Tentamen i Grundläggande Programvaruutveckling, TDA548

Joachim von Hacht/Magnus Myreen

Datum: 2016-12-20

Tid: 08.30-12.30

Hjälpmedel: Engelskt-Valfritt språk lexikon

Betygsgränser:

- U: -23
- 3: 24-37
- 4: 38-47
- 5 : 48-60 (max 60)

Lärare: Joachim von Hacht/Magnus Myreen. Någon besöker ca 10.00 och 11.30, tel. 031/7721003

Granskning: Anslås på kurssida.

Instruktioner:

- För full poäng på essä-frågor krävs ett läsbart, begripligt och heltäckande svar. Generellt 1p för varje relevant aspekt av problemet. Oprecisa eller alltför generella (vaga) svar ger inga poäng. Konkretisera och/eller ge exempel. Det är aldrig någon risk att vara övertydlig!
- Det räcker med enbart relevanta kodavsnitt, övrig kod ersätts med "..." (aldrig import, main-metod, etc....)
- Överkomplicerade lösningar kan ge poängavdrag.
- Vi utgår från att användaren alltid skriver rätt och/eller gör rätt (d.v.s ingen felhantering behövs). Om felhantering skall ingå anges detta specifikt.

LYCKA TILL...

1. Vad avses med? 4p

   a) Konstruktor

   b) Formell parameter

   Förklara med en eller ett par meningar, du får gärna förtydliga med en skiss eller med kod.

2. Vilka av raderna 1-8 nedan kompilerar ej? Motivera varför! 2p

```
int i = a(5);          // 1
double d = a(5.0);     // 2
d = a(5);              // 3
d = c(1, 2.0, 3);      // 4
d = a(1) + c(2, 3);    // 5

int a(int n) { return n + 1; }               // 6
double b(double n) { return n + 1; }         // 7
double c(int n, double d) { return n + d;}   // 8
```

3. Det förefaller som att man, givet ett positivt heltal n, kan komma till talet 1 genom att upprepade gånger tillämpa två enkla regler för n. 4p

   - Om n är jämt: Sätt n till $\frac{n}{2}$ (heltalsdivision)
   - Om n är udda: Sätt n till $3n + 1$.

   Skriv en metod, plot, som givet n, använder reglerna för att skriva ut alla tal $[n, 1]$ (vi antar att vi kommer till 1). Exempel:

```
plot(6)   ger utskrift   6, 3, 10, 5, 16, 8, 4, 2, 1
plot(1)   ger utskrift   1
plot(23)  ger utskrift  23, 70, 35, 106, 53, 160, 80, 40,
                                    20, 10, 5, 16, 8, 4, 2, 1
```

4. Rita en bild som visar variabler, värden, referenser och objekt samt hur dessa förhåller sig till varann före, respektive efter anropet av metoden process (rita som vi ritat under föreläsningarna, lådor, pilar o.s.v.). 6p

```
// Så här ser det ut innan anropet
C[] ca1 = {new C(1), new C(2), new C(3)};
C[] ca2 = {new C(3), new C(2), new C(5)};

process( ca1, ca2 );       // Anropet

// Metoden process. Hur ser det ut efter att denna körts?
void process(C[] cArr1, C[] cArr2) {
    for (int i = 0; i < cArr1.length; i++) {
        if (cArr1[i].n == cArr2[i].n) {
          cArr1[i] = cArr2[i];
        } else {
          cArr1[i].n = cArr2[i].n;
        }
    }
}

public class C {
    int n;
   C(int n) {
        this.n = n;
    }
}
```

5. Skriv en metod, intersection(a1, a2), som givet två heltals-arrayer, a1 och a2, returnerar en array med de element som finns i både a1 och a2. De givna arrayerna får inte ändras. Resultatet behöver inte vara ordnat på något speciellt sätt. Exempel: 6p

```
a1               a2               Resultat
[ 1,2,3,4,5 ]    [ 5,8,9,3,0 ]     [ 3,5 ]  eller [ 5,3 ]
[ 1,2,3 ]        [ 4,5,6 ]         []
[ 1,2,3 ]        []                []
[ 1,2,3 ]        [ 1,2,3 ]         [ 1,2,3 ]
```

