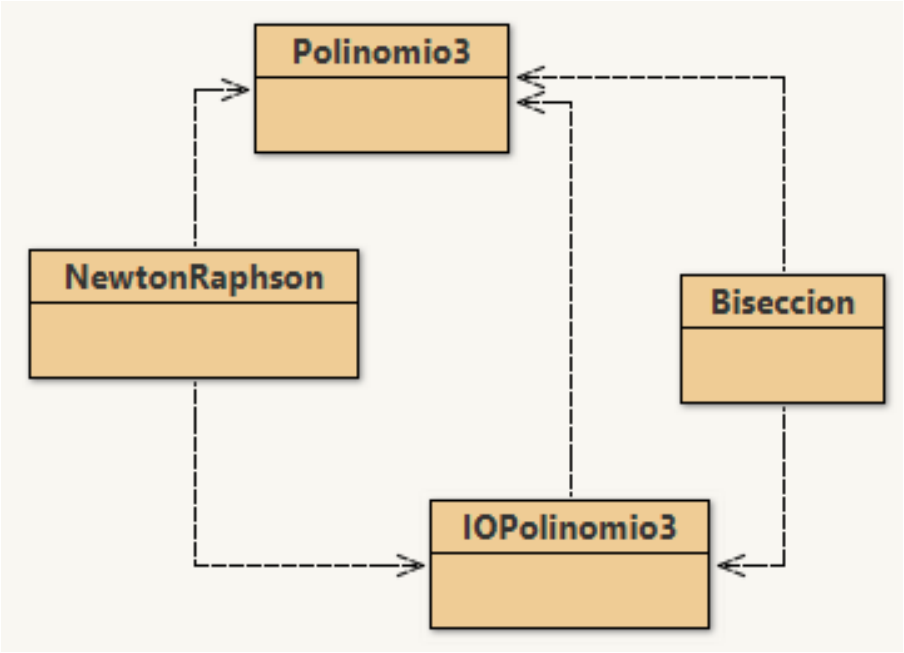
Práctica 7: Raíz de un polinomio de tercer grado

Objetivos: Practicar con la entrada/salida y los bucles

Descripción: Se desea escribir software que permita trabajar con polinomios de orden 3 y calcular un cero de un polinomio usando dos técnicas de cálculo numérico.

El diseño consta de cuatro clases



Clases

- `Polinomio3`: contiene los coeficientes del polinomio y métodos para trabajar con él (*se da hecha*)
- `IOPolinomio3`: con operaciones de entrada/salida para polinomios
- `NewtonRaphson`: *programa principal* que aplica la técnica de Newton-Raphson para hallar una raíz
- `Biseccion`: *programa principal* que aplica la técnica de la bisección para hallar una raíz (es la parte avanzada)

Clase `Polinomio3`

- Atributos para guardar un polinomio de orden 3: `a`, `b`, `c` y `d`

$$ax^3 + bx^2 + cx + d$$

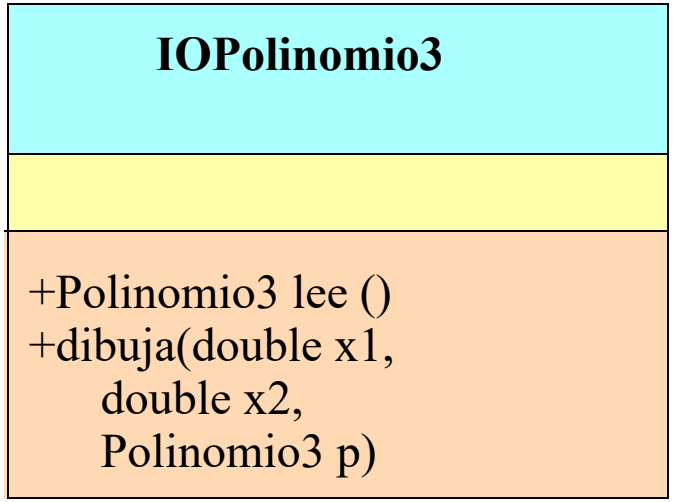
- *Constructor* que recibe como parámetros sus coeficientes
- `valor`(): Método que retorna el valor del polinomio en un punto `x` que se pasa como parámetro
- `derivada`(): Método que retorna el valor de la derivada del polinomio en un punto `x` que se pasa como parámetro

$$3ax^2 + 2bx + c$$

Clase `IOPolinomio3`

`lee`(): lee los cuatro coeficientes de un polinomio de grado 3, crea un objeto de la clase `Polinomio3` con los datos leídos y lo retorna

`dibuja`(): dibuja una gráfica del polinomio `p` con al menos 100 valores de `x` entre `x1` y `x2`



Técnica de Newton-Raphson

La técnica de Newton-Raphson para calcular una raíz real de una función real $f(x)$ consiste en ir obteniendo sucesivos valores x_k, para $k>0$, a partir de un valor inicial x_0, de acuerdo con la siguiente expresión:

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}$$

La técnica de Newton-Raphson no siempre converge

- Debe limitarse el máximo numero de iteraciones a un valor entero configurable
- Si se excede ese límite, se considerará que la técnica no ha convergido

