TDDC77s Datorbaserad Tentamen

2015-01-12 kl 14:00-18.00

- Läraren: Ahmed Rezine (+46 (0)13 28 1938 eller +46 (0)72 203 1978).
- Kursassistanter: Andreas Runfalk, Anton Amlinger, Hannah Börjesson, Henrik Laurentz.
- 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 uppgifeterna ställs till de kursassisstenter när de kommer förbi eller till läraren via telefon.
- 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. Du kommer inte att få live feedback om dina lösningar under tentan.
- API:et är under `http://www.ida.liu.se/~TDDC77/extra/api-7/index.html`.

Rättning:

- Steg 1: Vi kompilerar och provkör ditt program. Du får underkänt koden om ditt program inte går att kompilera eller om det inte klarar de givna testfallen.
- Steg 2: Vi tittar på programkoden. Ditt program blir underkänt om den inte uppfyller vissa krav. Ditt program blir annars godkänt.

1 Fold

Skriv ett program som läser in sina argument från terminalen:

- Man anropar programmet med "`java Fold a1 a2 ... an op b`", där:
  - varje `a1 ... an` samt `b` är heltal. `n` kan vara noll.
  - `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: Fold a1 ... an op b
  With  : a1 ... an and b are integers and op in {+,-}
  E.g.  : Fold 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 och skriva ut värdet `(..((b op a1) op a2) ... op an)`.
- Exempel på möjliga körningar:

  ```
  $ java Fold  + 1
  1

  $ java Fold 1 2 3 4 - 2
  -8

  $ java Fold 1 -2 -3 4 - 3
  3

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

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

2 MyDeque

Nedan följer ett antal faktapunkter. 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 filer `MyDequeDriver.java` och `MyDequeDriver.output.txt` 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 få, när man kör `java MyDequeDriver` från terminalen, samma utmatning som innehållet av `MyDequeDriver.output.txt`. Skicka alla Java-filer. Du ska INTE använda dig av `Collections` (e.g., `Deque`, `list`, `Queue`, `ArrayDeque`, etc).

Specifikationer:

- En `Queue` är ett gränssnitt (interface) som har tre metoder: `boolean isEmpty()`, `String dequeue()`, och `void enqueue(String item)`.
- En `Stack` är ett gränssnitt (interface) som har tre metoder: `String pop()`, `void push(String item)`, och `boolean isEmpty()`.
- `MyDeque` är en klass som implementerar båda `Queue` och `Stack`. Den har fyra instans variabler: en array av String värden `content` samt tre `int` variabler `size`, `tail` och `head`.
- Värdet av `size` ges vid instansiering. `content` instansieras då till en array av storlek `size` medan båda `head` och `tail` instansieras till värdet `0`.
- Elementena sparas i arrayen mellan `tail` (inklusive) och `head` (exklusive). Se Fig. 1.
- `isEmpty` metoden returnerar `true` precis när `tail==head`.
- `push` gör ingenting om `((head+1)%size)==tail`. Annars sparar den sitt argument i `content[head]` innan den tilldelar `(head+1)%size` till `head`.
- `enqueue` gör ingenting om `((tail+size-1)%size)==head`, annars tilldelar den `(tail+size-1)%size` till `tail` och sparar sitt argument i `content[tail]`.
- `pop` returnerar `null` om `tail==head`. Den tilldelar annars värdet `(head+size-1)%size` till `head` och sen returnerar `content[head]`.
- `dequeue` returnerar `null` om `head==tail`. Den sparar annars `content[tail]` i en lokal variabel, den tilldelar `(tail+1)%size` till `tail`, och den returnerar värdet av den lokala variabeln.
- `MyDeque` har en metod `public String toString()` som åsidosättar (overrides) Object:s `toString` metoden. Den nya metoden ska returnera en `String` som innehåller alla element från `tail` till (men utan) `head`. Metoden returnerar `< content[tail] content[(tail+1)%size]... >`.

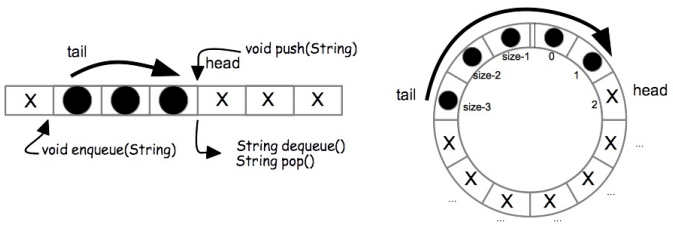

Figure 1: Representation av en MyDeque. Elementena sparas mellan `tail` (inklusive) och `head` (exklusive)

3 Tripp trapp trull

Kopiera filen `given_files/TrippTrappTrull.java` till en ny mapp under din hemkatalog. Vi ska implementera "tripp trapp trupp" spelet. Två spelare, "x" och "o", väljer i turordning positioner i ett `3*3` stor brädde. Den första som lyckas med att ta tre positioner i ett horisontalt, vertikalt eller diagonalt linje vinner. 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 skulle kunna se ut så här:

```
$ java TrippTrappTrull
1 . . .
2 . . .
3 . . .
  1 2 3

player o chooses a free position:
row: enter an integer between 1 and 3:1
column:enter an integer between 1 and 3:1
1 o . .
2 . . .
3 . . .
  1 2 3

player x chooses a free position:
row: enter an integer between 1 and 3:2
column:enter an integer between 1 and 3:1
1 o . .
2 x . .
3 . . .
  1 2 3

player o chooses a free position:
row: enter an integer between 1 and 3:2
column:enter an integer between 1 and 3:2
1 o . .
2 x o .
3 . . .
  1 2 3

player x chooses a free position:
row: enter an integer between 1 and 3:2
column:enter an integer between 1 and 3:3
1 o . .
2 x o x
3 . . .
  1 2 3

player o chooses a free position:
row: enter an integer between 1 and 3:3
column:enter an integer between 1 and 3:3
1 o . .
2 x o x
3 . . o
  1 2 3

Player o wins!
```