Tema laborator 3
Creare de clase si obiecte

In acest laborator veti crea o clasa, o veti instantia si veti manipula obiecte ale acestei clase. Veti lucra si cu cu input/output de la tastatura.

Ca aplicatie practica, veti implementa o functie de gradul 2. Clasa voastra trebuie sa returneze valoarea functiei (ex f(x)) si valoarea derivatei (f'(x)) pentru un punct.

Aveti deja implementata o metoda care gaseste radacina unei functii (solutia ecuatiei f(x) = 0).

1 Materiale aditionale ce trebuie studiate

Cititi si intelegeti (de pe : http://users.utcluj.ro/~jim/OOPR/lab-announce.html)
Laborator 2: Tipuri primitive şi IO pe consolă (Doar capitolul 4)
Laborator 4: InstructiuniDeControl+ClaseSimple (Doar capitolul 4)
Laborator 5: Clase, obiecte şi tablouri (fara capitolul 2)

2 Teorie

Conceptul de incapsulare este concretizat prin clase. Intr-o clasa se incapsuleaza datele impreuna cu codul ce manipuleaza aceste date. Obiectele sunt instantieri ale clasei, fiecare obiect avand zona lui de memorie privata.

Clasele sunt ca niste sabloane dupa care se creaza obiectele.

Constructorii sunt niste metode speciale ce se apeleaza imediat dupa ce a fost alocata memoria pentru membri (datele clasei).

Functia de gradul 2 are forma $f(x) = ax^{\wedge 2}+bx+c$. Derivata are forma $f'(x) = 2ax+b$. a,b si c sunt constante si date pentru o anumita functie.

3 Practica

Descarcati si importati proiectul. Vezi laborator 2 pentru detalii despre acest pas.

3.1 Crearea unei clase

Creati o noua clasa numita Func in pachetul ro.utcluj.poo.lab03. Aceasta clasa va implementa ecuatia de gradul 2.

Creati 3 membri de tip double, pentru cele 3 constante ale functiei.

Creati un constructor care sa accepte 3 valori de tip double si sa initializeze constantele functiei.

Creati o metoda publica cu numele getFuncValue, ce accepta un double si returneaza un double. Aceasta functie va returna valoarea lui f(x) in punctul x (x este parametrul de intrare).

Creati o metoda publica cu numele getDerivValue ce accepta un double si returneaza un double. Aceasta functie va returna valoarea lui f'(x) in punctul x (x este parametrul de intrare).

Mare atentie la numele clasei si a metodelor! Trebuie sa fie identic cu cel specificat aici altfel programul nu compileaza.

In acest moment, testele din clasa FuncTest trebuie sa se poata compila si sa treaca.

3.2 Operarea cu obiecte

In metoda main() a clasei Main03:

Comentati codul existent. (Nu il stergeti, va veti inspira din el mai incolo)

Creati o variabila numita f1 de tip Func si creati un obiect pentru aceasta variabila. Acest obiect trebuie sa modeleze functia f(x) = x^2-x-2;

Folositi metoda statica findRoots a clasei RootFinding ca sa gasiti radacina lui f(x). Afisati pe ecran rezultatul. Incercati cu diverse valori de pornire. Ex 3, sau -2.

Creati mai multe variabile, ce sa modeleze mai multe functii. Calculati si afisati pentru fiecare, radacina sau radacinile daca credeti ca sunt mai multe. Creati cel putin 4-5 astfel de obiecte si afisati pentru fiecare. (Pentru cum vizualizati usor functii vezi capitolul Referinte) Observati ce se intampla cand f(x) nu are radacini reale.

3.3 Crearea de metode

La punctul anterior ati scris cod de fiecare data cand ati calculat si afisat radacinile pe ecran. O parte din cod este identic (ex. cel care calculeaza si afiseaza radacina). Acum trebuie sa creati o metoda in Main03 care sa ia un obiect Func, un punct de pornire si sa afiseze pe ecran radacina. Astfel, operatiile repetitive sunt scrise o singura data si doar folosite de mai multe ori.

Optional cititi parametrii functiei de la tastatura (folositi metoda kbdRead)

***AVANSAT

Adaugati o metoda numita toString() la clasa Func, metoda ce genereaza reprezentarea eleganta a functiei (ex: ”f(x)=2x^2-4x+1”). Folositi aceasta functie cand afisati radacinile (exemplu de mesaj „Pentru f(x)=.... radacina este 3”)

Mare atentie! Metoda toString() returneaza un string! NU afiseaza nimic pe ecran!

***AVANSAT

3.4 Extinderea functionalitatii

Metoda lui Newton merge si pe alte tipuri de ecuatii, atata timp cat ele sunt continue si derivabile.

Sa incercam sa gasim radacinile unei ecuatii de gradul 3: f(x)= x^3 -5x^2+3x-3 Radacina ar trebui sa fie pe la 4.5. Sa incercam radacinile unei functii trigonometrice. Sau unei functii mai complexe (cu radicali, logaritmi, etc)

1. Cum ati modela asta in cod? Ati modifica Func? Ati crea o noua clasa?
2. Daca creati o noua clasa, va functiona ea (va compila codul) cu metoda findRoots? Va trebui sa scrieti o noua metoda findRoots2, aproape identica cu prima. Singura diferenta ar fi tipul de intrare.

3. Daca modific clasa Func, nu mai trebuie sa modific metoda findRoots. Dar, daca imi trebuie si ecuatii de grad 2 si de grad 3 in acelasi timp, nu am cum sa fac asta din cod.
4. Ce ar trebui sa faceti ca findRoots sa functioneze pe orice functie? De exemplu sa aveti cateva clase care implementeaza functii (fiecare clasa cu tipul ei de functie) si sa aveti o singura metoda findRoots (Hint: folosind interfete sau supraincarcarea, in laboratoarele viitoare)

4 Evaluare

Tema se considera incheiata cand

1. Ati creat clasa Func conform specificatiilor
2. Ati efectuat operatiile cu obiecte
3. Ati creat metodele de la punctul 3.3
4. Ati incercat sa raspundeti la intrebarile de la 3.4 si ati codat macar o functie de gradul 3 indifierent prin ce metoda.
5. Toate testele sunt verzi.

5 Referinte:

http://en.wikipedia.org/wiki/Newton's_method

Pentru a „desena” grafice, incercati asta: Dati in google search dupa cuvantul plot urmat de ecuatie. Ex:

plot x^3 -5x^2+3x-3

Rezulta: