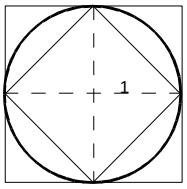
Oblig3Pi- en matematisk rettet obligatorisk oppgave nr. 3 (av 4) i INF1000 høsten 2009 – ett av to alternativer for oblig 3.

Leveringsfrist
Oppgaven må leveres senest fredag 16. oktober kl 16.00. Viktig: les slutten av oppgaven for detaljerte leveringskrav.

Formål
Formålet med denne oppgaven er å gi trening i bruk av klasser og objekter, arrayer og enkel utskrift til fil samt presis programmering av (kjente) matematiske metoder. Oppgaven er spesielt rettet mot matematisk interesserte studenter, og må regnes som noe krevende (som også den andre, alternative oblig3 er).

Oppgave
Oppgaven består i å lage en objektorientert løsning på å regne ut den matematiske konstanten π (3.1415926…) med et brukervalgt antall sifre etter John Machins formel fra 1706.. Oppgavebesvarelsen skal følger kravene nedenfor. Det er da ikke tillatt å bruke andre biblioteker i Java enn easyIO til å løse (deler av) denne oppgaven eller bruke en helt annen formel for Pi. Svaret på oppgaven er to filer, 'Oblig3Pi.java' med programmet, og en annen fil med (minst) 12 000 riktige desimaler i Pi regnet ut av programmet ditt. I tillegg skal du i kommentarer i java-filen vise hvordan du ved bare å forandre under 10 linjer i koden din, kan implementere Størmers formel fra 1896 (som du finner i vedlegget) isteden for Machins formel.

Litt matematisk bakgrunn
Pi (π) defineres som forholdet mellom omkretsen og diameteren på en sirkel. Det er ikke mulig å representere π eksakt med et endelig antall sifre eller som en endelig brøk. Det vil derfor alltid være 'flere sifre' i π . De gamle grekere og andre kulturfolk prøvde å finne en god nok verdi for π ved å beregne omkretsen av regulære mangekanter (firkanter, femkanter …, 48-kanter,..) som er innskrevet og omskrevet en sirkel med radus=1. Verdien av π vil da ligge mellom verdien av forholdet mellom diameteren 2 og hhv. den omskrevne og den innskrevne mangekanten.. I figuren til høyre ser at den omskrevne firkanten har lengde 8 og den innskrevne har lengde 4* √2 . Vi får da at (4* √2)/2 < π < 8/2 , eller



2.828 < π < 4. Med 48-kanter fikk Archimedes følgende resultat: $3\frac{10}{71} < \pi < 3\frac{1}{7}$, eller 3.1407..< π < 3.1428.. Dette ser vi er en ganske dårlig fastsetting av π og selv om vi kan forbedre dette vesentlig med å bruke mangekanter med enda flere hjørner, blir beregningene vanskelige og tidkrevende.

En annen angrepsvinkel er å se på tangens-funksjonen og den inverse funksjonen arctan. Måler vi vinklene i radianer (hvor da 45^o = π/4 radian) så vet vi at tan(π/4) = 1 (se figuren ovenfor som har 4 slike trekanter med 90^0, 45^0 og 45^0), og følgelig er da arctan(1) = π/4.
I 1706 fant John Machin en formel som vi skal bruke i denne oppgaven basert på å erstatte den ene vinkelen π/4 i formler for tan(x) med flere små vinkler:

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

Denne formelen har i stor grad blitt brukt til beregning av π fra slutten av 1700-tallet og frem til slutten av 1950-tallet både for håndregning og senere med datamaskin. John Machin satte selv rekord ved å beregne 100 sifre i π med denne formelen. Nå nyttes det langt mer avanserte formler, som nok er langt raskere, men

også lang mer kompliserte å skrive program for (en egen bok om π: –"J. Arndt & C. Haenel: pi – unleashed" har hele 154 ulike formler, hvorav én er Machins)

