**Ejercicios sobre la práctica 4**

**Centro**: E. P. S. de Gandía  
**Titulación**: Grado en Ingeniería de Sistemas de Telecomunicación, Sonido e Imagen  
**Asignatura**: Programación 1

**Ejercicios sobre la clase Polinomio**

1. Añadir a la clase Polinomio un método gradoMayor que reciba un Polinomo y devuelva un booleano indicando si el Polinomio actual tiene mayor grado que el Polinomio dado como parámetro.   
   Ejemplo: Dados los Polinomios p con los coeficientes {1, 0, 1, 3}, q con los coeficientes {1, 2} y r con los coeficientes {1, 3, 2, 0, 1}, p.gradoMayor(q) devuelve true, q.gradoMayor(r) devuelve false y p.gradoMayor(r) devuelve false.
2. Añadir a la clase Polinomio un método iguales que reciba un Polinomio y devuelva un booleano indicando si el Polinomio actual y el Polinomio dado como parámetro son iguales (es decir, si tienen los mismos coeficientes).   
   Ejemplo: Dado un Polinomio p con los coeficientes {1, 0, 1, 3} y otro Polinomio p2 con los mismos coeficientes, p.iguales(p2) devuelve true.
3. Añadir a la clase Polinomio un método copia que devuelva un nuevo Polinomio con los mismos coeficientes que el Polinomio actual.   
   Ejemplo: Dado un Polinomio p con los coeficientes {1, 0, 1, 3}, p.copia() devuelve un Polinomio que cuyos coeficientes son {1, 0, 1, 3}.
4. Añadir a la clase Polinomio un método cuantosIguales que reciba un número real y devuelva un booleano indicando cuántos coeficientes del Polinomio son iguales al número dado como parámetro.   
   Ejemplo: Dado un Polinomio p con los coeficientes {1, 0, 1, 3}, p.cuantosIguales(1) devuelve 1.
5. Añadir a la clase Polinomio un método tieneCoeficiente que reciba un número real y devuelva un booleano indicando si alguno de los coeficientes del Polinomio es igual al número dado como parámetro. Implementar el método tieneCoeficiente llamando al método cuantosIguales del ejercicio anterior.   
   Ejemplos: Dado un Polinomio p con los coeficientes {1, 0, 1, 3}, p.tieneCoeficiente(1) devuelve true y p.tieneCoeficiente(2) devuelve false.
6. Añadir a la clase Polinomio un segundo método derivada que reciba un número real y devuelva el valor de la derivada del polinomio actual en dicho punto.   
   Ejemplo: Dado un Polinomio p con los coeficientes {-1, 2, -3}, p.derivada(-1) devuelve 8.
7. Añadir a la clase Polinomio un método integralDefinida que reciba dos números reales a y b y devuelva el valor de la integral del polinomio actual entre a y b.   
   Ejemplo: Dado un Polinomio p con los coeficientes {-1, 2, -3}, p.integralDefinida(0,1) devuelve -1.
8. Añadir a la clase Polinomio un método valorMayorQue que reciba dos números reales r1 y r2 y devuelva un boolean indicando si el valor del Polinomio en r1 es mayor que r2.   
   Ejemplo: Dado un Polinomio p con los coeficientes {1, 2} (para p(x)=1+2x), p.valorMayorQue(1,2) devuelve true, ya que p(1) es 3, que es mayor que 2.
9. Añadir a la clase Polinomio un segundo método grado que reciba un número real y devuelva un ArrayList con los grados de los términos del Polinomio cuyo coeficiente sea igual al parámetro dado.   
   Ejemplos: Dado un Polinomio p con los coeficientes {1, 0, 1, 3}, p.grado(1) devuelve un ArrayList tal que {0,2} (porque los coeficientes de grado 0 y 2 son 1) y p.grado(0) devuelve un ArrayList tal que {1} (porque el coeficiente de grado es 0).
10. Añadir a la clase Polinomio un método entrePolinomios que reciba un número real x y dos Polinomios a y b y devuelva un boolean indicando si el valor del Polinomio actual en x está entre los correspondientes valores de a y b en x.   
    Ejemplo: Dados los Polinomios p con los coeficientes {1, 0, 1, 3}, q con los coeficientes {1, 2} y r con los coeficientes {1, 3, 2, 0, 1}, p.entrePolinomios(1, q, r) devuelve true ya que p(1)==5 está entre q(1)==3 y r(1)==7.
11. **¡Difícil!**   
    Añadir a la clase Polinomio un método aTextoExtendido que devuelva una cadena con el polinomio expresado en función de x.   
    Ejemplos:
    * Para un Polinomio con el coeficiente {1} devuelve "y = 1.0"
    * Para un Polinomio con los coeficientes {1,2} devuelve "y = 1.0 + 2.0\*x"
    * Para un Polinomio con los coeficientes {1,-2,3} devuelve "y = 1.0 - 2.0\*x + 3.0\*x^2"
    * Para un Polinomio con los coeficientes {1,-1,1} devuelve "y = 1.0 - x + x^2"
    * Para un Polinomio con los coeficientes {0,1} devuelve "y = x"
    * Para un Polinomio con los coeficientes {0,2} devuelve "y = 2.0\*x"
    * Para un Polinomio con los coeficientes {0,-1,1} devuelve "y = - 1.0\*x + x^2"
    * Para un Polinomio con los coeficientes {0,2,3} devuelve "y = 2.0\*x + 3.0\*x^2"
    * Para un Polinomio con los coeficientes {0,0,3} devuelve "y = 3.0\*x^2"