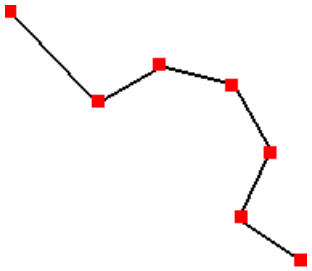
6. Skriv en klass PolyLine för att representera en kontinuerlig linje av linjesegment i 2D, se Figur 1. Ett linjesegment skall representeras som fyra reella tal, x1, y1, x2, y2. Första par anger startpunkt och sista anger slutpunkt. Eftersom linjen är kontinuerlig kan segment dela start- och slutpunkt. Följande gäller: 12p

   a) Klassen skall ha en konstruktor som tar en array av reella tal för linjesegmentens start- och slutpunkter.

   b) Klassen skall ha en metod translate(dx, dy) som flyttar ett PolyLine-objekt dx i x-led och dy i y-led.

   c) Klassen skall ha en metod concat som sammanfogar aktuellt objekt med ett annat PolyLine-objekt. Som resultat returneras ett nytt PolyLine-objekt där det aktuella objektets slutpunkt har sammanfogats med det andra objektets startpunkt. Det nya objektet skall vara helt skilt från de andra objekten (operanderna).

   d) Visa hur du instansierar två PolyLine-objekt och därefter slår ihop dessa m.h.a. concat-metoden.



Figur 1: En polyline med 6 segment.

7. Skriv en metod, sumFromString, som summerar alla heltal i en sträng. I strängen finns bara positiva heltal. Exempel: 6p

```
"arne12fia13pelle8lisa7" ger summan 40
"arnelisa" ger summan 0
"12345" ger summan 12345
"1mmm2nn?n3nn##n4b&b5" ger summan 15
```

Tillåtna metoder från olika klasser:

```
String
- charAt(int i), ger tecknet vid index i.
- indexOf(char ch), ger index för tecknet ch, -1 om tecknet saknas.
- length() ger längden av strängen.
- subString(int start, int end), ger en delsträng från
  start (inkl.) till end-1.
- subString(int start), ger en delsträng från start (inkl.)
  till strängens slut.
- toCharArray(), gör om strängen till en array med tecken
- split(String str), delar upp en sträng i en array av delsträngar
  utifrån ett visst tecken. Returnerar en String-array (String[])
  Exempel "aaa:bb:cccc:dd".split(":") -> [ "aaa", "bb", "cccc", "dd" ]
StringBuilder
- append(String s), lägger till strängen s sist i Stringbuilder-objektet.
- append( char ch ), som ovan
- setLength(), sätter aktuell längd, setLength(0) raderar alla tecken.
- toString(), omvandlar StringBuilder-objektet till en String.
Character
- isDigit() får användas (klassmetod). Metoden returnerar
sant för tecknen 0-9.
Integer
- valueOf(String s) får användas (klassmetod). Metoden
omvandlar en sträng till ett heltal. OBS! Att metoden kastar
ett undantag för tomma strängen
```

8. Förklara *med hjälp av exempel* följande begrepp:

   a) statisk typ 1p

   b) dynamisk typ 1p

   c) typkonvertering 2p

9. Metoden `test` skriver ut ett mönster av tecken.

```
void setMark(int[][] a, int x, int y) {
    a[x][y] = 1;     // rad X
}
void mark(int[][] a, int x, int y, int depth) {
    if (depth == 0) { return; }
    setMark(a,x,y);
    mark(a,x+1,y,depth-1);
    mark(a,x-1,y,depth-1);
    mark(a,x,y+1,depth-1);
    mark(a,x,y-1,depth-1);
}
void test(int dim, int depth) {
    // skriv in nollor
    int[][] a = new int[dim][dim];
    for (int i=0;i<dim;i++) {
        for (int j=0;j<dim;j++) {
            a[i][j] = 0;
        }
    }
    // skriv in siffror
    mark(a,dim/2,dim/2,depth);
    // skriv ut siffrorna
    for (int i=0;i<dim;i++) {
        for (int j=0;j<dim;j++) {
            if (a[i][j] == 0) {
                System.out.print(" ");
            } else {
                System.out.print(a[i][j]);
            }
        }
        System.out.println();
    }
}
```

a) Vad skrivs ut när `test(5,2)` körs? Vad skrivs när `test(5,3)` körs? 4p

b) Skriv kod för en ny version av `mark` metoden. Din metod skall inte använda rekursion men skall ha samma effekt som ett anrop till originalversionen ovan. 6p

c) Vad blir utskriften av `test(5,3)` ifall raden som är markerad ovan med `rad X` byts till `a[x][y] = 1 + a[x][y];`? Ditt svar ska vara baserat på den rekursiva versionen `mark` given i koden ovan. 6p