Den største ulempen med Machins formel er at tiden det tar å regne ut N sifre er proporsjonal med N^2 - dvs. at 10-dobler man antallet sifre man vil regne ut, så 100-dobles regnetiden. Moderne metoder har en kjøretid som er N*logN. En tidobling av antall sifre vil da bare bruke 23 ganger så lang tid. Machins formel ble siste gang brukt til å sette en rekord med 16 167 sifre i 1959 på en datamaskin. Dagens rekord er på 1.24 billioner sifre, men så sent som i 1973 var rekorden mindre enn 1 million sifre. Det programmet dere skal skrive i denne oppgaven vil nok kunne slå rekorden fra 1959 på under 30 sekunder, og vil 'bare' bruke noen få dager på å slå 1973-rekorden, men ville være helt ubrukelig til å slå dagens rekord (det ville anslagsvis ta 11 000 år).

Beregning av arctan-funksjonen

Vi ser av Machins formel at beregningen av π består i å regne ut arctan for to ulike argumenter med tilstrekkelig antall sifre, samt multiplisere disse med hhv. 16 og 4, og så trekke disse to rekken av sifre fra hverandre. Vi setter inn rekkeutviklingen for arctan-funksjonen i formelen og får dette uttrykket:

$$\pi = 16\arctan\frac{1}{5} - 4\arctan\frac{1}{239} = 16\left[\frac{1}{5} - \frac{1}{3*5^3} + \frac{1}{5*5^5} - \frac{1}{7*5^7} + \ldots\right] - 4\left[\frac{1}{239} - \frac{1}{3*239^3} + \frac{1}{5*239^5} - \frac{1}{7*239^7} + \ldots\right]$$

Hvis vi navngir leddene i rekkeutviklingen for arctan(1/5) som $L5_i$ (i= 0, 1,2,..), ser vi at :

$$L5_0 = \frac{1}{5}, \quad og \quad L5_i = -L5_{i-1}\frac{2i-1}{(2i+1)*5^2} \quad for \quad i > 0$$

og likeledes for arctan(1/239) hvis vi kaller leddene $L239_i$ (i= 0, 1,2,..):

$$L239_0 = \frac{1}{239}, \quad og \quad L239_i = -L239_{i-1}\frac{2i-1}{(2i+1)*239^2} \quad for \quad i > 0$$

Vi ser at har vi bare laget det første leddet i en arctan(1/x)-serie med tilstrekkelige antall sifre, kan vi finne neste ledd bare med å multiplisere leddene med (2i-1) og dele på $(2i+1)*x^2$ (Rekken for arctan(1/5) konvergerer med omlag 1.4 desimale sifre per ledd, mens arctan(1/239) konvergerer med omlag 4.7 desimale sifre per ledd).

Hvordan regne med vilkårlig antall sifre i en datamaskin

Vi kan ikke nytte flyttall (double) til å oppbevare et vilkårlig antall sifre. Istedenfor regner vi bare med en array av heltall, hvor vi har et visst antall desimale sifre lagret i hvert array-element. I denne oppgaven velger vi å ha 4 desimale sifre lagret i hvert heltall. En av grunnene til å velge 4 , er at det fortsatt er plass i en int til resultatet av at to slike tall multipliseres med hverandre hvis ingen av tallene er større enn 4 sifre.

Det som oppgaven går ut på er i hovedsak å programmere er en relativt enkel generalisering av de regnereglene vi lærte på barneskolen om hvordan man multipliserer, dividerer, adderer og subtraherer. Det vi må huske etter å ha foretatt en slik regneoperasjon er at etterpå må intet array-element ha mer enn 4 sifre (ikke være større enn 9999). Det vi da må huske på ved addisjon og multiplikasjon, er å overføre det som er mer enn 4 sifre som 'mente' til neste siffergruppe. Likeledes må vi også 'å låne' fra forrige siffergruppe hvis den siffergruppen vi skal trekke fra er større enn den vi trekker den fra (forskjellen blir selvsagt at nå låner vi 10000, og ikke 'bare' 10 når vi låner fra forrige 'siffergruppe'.) Vi kan se på to eksempler på slik regning med to-og-to sifre (dere skal ut fra disse eksemplene selv implementere fire-og-fire sifre):

b) Addisjon to-og-to (a = a + b) :

```
mente ->   1    1    1
          32   87   77   94     a
     +    01   44   22   61     b
     --------------------
     =    34   32   00   55     ny verdi av a
```

