***Teme:***
*Rekurzivni postopki (utrjevanje)*
*Izbrani postopki*

**Dijaki pri izvajanju izberejo 2 nalogi in jih rešijo.**

Opomba pred začetkom:

(dokument kot vir):

viri/900_Karatsuba_Offman.pdf

(risanje po Canvas-u, okvir za preprosto animacijo):

vsebine/20_JavaFX2/testCanvas_half_demo_doplnjen.zip

## Naloga 1

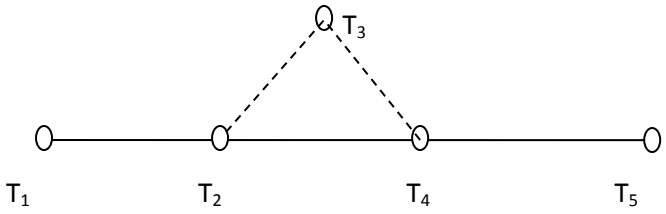
Napišite rekurzivno realizacijo izvedbe postopka množenja velikih števil. Postopek je opisan v relevantnem dokumentu, lahko pa si ga ogledate tudi na internetu. Predpostavite, da boste med seboj množili 2 vsaj 2 mestni števili. Predlagam, da ugotovite, kako velika števila lahko zmnožite z vgrajenimi tipi. Zapis osnovnega števila je konec koncev lahko tudi (zelo dolg) niz.

## Naloga 2

Sestavite javanski program/programček (application/applet), ki bo izrisoval Kochov fraktal, kot je bil predstavljen na učni uri. Koda naj implementira DF postopek (diskusija o teh vrstah postopkov poteka/bo pri urah teorije) : najprej naj izračuna vse potrebne točke in nato izriše zahtevan fraktal.

Izvajanje postopka na vsaki stopnji izvajanja naj poteka kot:

1. daljico razdelimo na tri enako dolge kose,
2. srednji kos razpolovimo in nad njim razpnemo enakostranični trikotnik,
3. ponovimo predhodni dve točki za vsako izmed štirih dobljenih daljic.




v pomoč naj vam bo naslednji kos kode:

```
public void drawFractal (int x1, int y1, int x5, int y5,
                         Graphics g)
   {
      int deltaX, deltaY, x2, y2, x3, y3, x4, y4;

      if ( java.lang.Math.abs(x5 - x1)<7 )
         g.drawLine (x1, y1, x5, y5);
      else
      {
         deltaX = x5 - x1;
         deltaY = y5 - y1;

         x2 = x1 + deltaX / 3;
         y2 = y1 + deltaY / 3;

        double SQ = java.lang.Math.sqrt(3.0) / 6;  // višina

         x3 = (int) ((x1+x5)/2 + SQ * (y1-y5));
         y3 = (int) ((y1+y5)/2 + SQ * (x5-x1));

         x4 = x1 + deltaX * 2/3;
         y4 = y1 + deltaY * 2/3;

         drawFractal (x1, y1, x2, y2, g);
         drawFractal (x2, y2, x3, y3, g);
         drawFractal (x3, y3, x4, y4, g);
         drawFractal (x4, y4, x5, y5, g);
      }
   }
```

Sama realizacija programa/appleta naj bo taka, da bo omogočal izris fraktala določene stopnje. Pri tem naj bo stopnja 1 ravna črta, stopnja 2 kot je narisano na podani sliki, … Z mastnim tiskom v kodi je označen pogoj za končanje rekurzivnega postopka ( distanca med končnima točkama daljice po x-u pod 7 pik/pixlov). Tega bo potrebno spremeniti v npr. stopnja > 1 in dodati glavi metode dodaten parameter 'stopnja' ….

## Naloga 3

Cilj naloge je primerjava postopkov numerične integracije. Pri tem bomo v precep vzeli 3 postopke : aproksimacijo s pravokotniki(kvadratična metoda), aproksimacijo s trapezi(trapezoidna metoda) in Simpsonovo metodo. Poenostavitev simpsonove metode bo podana v nadaljevanju.