Cuando hay convergencia, la técnica de Newton-Raphson finaliza si $|f(x_k)| < tolerancia$

Programa principal

La clase `NewtonRaphson` contendrá el `main` en el que se aplica la técnica de Newton-Raphson haciendo lo siguiente:

- Obtiene un polinomio leyendo sus valores de teclado con `lee`()
- pide por teclado:
  - el valor inicial a ser usado en Newton-Raphson: `x0`
  - el número máximo de iteraciones
  - el valor de la tolerancia (valor por debajo del que consideraremos, en valor absoluto, que la función vale cero)
- calcula una raíz real de ese polinomio aplicando la técnica de Newton-Raphson
  - de no alcanzarse la convergencia, debe mostrarse un mensaje de error y acabar el programa

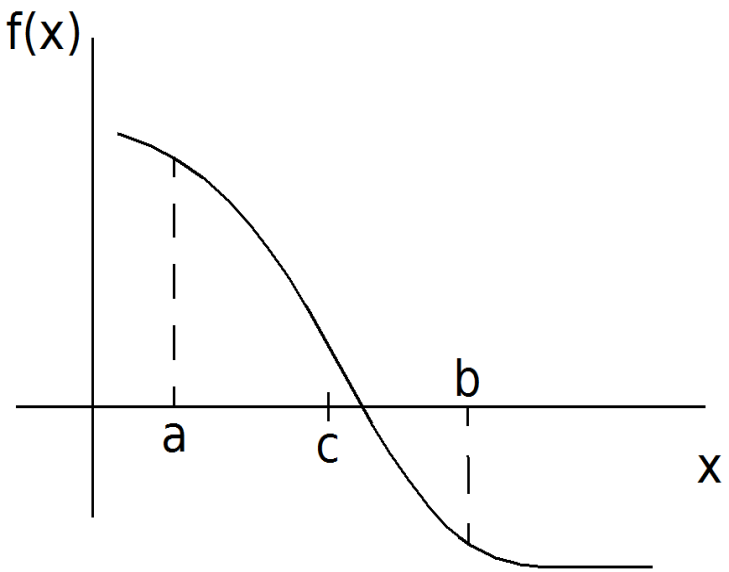
Programa principal (cont.)

- muestra en pantalla:
  - la raíz
  - el valor del polinomio en la raíz
  - el número de iteraciones realizadas para alcanzar la convergencia
- muestra en pantalla una gráfica del polinomio entre los puntos (x-Δ,x+Δ), siendo x la raíz y $\Delta = max(0.1, |0.1*x|)$
  - Nota: $|y|$ es el valor absoluto de y

Parte avanzada

Crear en la clase `Biseccion` otro programa principal que aplique esta técnica para hallar una raíz

La técnica de la *bisección* permite encontrar una raíz de la función *f*(*x*), es decir, un valor *x* que haga *f*(*x*)=0 o cercano a cero. En nuestro caso la función *f*(*x*) es el polinomio



Se parte de dos valores de *x* (*a* y *b*), para los que el signo de *f*(*x*) es diferente (como en la figura)

Si los signos de *f*(*a*) y *f*(*b*) fuesen iguales la técnica de la bisección no se puede aplicar y se deberá mostrar un mensaje de error

Técnica de la bisección

Se comprueba el signo de $f(x)$ en el punto intermedio $c=(a+b)/2$

- Si el signo en c es igual al signo en a (como en la figura), se reemplaza a por c
- En caso contrario se reemplaza b por c

Este proceso se repite hasta que $f(c)$ sea cercano a cero más menos un margen llamado "tolerancia"

En ese momento, el punto intermedio c es la raíz buscada

Si los signos de a y b son correctos y la función es derivable en ese intervalo esta técnica siempre converge

Programa principal de la parte avanzada

- Obtiene un polinomio leyendo sus valores de teclado con `lee`()
- pide por teclado:
  - los límites inferior y superior a ser usados en la bisección: `a` y `b`
  - el valor de la tolerancia (valor por debajo del que consideraremos, en valor absoluto, que la función vale cero)
- si los signos de $f(a)$ y $f(b)$ son iguales se muestra un mensaje de error y el programa acaba
- calcula una raíz real de ese polinomio aplicando la técnica de la bisección, y muestra en pantalla:
  - la raíz
  - el valor del polinomio en la raíz
  - el número de iteraciones realizadas para alcanzar la convergencia
- muestra en pantalla una gráfica del polinomio entre los puntos $(x-\Delta, x+\Delta)$, siendo x la raíz y $\Delta = max(0.1, |0.1*x|)$
  - `|y|` es el valor absoluto de `y`

Entregar

El proyecto Bluej en un archivo comprimido

Informe:

- Parte básica:
  - código de la clase `IOPolinomio3`
  - código del programa principal realizado
  - resultados obtenidos usando los datos que aparecen en el informe:
    - en la consola Java
    - captura de pantalla de la gráfica
- Parte avanzada
  - código del programa principal de la parte avanzada
  - resultados obtenidos usando los datos que aparecen en el informe:
    - en la consola Java
    - captura de pantalla de la gráfica