Ved addisjon kan vi lagre svaret på a+b rett i a etter hvert som vi adderer oss fra høyre mot venstre hvis vi ikke skal bruke den gamle verdien til a noe annet sted.

c) Multiplikasjon av et langt tall (a) med et 'enkelt' heltall (k) (a = a*k) :

I oppgaven vi skal løse, ser vi at vi ikke trenger å multiplisere to slike lange tall med hverandre (det er mulig, men trenges altså ikke her). Det vi skal gjøre er gange et mangesifret (12000 stk) med et enkelt heltall (og så dividere med et annet heltall for å få neste ledd i rekka for hhv arctan(1/5) og arctan(1/239).

Vi tar da enkelt , bakfra (fra høyre mot venstre) og multipliserer hver siffergruppe med hele det 'lille' tallet (k, her lik 174). Vi eksemplifiserer igjen regneprosessen med to og to sifre i hver gruppe – du må selv så generalisere dette til 4-og-4 sifre.

```
                    a      *    k

               87   27   94   *    174
  ------------------------------------
  =     1   51   86   63   56
```

Forklaring til svaret 94*174 = 16356, dvs. 56 som siste siffergruppe og 165 som mente. Neste siffergruppe blir 27* 174+165= 4698 + 165 = 4863. Det gir 63 som neste siffergruppe og 48 som mente,…osv.

Vi ser at vi kan lagre svarene etter hvert som vi regner dem ut direkte i a, fordi vi ikke må ha den 'gamle verdien av a' for å regne ut neste siffergruppe i multipliseringen.

Subtraksjon går omlag som addisjon, men husk regelen fra barneskolen at vi kan ikke låne fra en gruppe som er 0, Da må vi låne fra gruppen til venstre for denne igjen (evt. osv.) , legge fra oss 99 der det tidligere var 0, og så overføre 100 som lån til den siffergruppa som var mindre enn det som skulle trekkes fra..

N.B. I oppgaven får du bare bruk for multiplikasjon og divisjon mellom ett mangesifret tall og en enkel `int` (eller `long`) , mens addisjon og subtraksjon går mellom to mangesifrete tall.. Det gjør oppgaven lettere, men mange av betraktningene ovenfor med mente osv. er fortsatt gyldige.

Divisjon går faktisk lettest av alle regneartene, men du må internt i divisjonsmetoden regne med `long` – variable fordi mellomresultatene lett kan overstiger maksimal verdi for en `int`.

Krav til den objektorienterte løsningen

Løsningen ***skal*** bestå av 3 klasser: Oblig3Pi, Pi og Arctan.

Oblig3Pi mottar to parametere fra kommandolinja: <n =ant. sifre som skal regnes ut> og <res=navn på fil for resultatet> og gir feilmelding hvis parameterne ikke er riktige. Hvis OK parametere, lager den et objekt av klassen Pi med de to parameterne. Deretter kaller den print-metoden i Pi-objektet for å få skrevet ut resultatet.

Klassen Pi har en heltalls-array som er lang nok til å holde (minst) n sifre i π (se tipsene). I tillegg inneholder klassen Pi en print-metode for å skrive ut svaret både på skjerm og til resultatfilen, og det er denne metoden som blir kalt fra **`main`** i Oblig3Pi. I tillegg inneholder klassen Pi en add-metode og en sub-metode. Add-metoden har en peker til et objekt av klassen Arctan for å legge verdien av den arctan-rekken til arrayen som holder verdien av π i klassen Pi. Sub-metoden har også peker til et arctan-objekt som parameter og trekker fra verdien av arctan-rekken fra den hittil utregnete π-verdien.

