ad esempio, un oggetto **Polinomio** può rappresentare il polinomio $p(x) = x^3 + 2x - 1$
- un oggetto **Polinomio** sa eseguire le seguenti operazioni
  - calcolare il proprio grado
  - calcolare il proprio valore in un punto
  - calcolare una stringa che sia una propria descrizione

Altre operazioni

- creazione di un nuovo **Polinomio**
  - dati i coefficienti del polinomio

Polinomio — interfaccia

Metodi d'istanza di **Polinomio**

- metodo d'istanza **int grado()**
  - restituisce il grado del polinomio
  - ad esempio, il polinomio $p(x) = x^3 + 2x - 1$ ha grado 3
- metodo d'istanza **double valore(double x)**
  - restituisce il valore del polinomio nel punto **x**
  - ad esempio, il polinomio $p(x) = x^3 + 2x - 1$ valutato nel punto x=2 vale 11
- metodo d'istanza **String toString()**
  - restituisce una descrizione del polinomio sotto forma di stringa
  - ad esempio, il polinomio $p(x) = x^3 + 2x - 1$ può essere la stringa **"+1.0*x^3 +2.0*x -1.0"**

Polinomio — interfaccia

Costruttori

- costruttore **Polinomio(double[] coefficienti)**
  - crea un nuovo **Polinomio**, dato un array di **coefficienti**
- vengono fatte la seguente ipotesi sull'interpretazione dell'array **coefficienti**
  - l'elemento di indice **i** dell'array **coefficienti** rappresenta il coefficiente del termine di grado **i** del polinomio da creare
  - ad esempio, il primo elemento di **coefficienti** rappresenta il coefficiente del termine costante del polinomio
  - se il polinomio da creare ha grado maggiore di zero, allora l'ultimo elemento di **coefficienti** (il coefficiente del termine di grado più alto del polinomio) è diverso da zero
  - ad esempio, per creare il polinomio $p(x) = x^3 + 2x - 1$ deve essere usata la seguente espressione
    new Polinomio(new double[] { -1, 2, 0, 1 })
- il grado del polinomio da creare è quindi pari alla lunghezza dell'array **coefficienti** diminuita di 1

Polinomio — metodo di test

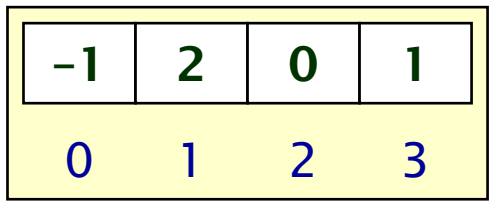
```
/* Applicazione di esempio di uso per Polinomio. */
public static void main(String[] args) {
    Polinomio p;

    p = new Polinomio( new double[] { -1, 2, 0, 1 } );
    /* p rappresenta x^3 +2*x -1 */

    System.out.println(p.grado());      // 3

    System.out.println(p.valore(0));    // -1.0
    System.out.println(p.valore(2));    // 11.0

    System.out.println(p.toString());
    // +1.0*x^3+2.0*x-1.0
}
```

Polinomio — variabili d'istanza

Un oggetto **Polinomio** rappresenta un polinomio a coefficienti reali

- le proprietà di un polinomio a coefficienti reali sono
  - il grado N del polinomio
  - gli N+1 coefficienti dei termini del polinomio
- queste proprietà possono essere rappresentate da un array **coefficienti** di numero reali
  - l'elemento i-esimo di **coefficienti** rappresenta il coefficiente del termine di grado i del polinomio
  - il grado del polinomio viene rappresentato (indirettamente) dalla lunghezza dell'array **coefficienti**

Ad esempio, il seguente array rappresenta lo stato del polinomio $p(x) = x^3 + 2x - 1$



Polinomio — struttura della classe

```
/** Un Polinomio rappresenta un polinomio
 *  a coefficienti reali.                    */
class Polinomio {
    /** Coefficienti del polinomio. */
    private double[] coefficienti;
    /* l'elemento i-esimo rappresenta il coefficiente
     * del termine di grado i del polinomio */
    // chk: coefficienti!=null &&
    //      l'ultimo elemento di coefficienti
    //      è diverso da zero

    ... costruttori ...

    ... metodi ...
}
```

