Tentamen – obligatorisk del

Förklaringar

Den här delen av tentamen är obligatorisk och tillräcklig för betyget E

För att klara tentamen, måste studenten klara den här delen av tentamen. Om bara den här delen klaras, blir betyget på tentamen E. För att uppnå ett högre betyg, måste studenten samla ett tillräckligt antal poäng även på den extra delen av tentamen. Beroende på detta antal, kan betyget på tentamen bli D, C, B eller A.

För att klara den här delen av tentamen

För att klara den här delen av tentamen, måste studenten uppnå minst två tredjedelar av det totala antalet poäng på den här delen. Den här delen bidrar inte på något sätt till de högre betygen. Betyget på denna del är antingen P eller F.

Antalet poäng

Totalt: 34 poäng
För betyget P krävs minst: 22 poäng

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 kod ska skrivas i Java.

När en vektor eller ett objekt ritas, ska det klart framgå 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.

Uppgifter

Uppgift 1 (1 poäng + 1 poäng)

a)

```
int[]    u = {1, 2, 3, 4, 5};
int    pos = 0;
while (pos < u.length / 2)
{
    u[pos] = u[u.length - 1 - pos];
    pos++;
}
```

Hur ser den skapade vektorn ut när det här kodavsnittet har utförts: rita både vektorn och motsvarande referens.

b)

```
int[]    v = {1, 2, 3, 4, 5, 6};
int    e = 0;
for (int pos = 0; pos < v.length / 2; pos++)
{
    e = v[pos];
    v[pos] = v[v.length - 1 - pos];
    v[v.length - 1 - pos] = e;
}
```

Hur ser den skapade vektorn ut när det här kodavsnittet har utförts: rita både vektorn och den motsvarande referensen.

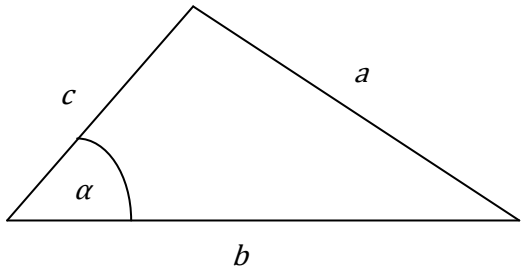
Uppgift 2 (3 poäng)

```
int[][]    v = new int[3][5];

int    rad = 0;
int    kol = 0;
for (kol = 0; kol < v[rad].length; kol++)
    v[rad][kol] = 5 - kol;

rad = 1;
for (kol = 0; kol < v[rad].length; kol++)
    v[rad][kol] = kol % 2;

rad = 2;
for (kol = 0; kol < v[rad].length; kol++)
    v[rad][kol] = (rad > kol)? rad : kol;
```

Rita den skapade vektorn, så att det framgår hur vektorn lagras i datorns minne. Både vektorns celler, motsvarande data och alla referenser ska finnas med. Det ska även finnas motsvarande beteckningar på referenserna.

Uppgift 3 (2 poäng + 1 poäng)

Längder av sidor i en triangel är a, b och c. Vinkeln som motsvarar sidan a är α.



Enligt cosinussatsen är:

$$a^2 = b^2 + c^2 - 2 * b * c * \cos \alpha$$

a) Skapa en statisk metod `vinkel`, som tar emot längderna a, b och c, och returnerar vinkeln α i grader.

b) Anropa metoden `vinkel` för att bestämma vinkeln α i fall att: a = 2.5, b = 4.0, c = 3.5.

Uppgift 4 (2 poäng)

En statisk metod `min` tar emot en icke-tom, tvådimensionell vektor med heltal, och returnerar det minsta heltalet i vektorn. Skapa den metoden.

Uppgift 5 (3 poäng)

En algoritm bestämmer det största elementet i en heltalsmängd. Denna algoritm kan beskrivas så här:

Algoritm: *max*

Förvillkor:

$n \in N, n \geq 1, X = \{x_1, x_2, ..., x_n\} \subset Z$

(N – mängden av alla naturliga heltal, Z – mängden av alla heltal)

Eftervillkor:

$m \in N, 1 \leq m \leq n$: $x_m = maximum\ X$

```
max (n, X)
{
    i = 1
    m = 1
    while i < n
    {
        i++
        if (x_i > x_m)
            m = i
    }

    return x_m
}
```

Spåra denna algoritm i samband med följande mängd:

$X = \{7, 3, 2, 8, 9, 10, 4, 1\}$

Samla relevanta data i en tabell av följande form:

i	x_i	m	x_m

Uppgift 6 (2 poäng + 1 poäng)

En statisk metod `taBortInledandeNollor` tar emot en icke-tom teckensträng av godtycklig längd, som bara innehåller siffror. I början av strängen kan ett antal nollor finnas (till exempel `"000457635646670043577 77"`), men strängen består inte enbart av nollor. Metoden returnerar en ny teckensträng, som innehåller samma siffersekvens, men utan de eventuella inledande nollorna.

a) Skapa metoden `taBortInledandeNollor`. Metoden ska vara minneseffektiv: man ska inte omvandla teckensträngen till motsvarande teckenvektor.

b) Anropa metoden `taBortInledandeNollor` på något sätt.

Uppgift 7 (3 poäng)

```
class V
{
    public static void main (String[] args)
    {
```

```
        StringBuilder    str = new StringBuilder ("aaaa");
        str.insert (0, "X");
        System.out.println (str);

        str = change (str);
        str.reverse ();
        System.out.println (str);
    }


    public static StringBuilder change (StringBuilder s)
    {
        s.append ("Y");
        s = new StringBuilder ("bbbb");
        s.setCharAt (0, 'B');
        System.out.println (s);

        return s;
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Uppgift 8 (2 poäng + 2 poäng + 2 poäng)

En klass `Punkt` representerar en punkt i planet:

```
public class Punkt
{
    // punktens koordinater
    private double    x;
    private double    y;

    // Punkt initierar punkten utifrån givna koordinater
    public Punkt (double x, double y)
    {
        this.x = x;
        this.y = y;
    }


    // toString returnerar punktens strängrepresentation
    // Den returnerade strängen är på formen: [3.0, 4.0].

    // koden här


    // equals returnerar true om punkten har lika koordinater
    // som en given punkt, annars false.

    // koden här
}
```

a) Implementera metoden `toString`.

b) Implementera metoden `equals`.

c) Skapa två punkter av typen `Punkt`, och anropa därefter metoderna `toString` och `equals` i samband med punkterna.

Uppgift 9 (2 poäng + 2 poäng + 2 poäng)

Klassen `PointCollection` representerar en samling punkter:

```
import java.awt.Point;

class PointCollection
{
    // punkter i samlingen
    private Point[]    points;
```

```
    // PointCollection skapar en tom punktsamling
    public PointCollection ()
    {
        this.points = new Point[0];
    }

    // toString returnerar samlingens strängrepresentation
    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("{ ");
        for (int i = 0; i < points.length; i++)
            sb.append (points[i].toString ().substring (14) + " ");
        sb.append ("}");

        return sb.toString ();
    }

    // add lägger till en given punkt till samlingen
    public void add (Point point)
    {
        Point[]    p = new Point[this.points.length + 1];
        int    i = 0;
        for (i = 0; i < this.points.length; i++)
            p[i] = this.points[i];
        p[i] = new Point (point);

        this.points = p;
     }
}
```

a) Skapa en punktsamling av typen `PointCollection`, som innehåller punkterna `(3, 4)` och `(5, 6)`.

b) Vilken utskrift skapas när den skapade samlingen visas med metoden `System.out.println`.

c) Rita den skapade samlingen. Alla objekt och vektorer, med motsvarande referenser, ska finnas med.

Uppgift 10 (1 poäng + 1 poäng + 1 poäng)

Betrakta följande kodavsnitt:

```
Object[]    v = new Object[6];
v[0] = new String ("red");
v[1] = new java.awt.Color (255, 0, 0);
v[2] = new String ("green");
v[3] = new java.awt.Color (0, 255, 0);
v[4] = new String ("blue");
v[5] = new java.awt.Color (0, 0, 255);

for (int pos = 0; pos < v.length; pos++)
{
    // System.out.println (v[pos].toString ());  // (1)
    if (v[pos] instanceof String)
    {
        String    s = (String) v[pos];
        System.out.println (s.toUpperCase ());
        // System.out.println (v[pos].toLowerCase ());  // (2)
    }
}
```

a) Man lagrar i vektorn (som refereras av referensen) `v` både objekt av typen `String` och objekt av typen `Color`. Varför är det möjligt?

b) Vilken utskrift skapas när det här kodavsnittet exekveras?

c) Man kan inkludera satsen `(1)` i kodavsnittet, men inte satsen `(2)` (om man gör det uppstår ett kompileringsfel). Varför?