Klassen Arctan har tre parametere i konstruktøren: n (antall sifre) , verdi (som er den verdien det skal tas arctan av – i vårt tilfelle 1/5 og 1/239) og mult (den faktoren som evt. står foran arctan i formelen. I vårt tilfelle er det 16 og 4). I tillegg inneholder klassen metoder som helt nøyaktig implementerer addisjon, subtraksjon, multiplikasjon og divisjon i siffergrupper på 4 (etter de reglene vi lærte på barneskolen). Addisjon og subtraksjon går da mellom to lange tall, mens multiplikasjon og divisjon har ett langt tall og ett vanlig heltall. Konstruktøren skal motta parametrene og så utføre beregningene av hele arctan-rekka (dvs. lage ledd etter ledd i rekka med tilstrekkelig antall sifre og så summere/subtrahere disse til en array som holder verdien av hele rekka.) Dette gjør vi helt til det neste leddet vi lager er = 0 i alle de sifrene vi regner ut). Til sist multiplisert med 'mult', slik at når konstruktøren er ferdig, ligger svaret i en heltalls-array i arctan-objektet med (minst) n sifre. Klassen Arctan inneholder selvsagt, slik de andre klassene, andre metoder og variable du trenger for beregningene.

Det overstående skulle være nok til å løse oppgaven, men vi anbefaler å se på tipsene nedenfor.

Tips

1. Du kan regne med at programmet ditt ikke behøver å bli over 250 linjer med Java-kode.
2. **Fasit:** De første sifrene er: 3.1415 9265 3589 7932 3846 2643 3832 7950 2884 1971 og på plass 761 etter komma finner vi: 4999 9998 3729 (merk seks 9-tall på rad) og de siste sifrene er (tallene i parentes er hvilket siffernummer det tallet som kommer etter parentesen har:
   (11961) 3404 6564 7588 0405 1380 8016 3363 8874 2163 7140
   (12001) 6435 4955 6186; og som nevnt i tips 6, kan de aller siste sifrene være gale.
3. Du vet at rekkene du skal regne ut konvergerer og at hvert ledd er mindre enn det foregående og at svaret bare har ett siffer foran komma (slik at det blir en del tester du slipper å gjøre i koden).
4. Det lønner seg å regne ut hele tiden den positive verdien av hvert ledd i arctan-rekkene og så hhv. trekke fra og legge denne til arrayen som holder summen av rekka.
5. Siden en arctan-rekke skal ha et positivt ledd etterfulgt av at vi trekker fra neste ledd, kan det lønne seg i metoden som beregner hele rekka i en løkke, å gjøre to kall på metoden som regner ut neste ledd – hvor vi legger til det første og så trekker fra det neste leddet.
6. Du bør av regne med litt flere sifre, f.eks. 4 ekstra, enn det kravet oppgaven stiller (12 000) fordi de siste sifrene lett blir unøyaktige (du vet ikke om alle de små bidragene fra de leddene utenfor det siste leddet du regner ut, ville gi ett tillegg i det siste, og evt. også det nest siste sifferet,...)
7. Det kan lønne seg, når du regner ut neste ledd, å foreta multiplikasjonen før divisjonen (gir mindre feil i de siste sifrene).
8. Du bør i tillegg gjøre arrayene dine 1-2 plasser lengre i hver ende enn det som skal til for å representere de ca.12004 sifrene du regner på. Dvs. at du plasserer 3 (heltallsdelen av pi) i plass nr. 1 og ikke nummer 0 i arrayene dine. Dette fordi når du regner på en array 'a', så vil du ofte referere til a[i-1] og a[i+1] uten at du hele tiden ønsker å sjekke om du har kommet utenfor arrayen.
9. Det lønner seg å grundig å skrive ned reglene for de fire regneartene (regnes det fra venstre mot høyre eller høyre mot venstre; hvordan og når overføres mente og må resultatet i en

siffergruppe 'normaliseres' etter utregningen,...). Her vil du få bruk for % og /-operatorene (rest og heltallsdivisjon).

