

Tentamen

Förklaringar

Utforma noggrant dina svar, kodavsnitt och bilder

Formulera dina svar kortfattat och noggrant.

Koden ska utformas så att det lätt går att följa och förstå den. I vissa situationer kan lämpliga kommentarer bidra till förståelse. Små syntaktiska fel i koden kan eventuellt tolereras. Om delar i ett kodavsnitt inte kan exakt formuleras, kan möjligen en välutformad pseudokod bidra till lösningen. Man ska inte skriva mer kod än som behövs: om bara en metod krävs, behöver inte en hel klass skapas. All programmeringskod ska skrivas i Java.

När en vektor eller ett objekt ritas, ska det klart framgå vilken referens refererar till denna vektor eller detta objekt, och vilka data som finns inuti denna vektor eller detta objekt. När en vektor eller ett objekt innehåller en referens, ska även den refererade resursen (ett objekt eller en vektor) ritas. Man ska förse alla referenser med relevanta beteckningar.

Antalet poäng och betygsgränser

Totalt: 42 poäng

För betyget E räcker : 21 poäng

För betyget D räcker: 25 poäng

För betyget C räcker: 29 poäng

För betyget B räcker: 33 poäng

För betyget A räcker: 37 poäng

.

Uppgifter

Uppgift 1 (3 poäng + 3 poäng)

```
int[][] b = { {0, 1, 2},
               {3, 2, 1},
               {2, 3, 4} };

int[] u = new int[b.length];
int m = 0;
for (int i = 0; i < u.length; i++)
{
    m = b[i][0];
    for (int j = 1; j < b[i].length; j++)
        if (b[i][j] > m)
            m = b[i][j];
    u[i] = m;
}

int[][] v = new int[2][];
v[0] = b[2];
v[1] = b[1];
```

a) Rita den vektor som refereras med referensen `u`.

b) Rita den vektor som refereras med referensen `v`.

Uppgift 2 (3 poäng + 3 poäng + 3 poäng)

Klassen `Triangle` representerar en triangel:

```
class Triangle
```

```

{
    // triangelns sidlängder
    private double    a;
    private double    b;
    private double    c;

    public Triangle (double a, double b, double c)
    {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    // perimeter returnerar triangelns omkrets
    public double perimeter ()
    {
        return  a + b + c;
    }

    // area returnerar triangelns area
    public double area ()
    {
        double    s = (a + b + c) / 2;
        double    ar = Math.sqrt (s * (s - a) * (s - b) * (s - c));

        return ar;
    }
}

```

- a) En statisk metod, `maxTriangle`, tar emot en vektor med trianglar (objekt av typen `Triangle`), och returnerar den triangel i vektorn som har största omkrets. Skapa den metoden.
- b) En statisk metod, `totalArea`, tar emot en vektor med trianglar (objekt av typen `Triangle`), och returnerar den totala arean av dessa trianglar. Skapa den metoden.
- c) Skapa en vektor med trianglar (objekt av typen `Triangle`), och anropa metoderna `maxTriangle` och `totalArea` på något sätt.

Uppgift 3 (3 poäng + 3 poäng + 3 poäng)

Klassen `DigitHandler` hanterar en sekvens med siffror på olika sätt:

```

class DigitHandler
{
    private byte[]    digits;

    public DigitHandler (String digits)
    {
        this.digits = new byte[digits.length ()];
        for (int pos = 0; pos < this.digits.length; pos++)
            this.digits[pos] = (byte) (digits.charAt(pos) - 48);
    }

    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("");
        for (byte digit : digits)
            sb.append (digit);

        return sb.toString ();
    }

    public void circularShiftLeft ()
    // koden saknas här

    public void circularShiftRight ()

```

```
// koden saknas här
}
```

En instans av klassen `DigitHandler` skapas och används så här:

```
DigitHandler handler = new DigitHandler ("12345"); (1)
System.out.println (handler);
System.out.println ();

handler.circularShiftLeft ();
System.out.println (handler);
handler.circularShiftLeft ();
System.out.println (handler);
handler.circularShiftRight ();
System.out.println (handler);
handler.circularShiftRight ();
System.out.println (handler);
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
12345

23451
34512
23451
12345
```

- Hur ser det objekt ut som refereras med referensen `handler` när satsen (1) har exekverats? Rita objektet.
- Implementera metoden `circularShiftLeft`.
- Implementera metoden `circularShiftRight`.

