TDDC77s Datorbaserad Tentamen

2014-12-03 kl 08:00-12.00

SU10, SU11, SU12 och SU13

- Det är inte tillåtet att ha telefoner, böcker, icke tomma papper eller annat hjälpmaterial till hands under tentamen.
- Det är tillåtet att ha lösa tomma papper och pennor.
- Frågor om uppgifterna svaras endast via kommunikationsfönstret.
- Räck upp handen för att få hjälp om något inte fungerar i systemet
- För betyg 3 krävs godkänt på en av uppgifterna. För betyg 4 krävs två godkända uppgifter. För betyg 5 krävs tre godkända uppgifter.
- Det spelar **ingen roll** i vilken ordning du löser uppgifterna. Börja med den som verkar enklast för dig. Fortsätt med nästa uppgift så fort du har skickat en lösning.

Rättning:

- Steg 1: Vi kompilerar och provkör ditt program. Du får komplettering eller underkänt (se steg 2) om ditt program inte går att kompilera eller om det inte klarar de givna testfallen.
- Steg 2: Vi tittar på programkoden. Programmet kan bli godkänt (isf går vi till steg 3). Annars:
  - Ditt program får komplettering. E.g. vi har några testfall som programmet inte klarar, eller programmet uppfyller inte vissa krav. Du får kommentarer och möjligheten, om tiden tillåter det, att skicka ett nytt program.
  - Ditt program blir underkänt: programmet går inte att kompilera eller klarar inte de testfall som kommer med uppgiften. Du får inte möjlighet att skicka komplettering för den uppgiften. Du får fortsätta med nästa uppgift.
- Steg 3: Om uppgiften blev godkänd uppdateras strax ditt betyg.

1 Map

Skriv ett program som läser in sina argument från terminalen:

- Man anropar programmet med "`java Map a1 a2 ... an op b`", där:
  - `a1 ... an` (där `n` kan vara noll) samt `b` är heltal
  - `op` är en av `+` eller `-`.
- Programmet ska hantera alla situationer där:
  - det förväntas ett heltal men användaren skriver in något annat, eller
  - det förväntas en `+` eller en `-` men användaren skriver in något annat

  I dessa fall ska programmet inte krascha utan terminera (utan errors) efter att ha skrivit ut:

  ```
  Syntax: Map a1 ... an op b
  With  : a1 ... an and b are integers and op in {+,-}
  E.g.  : Map 1 2 3 + 4
  ```

- Du ska använda dig av följande metod i klassen `Integer`:
  `public static int parseInt(String s) throws NumberFormatException`
- Antar att programmet får `a1 ... an op b` som argument. Programmet ska då räkna upp `n` heltal `c1, ... cn` där `ci=(ai op b)` för varje `i` mellan `1 och n`.
- Exempel på möjliga körningar:

  ```
  $ java Map  + 1


  $ java Map 1 2 3 4 - 2
  -1 0 1 2

  $ java Map 1 -2 -3 4 + 3
  4 1 0 7

  $ java Map 1
  Syntax: Map a1 ... an op b
  With  : a1 ... an and b are integers and op in {+,-}
  E.g.  : Map 1 2 3 + 4

  $ java Map 1 +
  Syntax: Map a1 ... an op b
  With  : a1 ... an and b are integers and op in {+,-}
  E.g.  : Map 1 2 3 + 4
  ```

2 Functions

Nedan följer ett antal faktapunkter. Din uppgift är att skapa den specificerade klasshierarki. Ange så pass mycket, MEN INTE MER, av alla klasser, gränssnitt (interfaces), variabler, metoddeklarationer och implementationer så att allt som är specificerat i uppgiften implementeras. Din kod skall vara väl dokumenterad. Den ska dessutom respektera inkapsling och använda sig av lämpliga namn som tillämpar Java konventioner. Kopiera filen `FunctionsDriver.java` som finns under `given_files/` till en ny mapp under din hemkatalog. Skapa och lägg till de Java-filer som behövs för din lösning. Man ska kunna kompilera dina filer och köra `FunctionsDriver`. Skicka alla Java-filer. Du ska använda dig av `Math.pow( )` från API:et (`http://www.ida.liu.se/~TDDC77/extra/api-7/index.html` med chrome). Observera att derivaten av en polynom $p(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x^1 + a_nx^0$ är polynomen $a_0nx^{n-1} + a_1(n-1)x^{n-2} + \ldots + a_{n-1}x^0$

Specifikationer:

- En `Function` är en klass som har en abstrakt metod `int compute(int x)`. Klassen har också en icke-abstrakt metod `boolean isFixpoint(int x)` som returnerar `(compute(x) == x)`.
- En `Differentiable` är ett gränssnitt (interface) men en unik metod: `int differentiate(int x)`.
- En `Polynom` är en klass som ärver från `Function` och som implementerar `Differentiable`. Klassen har en Array av `int` som representerar polynomens koefficienter; E.g., `[1, 2]` representerar polynomet $1x^1 + 2x^0$ och `[4,-1,0]` representerar polynomet $4x^2 - 1x^1 + 0x^0$. Klassen har också en konstruktor `Polynom(int[] koeff)`.
- En `Delta` är en klass som ärver från `Function` men som implementerar inte `Differentiable`. Klassen:s compute metod returnerar 1 om argumentet är 0 och 0 annars.
- `Polynom` och `Delta` ska var ha en metod `public String toString()` som åsidosättar (overrides) Object:s `toString` metod. De nya metoderna ska returnera en `String` som innehåller:
  - `a0 x^n + ... + an x^0` där `a0 ... a1 an` är polynomkoefficienter för polynomen, och
  - `1 om x är 0 och 0 annars`.

3 Livets spel

Kopiera filen `given_files/Life.java` till en ny mapp under din hemkatalog. Vi ska implementera ett känt spel som modellerar olika biologiska processer. Spelet heter "Life". Det är en simulering som sker på ett tvådimensionellt brädde. Enkla regler bestämmer om när celler föds eller dö under simuleringen. Simulering börjar från en generation som är den enda input till simuleringen. Generationen beskriver vilka celler är levande i början (en sån initialisering finns redan i metoden `main`. Ni ska inte ändra på den). Sen räknar simuleringen upp en ny generation från den nuvarande generationen. Läs först filen `Life.java` för information om uppdateringen av generationer. Komplettera den givna koden enligt de anvisningar som finns i form av kommentarer märkta med "TODO: ...". Koden skall skrivas enligt Javas kodkonventioner och kommenteras väl. Koden ska kunna köras. En körning kan se ut så här:

```
$ java Life
.       x   x
.     x
.     x
.     x     x
.     x x x
.
.
.
. . . . . .
print next generation? y/n: y
.
.     x x
.   x x x
.   x x   x
.     x x x
.       x
.
.
. . . . . .
print next generation? y/n: y
.
.   x   x
.         x
.         x
.   x     x
.     x x x
.
. . . . . .
print next generation? y/n: n
good bye!
```