

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 = { {1, 2, 3},
               {4, 5, 6},
               {7, 8, 9} };

int[] u = new int[b.length];
int k = b[0].length - 1;
for (int i = 0; i < u.length; i++)
{
    u[i] = b[i][k];
    k = k - 1;
}

int[][] v = new int[b.length][b[0].length];
for (int i = 0; i < v.length; i++)
    for (int j = 0; j < v[i].length; j++)
        v[i][j] = ((i + j) % 2 == 1) ? 0 : b[i][j];
```

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 `RationalNumber` representerar ett rationellt tal:

```
class RationalNumber
{
```

```

// täljaren och nämnaren
private int    p;
private int    q;

public RationalNumber (int p, int q)
{
    if (q == 0)
        throw new java.lang.IllegalArgumentException ("zero denominator");

    this.p = p;
    this.q = q;
}

public String toString ()
{
    String    s = "" + Math.abs (this.p) + "/" + Math.abs (this.q);
    if (this.p * this.q < 0)
        s = "-" + s;

    return s;
}

// sum returnerar summan av det här rationella talet
// och ett givet rationellt tal
public RationalNumber sum (RationalNumber r)
{
    int    sumP = this.p * r.q + r.p * this.q;
    int    sumQ = this.q * r.q;
    RationalNumber    sum = new RationalNumber (sumP, sumQ);

    return sum;
}

// lessThan returnerar true om det här rationella talet
// är mindre än ett givet rationellt tal, annars returnar
// false.
public boolean lessThan (RationalNumber r)
{
    int    differenceP = this.p * r.q - r.p * this.q;
    int    differenceQ = this.q * r.q;
    boolean    isLessThan = differenceP * differenceQ < 0;

    return isLessThan;
}
}

```

- a) En statisk metod, `totalSum`, tar emot en vektor med rationella tal (objekt av typen `RationalNumber`), och returnerar den totala summan av dessa tal (som ett objekt av typen `RationalNumber`). Skapa den metoden.
- b) En statisk metod, `maxNumber`, tar emot en vektor med rationella tal (objekt av typen `RationalNumber`), och returnerar det största talet i vektorn. Skapa den metoden.
- c) Skapa en vektor med rationella tal (objekt av typen `RationalNumber`), och anropa metoderna `totalSum` och `maxNumber` på något sätt.

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

Klassen `Element` representerar ett kemiskt element:

```

class Element
{
    private String    name;
    private String    symbol;

```

```
public Element (String name, String symbol)
{
    this.name = name;
    this.symbol = symbol;
}

public String toString ()
{
    return "<" + name + ", " + symbol + ">";
}
}
```

Klassen `ElementList` representerar en lista med kemiska element:

```
class ElementList
{
    public static final int    CAPACITY = 4;

    private Element[]    elements;
    private int    numberOfElements = 0;

    public ElementList (Element... elements)
    {
        if (elements.length > CAPACITY)
            throw new java.lang.IllegalArgumentException ("not enough capacity");

        this.elements = new Element[CAPACITY];
        for (int pos = 0; pos < elements.length; pos++)
            this.elements[pos] = elements[pos];
        numberOfElements = elements.length;
    }

    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("{");
        for (int pos = 0; pos < numberOfElements; pos++)
            sb.append (elements[pos]);
        sb.append ("}");

        return sb.toString ();
    }

    public void addFirst (Element element)
    // koden saknas här

    public Element takeFirst ()
    // koden saknas här
}
```

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

```
Element    e1 = new Element ("Zinc", "Zn");
Element    e2 = new Element ("Iron", "Fe");
Element    e3 = new Element ("Magnesium", "Mg");
Element    e4 = new Element ("Calcium", "Ca");

ElementList    list = new ElementList (e1, e2);
System.out.println (list);
System.out.println ();

list.addFirst (e3);
System.out.println (list);
list.addFirst (e4);
System.out.println (list);
```

```
Element    e = list.takeFirst ();
System.out.println (list);
e = list.takeFirst ();
System.out.println (list);
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
{<Zinc, Zn><Iron, Fe>}

{<Magnesium, Mg><Zinc, Zn><Iron, Fe>}
{<Calcium, Ca><Magnesium, Mg><Zinc, Zn><Iron, Fe>}
{<Magnesium, Mg><Zinc, Zn><Iron, Fe>}
{<Zinc, Zn><Iron, Fe>}
```

- a) Hur ser det objekt ut som refereras med referensen `list` när denna kodsekvens har exekverats? Rita objektet.
- b) Implementera metoden `addFirst`.
- c) Implementera metoden `takeFirst`.

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

Klassen `Point` representerar en punkt i planet:

```
class Point
{
    // punktens koordinater
    private double    x;
    private double    y;

    public Point (double x, double y)
    {
        this.x = x;
        this.y = y;
    }

    public String toString ()
    {
        return "(" + x + ", " + y + ")";
    }
}
```

Klassen `PointN` representerar en namngiven punkt i planet:

```
class PointN extends Point
{
    // punktens namn
    private String    name;

    // PointN skapar en namngiven punkt utifrån dess
    // namn och koordinater
    public PointN (String name, double x, double y)
        throws NameFormatException
    {
        // en sats saknas här

        char    c = name.charAt (0);
        if (!Character.isLetter (c))
            throw new NameFormatException ("wrong name " + name);

        // en sats saknas här
    }

    public String toString ()
```

```
// koden saknas här
}
```

Klasserna `Point` och `PointN` används så här:

```
Point p1 = new Point (1, 2);
System.out.println (p1);
Point p2 = new PointN ("A", 3, 4);
System.out.println (p2);
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
(1.0, 2.0)
A (3.0, 4.0)
```

- Ange de två satserna som saknas i konstruktorn i klassen `PointN`.
- Skapa metoden `toString` i klassen `PointN`.
- Skapa klassen `NameFormatException`.

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

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

```
public static void sort (int[] sequence)
{
    int    lastPos = sequence.length - 1;
    int    maxPos = 0;
    int    i = 0;
    for (int pos = lastPos; pos > 0; pos--)
    {
        maxPos = 0;
        for (int p = 1; p <= pos; p++)
            if (sequence[p] > sequence[maxPos])
                maxPos = p;

        if (maxPos != pos)
        {
            i = sequence[pos];
            sequence[pos] = sequence[maxPos];
            sequence[maxPos] = i;
        }

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

a) Metoden `sort` anropas så här:

```
int[] seq = {5, 1, 2, 3, 4};
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?

- Specificera algoritmens förvillkor och eftervillkor. Representera steg i algoritmen i form av pseudokod.
- Låt n beteckna antalet element som sorteras.

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