Uppgift 4 (3 poäng + 3 poäng + 3 poäng)

Klassen `Word` representerar ett ord. Tecken i ordet lagras i en sekvens länkade noder.

```
class Word
{
    private class Node
    {
        public char    c;
        public Node    next;

        public Node (char c)
        {
            this.c = c;
            this.next = null;
        }
    }

    // första nod i nodsekvensen
    private Node    first = null;

    // Word skapar ett ord ifrån en teckenvektor
    // koden saknas här

    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("");
        Node    node = first;
        if (node != null)
        {
```

```

        while (node.next != null)
        {
            sb.append (node.c);
            node = node.next;
        }
        sb.append (node.c);
    }
    sb.append ("]");

    return sb.toString ();
}

public void combine (Word w)
{
    if (w.first != null)
    {
        Node    n1 = this.first;
        Node    n2 = w.first;
        if (this.first == null)
        {
            this.first = new Node (n2.c);
            n1 = this.first;
            n2 = n2.next;
        }

        while (n1.next != null)
            n1 = n1.next;

        while (n2 != null)
        {
            n1.next = new Node (n2.c);
            n1 = n1.next;
            n2 = n2.next;
        }
    }
}
}

```

Klassen Word används så här:

```

char[]    v = {'L', 'i', 'f', 'e'};
Word      w = new Word (v);
System.out.println (w);

```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
[Life]
```

- Skapa konstruktorn i klassen Word.
- Rita det objekt som refereras med referensen w.
- Vilken utskrift skapas om klassen Word används så här?

```

char[]    v1 = {'L', 'i', 'g', 'h', 't'};
Word      w1 = new Word (v1);
char[]    v2 = {'L', 'o', 'v', 'e'};
Word      w2 = new Word (v2);
w1.combine (w2);
System.out.println (w1);

```

Uppgift 5 (3 poäng + 3 poäng + 3 poäng)

En algoritm sorterar en sekvens med element, som kan jämföras med operatoren *mindre* (<). Algoritmen används nedan för att sortera en sekvens med heltal:

```
public static void sort (int[] sequence)
```

```
{
    int    lastPos = sequence.length - 1;
    int    i = 0;
    for (int pos = 0; pos < lastPos; pos++)
    {
        for (int p = pos + 1; p <= lastPos; p++)
            if (sequence[p] < sequence[pos])
            {
                i = sequence[pos];
                sequence[pos] = sequence[p];
                sequence[p] = i;
            }

        System.out.println (java.util.Arrays.toString (sequence));
    }
}
```

a) Metoden `sort` anropas så här:

```
int[]    seq = {5, 4, 3, 2, 1};
System.out.println (java.util.Arrays.toString (seq));
System.out.println ();

sort (seq);
System.out.println ();
System.out.println (java.util.Arrays.toString (seq));
```

Vilken utskrift skapas när detta kodavsnitt utförs?

b) Specificera algoritmens förvillkor och eftervillkor. Representera steg i algoritmen i form av pseudokod.

c) Låt n beteckna antalet element som sorteras.

Bestäm tidskomplexiteten för algoritmen när det gäller antalet elementjämförelser. Kategorisera motsvarande komplexitetsfunktion: till vilken Θ -mängd tillhör den?

Bestäm även tidskomplexiteten för algoritmen i värsta fall när det gäller antalet elementutbyten. Till vilken Θ -mängd tillhör motsvarande komplexitetsfunktion?