NOME/COGNOME	MATRICOLA

Metodologie di Programmazione AA 2008/ 2009
Primo Quiz

Sia data la seguente interfaccia

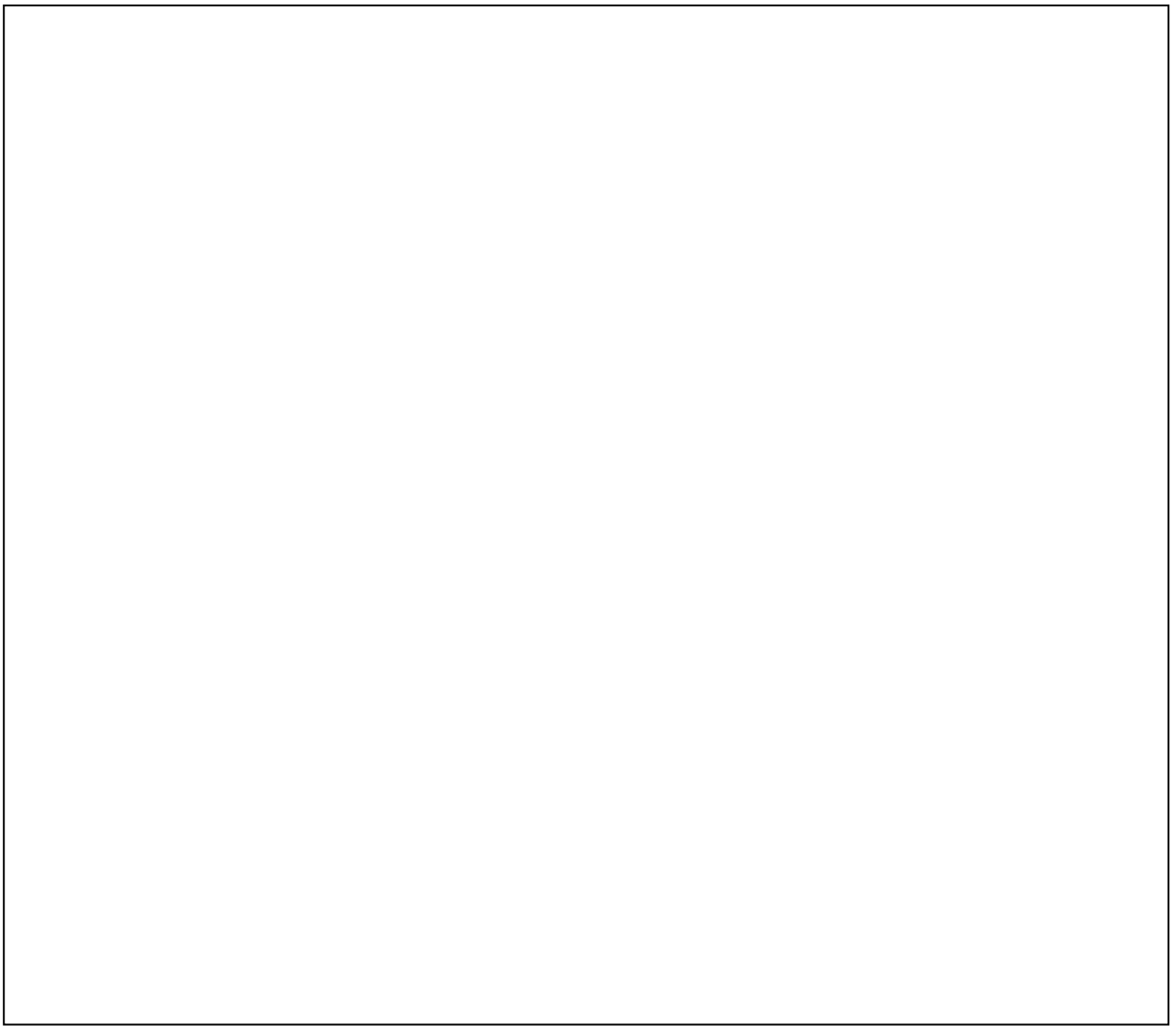
```
/** Un polinomio a coefficienti reali, c0 + c1x + c2x^2 + ... + cnx^n  */
interface Poly {

   /** @result = il grado di this */
    int degree();

   /** @result = il polinomio che rappresenta la derivata prima di this  */
    Poly derivative();
 }
```

Esercizio 1. Completate l' implementazione della classe **`ArrayPoly`** qui di seguito.

```
/**
 * Implementazione di Poly mediante un array di coefficienti
 */
class ArrayPoly implements Poly {

     /** Rappresentazione: coeffs[i] e' il coefficiente del monomio di grado
      *  i. La dimensione dell'array è uguale al grado del polinomio + 1. Ad
      *  esempio il polinomio 3x^2–2x^4 è rappresentato dall'array {0,0,3,0,-2}
      */
      private double[] coeffs;

     /** @param coeffs  è l'array dei coefficienti del polinomio da creare.
      *  La dimensione di questo array  è uguale al grado del polinomio + 1.
      *  Costruisce un ArrayPoly con i coefficienti determinati da coeffs.
      *  Se coeffs è null, restituisce il polinomio che rappresenta 0.
      */
      public ArrayPoly(double[] coeffs)
```

Esercizio 2. Completate l' implementazione della classe **`ListPoly`** qui di seguito

```
/**
 * Implementazione di Poly mediante una lista di monomi
 */
class ListPoly implements Poly  {

     /** Rappresentazione: la lista termini contiene solo monomi con
      *  coefficiente diverso da zero, e al più un monomio per ogni
      *  esponente del polinomio. Ad esempio, il polinomio 3x^2 – 2x^4
      *  è rappresentato dalla lista [(3,2);(-2,4)].
      */
      private static class Monomio {
            double coeff;
            int exp;
            public Monomio(double c, int e) { coeff = c; exp = e; }
      }
      private ArrayList<Monomio> termini;

      /** @param coeffs  è l'array dei coefficienti del polinomio da creare.
      *  La dimensione di questo array è uguale al grado del polinomio + 1.
      *  Costruisce un ListPoly con i coefficienti determinati da coeffs.
      *  Se coeffs è null, restituisce il polinomio che rappresenta 0.
      public ListPoly(double[] coeffs)
```



Esercizio 3. Supponiamo di estendere l'interfaccia **`Poly`** con la seguente specifica:

```
/**
 * @result = un iteratore che fornisce la sequenza di tutti i
 * coefficienti di this  (inclusi i coefficienti nulli)
 */
Iterator<Double> coefficients();
```

Implementate il metodo **`coefficients()`** nella classe **`ArrayPoly`** utilizzando la seguente definizione della interfaccia **`Iterator<T>`**

```
interface Iterator<T> {
        /** true se e solo se esistono altri elementi */
        boolean hasNext();
        /** restituisce l'elemento corrente e avanza.
         *  Se non ci sono più elementi restituisce null */
        T next();
}
```

Esercizio 4. Supponiamo di estendere ulteriormente l'interfaccia **`Poly`** con la seguente specifica

```
/**
 * @result = true se q e this rappresentano lo stesso polinomio
 */
boolean equals (Poly q);
```

Implementate il metodo **`equals()`** nella classe **`ListPoly`** utilizzando il metodo **`coefficients()`** dell'esercizio 3. Il risultato del metodo deve essere indipendente dalla rappresentazione del polinomio, ovvero ad esempio,**`(new ListPoly(c)).equals(new ArrayPoly(c)) = true,`** per ogni **`c:double[]`.**