10. Hvis du skriver ut med easyIO i siffergrupper med fire tall og med en blank mellom (f. eks 10 slike siffergrupper per linje), vil du oppleve at eventuelle null først i en slik gruppe ikke skrives ut. Hvis du først omgjør tallene til en String med følgene metode, får du også med 0-er først i siffergruppa som så kan skrives ut med metoden for å skrive ut en String i easyIO:

```
String lagString(int a) {
      String s = ""+a;
      while ( s.length() < 4)
         s= "0" + s;
      return s;
  }
```

11. Det vil halvere regnetiden hvis du passer på hvor langt ned i arrayen leddene i den arctan-rekken du nå regner ut alle har blitt null. Når du så begynner beregning av neste ledd, begynner du på det stedet fordi du vet at neste ledd er mindre. Husk at ved multiplikasjon vil du kunne 'heve' denne null-grensen (midlertidig).

Vedlegg – Størmers formel fra 1896

Når du har kodet en objektorientert løsning på Machins formel, så skal det bare minimale endringer (4-10 linjer) i programmet ditt for at du skal kunne regne ut π etter følgende formel funnet av den norske matematikeren Carl Størmer (den er ca. 30% raskere enn Machins formel). Hvis du får denne formelen til å gi samme svar, vet du at du med meget stor sannsynlighet har skrevet et riktig program:

$$\pi = 4\left(44\arctan\frac{1}{57} + 7\arctan\frac{1}{239} - 12\arctan\frac{1}{682} + 24\arctan\frac{1}{12943}\right)$$

Du skal, som nevnt ovenfor, i alle fall kommentere hvordan denne formelen kan regnes ut av ditt program med minimale endringer.

Leveringskrav

Hver student skal levere sin egenprogrammerte løsning, og plikter å ha lest og forstått følgende krav til innleverte oppgaver ved Institutt for informatikk:

http://www.ifi.uio.no/studinf/skjemaer/erklaring.pdf

Send programfilen (.java-filen) og resultatet av en testkjøring som er en fil med de (minst) 12000 sifrene i π som vedlegg når du leverer via Joly-systemet). Hele programmet ditt (.java-filen) skal være **på én fil,** og testkjøringen på en annen fil**.**

http://obelix.ifi.uio.no:8080/wizard.html

N.B. Husk at Joly-systemet bare virker på Blindern-maskinene (og hjemmefra hvis du har VPN eller Remote Desktop i Windows) Sender du inn innleveringen din fra en epostkonto fra yahoo eller hotmail, er det en stor sjanse at spamfilteret på Ifi tar besvarelsen og den vil aldri nå hjelpelærerer. Send derfor helst besvarelsen din fra den e-postkontoen du har fått her på Universitetet. Du er ansvarlig for at besvarelsen din blir levert!
(Virker ikke Joly, leverer du ved å sende inn som vedlegg til e-post til gruppelæreren din).

Tilbakemelding blir gitt innen to uker etter innleveringsfristen (og vanligvis litt raskere). Dersom besvarelsen vurderes til *ikke godkjent*, vil du få en begrunnelse, og hvis retteren anser at besvarelsen har potensiale til å kunne forbedres tilstrekkelig til å bli godkjent, vil du normalt få anledning til dette, og vil i så fall få en oversikt over hva som må forbedres. Fristen for dette vil normalt være 5 arbeidsdager. Oppgaven skal leveres elektronisk.

Hvis du har spørsmål vedrørende leveringsmåte eller annet, så kontakt gruppelæreren din i god tid før innleveringsfristen. Det er ditt ansvar at oppgaven kommer frem til øvingslæreren på riktig måte innen leveringsfristen.

Lykke til!