Polinomio — costruttore

```
/** Crea un nuovo Polinomio,
 *  dato l'array c dei coefficienti. */
public Polinomio(double[] c) {
    // pre: c!=null &&
    //      l'ultimo elemento di c è diverso da zero

    ...
}
```

Il costruttore potrebbe semplicemente assegnare alla variabile d'istanza **coefficienti** il riferimento all'array **c** passato come parametro

- questa soluzione è però soggetta a effetti collaterali non desiderati
  - in particolare, il polinomio verrebbe modificato se venisse cambiato il valore degli elementi dell'array usato come parametro attuale nell'invocazione del costruttore
- è quindi opportuno che il costruttore cloni l'array **c** (ovvero, crei una copia di **c**)

Polinomio — costruttore

```
/** Crea un nuovo Polinomio,
 *  dato l'array c dei coefficienti. */
public Polinomio(double[] c) {
    // pre: c!=null &&
    //      l'ultimo elemento di c è diverso da zero
    int i;    // indice per la scansione di c

    /* crea una copia di c */
    this.coefficienti = new double[c.length];
    for (i=0; i<c.length; i++)
        this.coefficienti[i] = c[i];
}
```

In questo modo, il polinomio creato è insensibile a eventuali variazioni dell'array usato come parametro attuale nell'invocazione del costruttore

Polinomio — int grado()

Il metodo **int grado()** deve calcolare e restituire il grado del polinomio

- il grado del polinomio è dato dalla lunghezza dell'array **coefficienti** diminuita di uno

```
/** Calcola il grado di questo Polinomio. */
public int grado() {
    return coefficienti.length - 1;
}
```

Polinomio — double valore(double x)

Il metodo **double valore(double x)** deve calcolare e restituire il valore del polinomio nel punto **x**

- indicando con n il grado del polinomio e con c_i il coefficiente del termine di grado i, deve calcolare il valore dell'espressione $c_n x^n + c_{n-1} x^{n-1} + \ldots + c_1 x + c_0$
- una prima soluzione può essere ottenuta definendo un metodo ausiliario potenza per il calcolo di potenze intere

```
/** Calcola il valore di questo Polinomio in x. */
public double valore(double x) {
    double v;    // valore del polinomio nel punto x
    int i;       // indice per iterare
                 // sui termini del polinomio

    /* calcola il valore del polinomio nel punto x */
    v = 0;
    for (i=0; i<coefficienti.length; i++)
        v += coefficienti[i]*potenza(x,i);
    return v;
}
```

Polinomio — il metodo potenza

Il metodo **double potenza(double x, int n)** calcola e restituisce il valore di x^n, con **n** naturale

```
/** Calcola il valore di x alla n. */
private static double potenza(double x, int n) {
    // pre: n>=0
    double p;    // x alla n
    int i;       // per iterare fino a n

    /* calcola x alla n */
    p = 1;
    for (i=0; i<n; i++)
        p = p*x;
    return p;
}
```

- il metodo **double potenza(double x, int n)** è evidentemente un metodo di supporto, ed è pertanto stato dichiarato privato
  - alternativamente, potrebbe essere usato il metodo **double pow(double a, double b)** di **Math**

Polinomio — double valore(double x)

La seguente osservazione permette di scrivere una implementazione più efficiente del metodo **valore**

- il calcolo della potenza i-esima avviene contestualmente al calcolo del valore parziale del polinomio limitatamente al termine di grado i

```
/** Calcola il valore di questo Polinomio in x. */
public double valore(double x) {
    double v;    // valore del polinomio nel punto x
    int i;    // per iterare sui termini del polinomio
    double xi;   // x alla i

    v = 0;
    xi = 1;             // ok per i = 0
    for (i=0; i<coefficienti.length; i++) {
        /* xi vale x alla i */
        v += coefficienti[i]*xi;
        xi = xi*x;     // ok per la prossima iterazione
    }
    return v;
}
```