Za osnovo določitev ploščin, ki jih oklepajo ovojnice funkcij z abscisno osjo bomo vzeli naslednji dve funkciji:

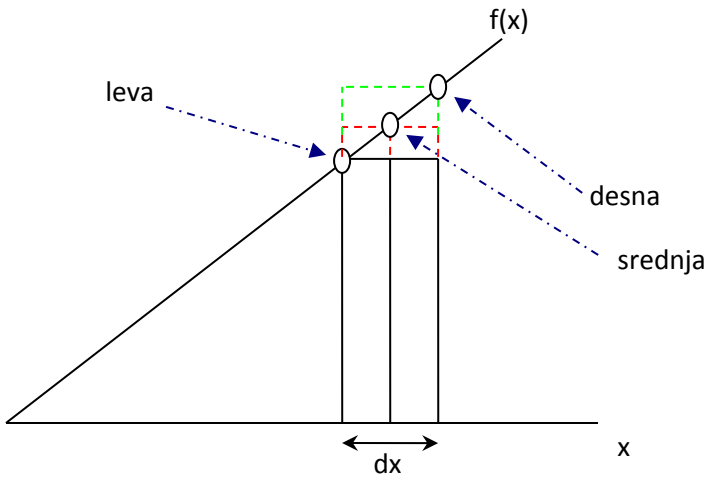
$$f_1(x) = \frac{1}{2}x + 2$$

$$f_2(x) = \frac{1}{6}(x^2 + 5)$$

Določali bomo ploščine na intervalu od 0 do 5. Pri tem si dejansko ploščino prve (linearne) funkcije izračunajte sami; ploščina druge pa se izračuna, kot je podano na spodnjem primeru:

$$pl = \int_0^5 \frac{1}{6}(x^2 + 5)dx = (\frac{1}{2*6}x^3 + \frac{5}{6}x)\Big|_0^5 = \frac{1}{12}*5^3 - \frac{1}{12}*0^3 + \frac{5}{6}*5 - \frac{5}{6}*0 = \mathbf{11.11111}$$

Pri kvadratični metodi ploščine posameznih pravokotnikov lahko računamo na 3 različne načine: levi, srednji, desni; odvisno od tega, katero točko vzamemo pri izračunu višine pravokotnika. V naši nalogi bomo vedno jemali srednjo, ki nam bo omogočila kasneje tudi simpsonovo aproksimacijo:




Na sliki je opaziti, da se ploščine pri kvadratični aproksimaciji razlikujejo glede na to, ali za višino pravokotnika uporabiti levo mejo intervala, desno mejo intervala ali vrednost funkcije na sredini intervala.

Simpsonova metoda

Vrednost ploščine odseka n izbrane širine (dx) se izračuna tako, da vzamete tretjino vrednosti, ki jo dobite kot rezultat izračuna ploščine istega odseka po trapezni metodi in temu prištejete dve tretjini vrednosti ploščine istega odseka izračunanega po kvadratični metodi z uporabo srednje vrednosti (midpoint). Formula:

$$S_n = \frac{1}{3}T(n) + \frac{2}{3}M(n)$$

Rezultat izvajanja programa naj bo primerjalna tabela:

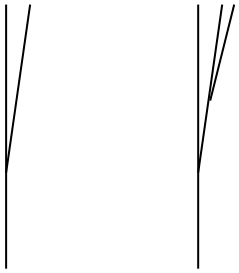
| | Intervalov | Kvadratična(mid) | trapezna | simpsonova | izračunano |
|---|---|---|---|---|---|
| F1 | 1 | | | | |
| | 2 | | | | |
| | 4 | | | | |
| | 10 | | | | |
| | 50 | | | | |
| | 100 | | | | |
| F2 | 1 | | | | 11.11111 |
| | 2 | | | | 11.11111 |
| | 4 | | | | 11.11111 |
| | 10 | | | | 11.11111 |
| | 50 | | | | 11.11111 |
| | 100 | | | | 11.11111 |

## Naloga 4

Recimo, da imamo šahovnico (8x8) polj in šahovsko figurico skakača, s principom premikanja 1,2 in 2,1 (možni skoki skakača). Iščemo sekvenco skokov skakača, iz poljubne izbrane začetne pozicije, ki s skoki pokrije celotno šahovnico, pri tem pa posamezno polje šahovnice obišče zgolj enkrat.

