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, 0, 0, 4},
                 {0, 2, 3, 0},
                 {0, 2, 3, 0},
                 {1, 0, 0, 4} };

int[]    u = new int[b.length];
for (int i = 0; i < u.length; i++)
    for (int j = 0; j < b[i].length; j++)
        u[i] = u[i] + b[i][j];

int[][]    v = new int[2][b.length];
for (int i = 0; i < b.length; i++)
{
     v[0][i] = b[i][i];
     v[1][i] = b[i][b[i].length - 1 - i];
}
```

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 `CharSet` representerar en teckenmängd.

```
class CharSet
{
```

```
    // tecken i mängden
    private char[]    elements;

    public CharSet (char[] elements)
    {
        this.elements = new char[elements.length];
        for (int i = 0; i < elements.length; i++)
            this.elements[i] = elements[i];
    }

    // countElements returnerar antalet tecken i mängden
    public int countElements ()
    {
        return elements.length;
    }

    // contains returnerar true om mängden innehåller
    // ett givet tecken, annars false.
    public boolean contains (char c)
    {
        boolean    contains = false;
        for (int i = 0; i < elements.length; i++)
            if (this.elements[i] == c)
            {
                contains = true;
                break;
            }

        return contains;

    }
}
```

a) En statisk metod, `largestSet`, tar emot en vektor med mängder (objekt av typen `CharSet`) och returnerar den mängd som har flest tecken. Skapa den metoden.

b) En statisk metod, `selectSets`, tar emot en vektor med mängder (objekt av typen `CharSet`) och ett tecken, och returnerar de mängder (som en vektor) som innehåller detta tecken. Skapa den metoden.

c) Skapa en vektor med mängder (objekt av typen `CharSet`). Anropa sedan metoderna `largestSet` och `selectSets` i samband med den vektorn.

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

Klassen `IntegerSequence` representerar en heltalssekvens.

```
class IntegerSequence
{
    // heltal i sekvensen
    private Integer[]    numbers;

    public IntegerSequence (Integer[] numbers)
    {
        if (numbers.length == 0)
            throw new java.lang.IllegalArgumentException ("empty array");

        this.numbers = new Integer[numbers.length];
        for (int pos = 0; pos < numbers.length; pos++)
            this.numbers[pos] = numbers[pos];
    }

    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("{");
        for (int pos = 0; pos < numbers.length - 1; pos++)
```

```
            sb.append (numbers[pos] +", ");
        sb.append (numbers[numbers.length - 1] + "}");

        return sb.toString ();
    }


    // addFirst lägger till ett givet heltal i början av sekvensen
    // koden saknas här


    // meanValue returnerar medelvärdet av heltalen i sekvensen
    // koden saknas här
}
```

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

```
Integer[]    numbers = { new Integer (2),
                         new Integer (4),
                         new Integer (8) };
IntegerSequence    seq = new IntegerSequence (numbers);
System.out.println (seq);

seq.addFirst (new Integer (1));
System.out.println (seq);

double     meanVal = seq.meanValue ();
System.out.println (meanVal);
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
{2, 4, 8}
{1, 2, 4, 8}
3.75
```

a) Hur ser det objekt ut som refereras med referensen `seq` när kodsekvensen har exekverats? Rita objektet.

b) Implementera metoden `addFirst`.

c) Implementera metoden `meanValue`.

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

Klassen `IntList` representerar en lista med heltal.

```
class IntList
{
    // Node representerar en nod
    private static class Node
    {
        public int     number;
        public Node    next;

        public Node (int number, Node next)
        {
            this.number = number;
            this.next = next;
        }
    }

    // referens till listans första nod
    private Node    first = null;

    public IntList (int[] numbers)
    {
```

```
        if (numbers != null)
        {
            Node    n = new Node (numbers[0], null);
            first = n;
            for (int pos = 1; pos < numbers.length; pos++)
            {
                n.next = new Node (numbers[pos], null);
                n = n.next;
            }
        }
    }

    public String toString ()
    {
        String    s = "[";
        Node    n = first;
        while (n != null)
        {
            s = s + n.number;
            if (!(n.next == null))
                s = s + ", ";
            n = n.next;
        }
        s = s + "]";

        return s;
    }


    // size returnerar antalet heltal i listan
    // koden saknas här


    // get returnerar det heltal som finns på en given position.
    // Om en ogiltig position anges, kastas ett undantag av typen
    // IndexOutOfBoundsException.
    // koden saknas här
}
```

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

```
int[]    numbers = {1, 4, 5, 10};
IntList    list = new IntList (numbers);
System.out.println (list);

System.out.println (list.size ());
System.out.println ();

for (int pos = 0; pos < list.size (); pos++)
     System.out.println (list.get (pos));
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
[1, 4, 5, 10]
4

1
4
5
10
```

a) Hur ser det objekt ut som refereras med referensen `list`? Rita objektet.

b) Implementera metoden `size`.

c) Implementera metoden `get`.

Uppgift 5 (4 poäng + 5 poäng)

En algoritm, som sorterar en sekvens med element, kan illustreras som nedan:

```
[3, 5, 4, 2, 1]

[3, 1, 4, 2, 5]
[3, 1, 2, 4, 5]
[2, 1, 3, 4, 5]
[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5]
```

a) 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?

b) Skapa en metod `sort` som tar emot en vektor med heltal, och sorterar den enligt givna algoritmen.