**Ejercicios sobre la práctica 4 (segunda parte)**

**Centro**: E. P. S. de Gandía  
**Titulación**: Grado en Ingeniería de Sistemas de Telecomunicación, Sonido e Imagen  
**Asignatura**: Programación 1

**Otras funciones**

1. Crear una clase PolinomioExtendido que permita definir polinomios tales que   
   p(x) = a-mx-m + ... + a-1x-1 + a0 + a1x1 + ... + anxn   
   como por ejemplo:   
   p(x) = 3x-2 + 4x-1 - 5 + x + 3x2 - x3   
   con las siguientes resticciones:
   1. La clase debe implementar la interfaz Funcion:

public class PolinomioExtendido implements Funcion { ... }

* 1. La clase debe *funcionar* con el siguiente fragmento de código:

1. Polinomio p = new Polinomio("-5 1 3 -1");
2. ArrayList l = new ArrayList();
3. l.add(4); // Termino de grado -1
4. l.add(3); // Termino de grado -2
5. PolinomioExtendido ext = new PolinomioExtendido(p, l);
6. int v = ext.valor(2);
7. int gpos = ext.gradoPositivo();
8. int gneg = ext.gradoNegativo();
9. int t = ext.talla();

de manera que:

* + - El valor de v sea el valor del polinomio extendido en el punto dado (en el ejemplo, 2.25)
    - El valor de gpos sea el *grado positivo* del polinomio extendido (en el ejemplo, 3)
    - El valor de gneg sea el *grado negativo* del polinomio extendido (en el ejemplo, -2)
    - El valor de t sea la *talla* del polinomio extendido, es decir, el número de términos que tiene (en el ejemplo, 6)

Se pide:

* 1. A partir del fragmento de código anterior, deducir el diseño de la clase PolinomioExtendido.
  2. Crear la clase PolinomioExtendido.java con las indicaciones anteriores.
  3. Crear un programa PruebaPolExt.java para probar la clase creada:
     + Crear objetos de tipo PolinomioExtendido.
     + Invocar sus métodos y comprobar que los resultados son correctos.
     + Calcular la integral de los polinomios creados, en el intervalo [1,2], con una precisión de 0.001.
     + Dibujar la gráfica de los polinomios creados, en el mismo intervalo con la misma precisión.