Polinomio — String toString()

Il metodo d'istanza **String toString()** deve calcolare una descrizione del polinomio sotto forma di stringa

- ad esempio, la descrizione del polinomio $p(x) = x^3 + 2x - 1$ può essere la stringa **"+1.0*x^3 +2.0*x -1.0"**
- ipotesi che vengono fatte circa la descrizione del polinomio
  - i termini il cui coefficiente vale zero non devono comparire nella descrizione — tranne per il caso del polinomio $p(x) = 0$, la cui rappresentazione è **"0"**
  - il termine di grado uno non riporta il grado del termine
  - il termine di grado zero è composto solo dal coefficiente
  - per i termini che hanno coefficiente negativo sono preceduti dal solo segno meno

Polinomio — String toString()

```
/** Calcola una descrizione di questo Polinomio. */
public String toString() {
    String d;    // descrizione di questo Polinomio
    int i;       // indice per i termini del polinomio
    double ci;   // il coefficiente i-esimo

    /* calcola la descrizione d di questo polinomio */
    if (this.grado()==0)    // polinomio di grado 0
        d = String.valueOf(coefficienti[0]);
    else {
        /* caso di polinomio di grado maggiore di 0 */
        d = "";
        for (i=grado(); i>=0; i--) {
            ... concatena a d la descrizione del
                termine di grado i ...
        }
    }
    return d;
}
```

Polinomio — String toString()

```
d = "";
for (i=grado(); i>=0; i--) {
    /* concatena a d la descrizione del
       termine di grado i */
    ci = coefficienti[i];
    /* solo termini con coeff. non nullo */
    if (ci!=0) {
        /* concatena + per i coeff. positivi */
        if (ci>0)
            d += "+";
        /* concatena coefficiente */
        d += ci;
        /* concatena il resto del termine */
        if (i>0)
            d += "*x";
        if (i>1)
            d += "^" + i;
    }
}
```

Esercizi

I seguenti esercizi richiedono di modificare e/o estendere la definizione della classe **Polinomio**

- definire il metodo d'istanza **boolean equals(Polinomio p)** — e il metodo **boolean equals(Object p)** — che verificare l'uguaglianza tra questo polinomio e il polinomio **p**
- definire un metodo di classe **somma** che, data una coppia di polinomi **a** e **b**, crea e restituisce un nuovo **Polinomio** uguale alla somma di **a** e **b**
  - si faccia attenzione al caso in cui **a** e **b** hanno grado diverso, e al caso in cui, pur avendo **a** e **b** lo stesso grado, la loro somma ha un grado inferiore
- modificare la definizione del costruttore rilassando il vincolo sull'ultimo elemento del parametro del costruttore
  - se l'ultimo o gli ultimi coefficienti del parametro sono nulli, il costruttore deve creare un array di lunghezza opportunamente minore di quello passato come parametro, in modo che comunque l'ultimo elemento della variabile d'istanza **coefficienti** sia non nullo

Esercizi

Intuitivamente, lo stato di un oggetto che rappresenta una stringa può essere rappresentato mediante un array di caratteri, la cui lunghezza è esattamente la lunghezza della stringa rappresentata

- definire una classe **MyString** che sia una nuova implementazione (di una parte) della classe **String**
- la classe **MyString** deve definire
  - una sola variabile d'istanza **cars** di tipo array di caratteri, per rappresentare i caratteri della stringa
  - un costruttore **MyString(String s)**, che crea un nuovo oggetto **MyString** per rappresentare la stringa **s**
  - un metodo **String toString()**, che restituisce la stringa rappresentata dall'oggetto **MyString**
  - i metodi **int length()**, **char charAt()**, **MyString concat(MyString s)**, **MyString substring(int inizio, int fine)**, **MyString substring(int inizio)**, equivalenti ai metodi corrispondenti della classe **String** (questi metodi devono essere realizzati operando sulla variabile d'istanza **cars**)