## Naloga 5

Predpostavite, da imate vertikalno črto (pravokotno na osnovnico koordinatnega sistema). Na tretjini višine od osnovnice požene (hm) drevo novo vejo v obliki daljice, od osnovnice proti najvišji točki višine drevesa, vendar je nagnjena od osnovnice v levo za npr. 50. Vsaka veja se deli na isti način. Delitve ni več, ko je veja krajša od 5 pik. Dve iteraciji sta dani:



## Naloga 6

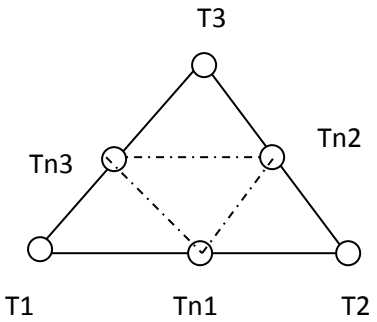
Sestavite javanski program, ki bo izrisoval trikotnik Siepinskega. Stopnjo (št. Korakov) izrisovanja omejite tako, da prenehate izrisovati, ko dolžina najkrajše stranice trikotnika pade pod 5 pik. Dela nimate veliko, zgolj preverite, popravite in izboljšajte predstavljen postopek.

Kot opomba :
Postope izvedete v fazah kot:

1. vsako stranico trikotnika razpolovimo,
2. izrišemo črte med razpolovišči stranic
3. ponovimo postopek za vsakega od nastalih trikotnikov

predlagam metodo : sierpinski(tocka1, tocka2,tocka3); // tockaX predstavlja oglišče trikotnika

mimogrede:

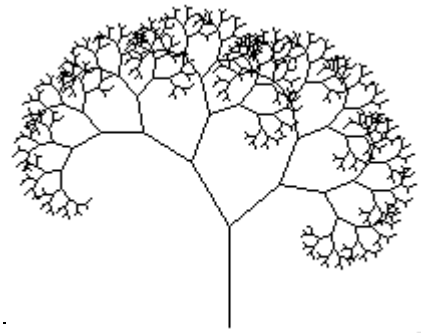



Iz trikotnika (T1,T2,T3) na vsakem koraku nastanejo štirje trikotniki:

- (T1,Tn1,Tn3),
- (Tn1,Tn2,Tn3),
- (Tn1,T2,Tn2),
- (Tn3,Tn2,T3)

## Naloga 7

Naloga je variacija naloge 6. Pri tem je cilj izrisati drevo, kot je podano na spodnji sliki (fraktal, ki izgleda kot drevo). Če ga pogledate malo bolje, vidite, da je drevo sestavljeno iz debla in dveh vej. Vsak izmed vej je dejansko 'enako' drevo, le manjše, pa malo nagnjeno. V bistvu pri implementaciji rišemo veje (črte), torej bomo za izris verjetno spisali metod s tremi parametri : Točko, ki predstavlja prijemališče oz. koordinato veje/debla, dolžino debla in kot, pod katerim se izriše deblo:



npr.:

narisiDrevo(Tocka izhodisceDebla, double dolzinaDebla, double kot);

pri tem je Tocka {double x, double y}

v vsaki iteraciji rišemo levo in desno poddrevo in sicer tako, da kot leve veje vsakič spremenimo za npr. 30o, desno pa za 50o.

Dolžina leve veje naj bo npr. 75%dolžine debla, desna pa 66% dolžine debla.

Npr.: drevo rišemo v sekvenci (pseudo):

```
narisiDrevo(t1, dolzinaDebla * 0.75, kot + 30stopinj);
narisiDrevo(t1, dolzinaDebla * 0.66, kot – 50stopinj);
```

Kreirajte drevo po danih zahtevah

Skušajte kreirati drugačno drevo s spremembo dolžin in kotov

Skušajte narediti drevo nepravilno (s tem, da določene veje ne izrišete, npr.: vejo izrišete z verjetnostjo 1/6.