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: 41 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 (2 poäng + 3 poäng)

```
int[]    u = new int[5];
for (int pos = u.length - 1; pos >= 0; pos--)
    u[pos] = (pos == u.length - 1)? 10 : u[pos + 1] - 2;

int[][]    v = new int[3][];
v[0] = new int[4];
v[1] = new int[3];
v[2] = new int[2];
for (int row = 0; row < v.length; row++)
    for (int column = 0; column < v[row].length; column++)
        v[row][column] = column;
```

a) Rita den vektor som refereras med referensen u.

b) Rita den vektor som refereras med referensen v.

Uppgift 2 (1 poäng + 2 poäng + 2 poäng)

Rita de referenser, vektorer och objekt som skapas i följande fall:

a)

```
String    s1 = new String ("abcde");
String    s2 = s1;
```

b)

```
StringBuilder    sb1 = new StringBuilder ("01234");
StringBuilder    sb2 = new StringBuilder ();
sb2.append ("01234");
```

c)

```
int[]    v1 = {0, 1, 2, 3, 4};
int[]    v2 = v1;
v2[0] = 5;
```

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

En klass `Point` representerar en punkt i planet:

```
class Point
{
     public static final Point    ORIGIN = new Point (0, 0);

    // punktens koordinater
    private double    x;
    private double    y;

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

     public double distance (Point p)
     {
         return Math.sqrt ( (this.x - p.x) * (this.x - p.x) +
                            (this.y - p.y) * (this.y - p.y) );
     }
}
```

En statisk metod, `selectPoint`, tar emot två punkter och returnerar den punkt som ligger närmast origo:

```
public static Point selectPoint (Point p1, Point p2)
{
    // koden saknas här
    // använd metoden distance
}
```

En statisk metod, `selectPoint`, tar emot fyra punkter och returnerar den punkt som ligger närmast origo:

```
public static Point selectPoint (Point p1, Point p2, Point p3, Point p4)
{
    // koden saknas här
    // använd metoden selectPoint för två punkter

}
```

a) Skapa metoden `selectPoint` för två punkter.

b) Skapa metoden `selectPoint` för fyra punkter.

c) Skapa fyra punkter, och anropa metoden `selectPoint` i samband med dem.

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

Klassen `Student` hanterar en students namn och betyg:

```
class Student
{
    // antalet betyg
    public static final int    GRADE_COUNT = 4;

    // studentens namn och betyg
    public String    name;
    public int[]    grades;
```

```
    public Student (String name)
    {
        this.name = name;
        grades = new int [GRADE_COUNT];
    }

    // setGrade sätter betyg på den uppgift vars index är givet
    public void setGrade (int gradeIndex, int grade)
    {
        this.grades[gradeIndex] = grade;
    }

    // toString returnerar studentens strängrepresentation
    // koden saknas här

    // averageGrade returnerar studentens medelbetyg
    // koden saknas här
}
```

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

```
Student    student = new Student ("Anna Ericsson");
student.setGrade (0, 3);
student.setGrade (1, 4);
student.setGrade (2, 5);
student.setGrade (3, 5);

System.out.println (student);
System.out.println (student.averageGrade ());
```

När detta kodavsnitt exekveras, skapas följande utskrift:

```
{Anna Ericsson: [3, 4, 5, 5]}
4.25
```

a) Implementera metoden `toString`.

b) Implementera metoden `averageGrade`.

c) Rita det objekt som refereras med referensen `student`.

Uppgift 5 (1 poäng + 2 poäng + 2 poäng + 1 poäng)

Gränssnittet `LetterSupplier`, och klasserna `LetterSupplierA`, `LetterSupplierB` och `LetterSupplierC`, definieras på följande vis:

```
interface LetterSupplier
{
    char lowerCaseLetter ();
    char upperCaseLetter ();
}

class LetterSupplierA implements LetterSupplier
{
    public char lowerCaseLetter ()
    {
        return 'a';
    }

    public char upperCaseLetter ()
    {
        return 'A';
    }
}

class LetterSupplierB implements LetterSupplier
{
```

```
    public char lowerCaseLetter ()
    {
        return 'b';
    }

    public char upperCaseLetter ()
    {
        return 'B';
    }
}

class LetterSupplierC extends LetterSupplierA
{
    public char upperCaseLetter ()
    {
        return 'C';
    }

    public char letter ()
    {
        return 'X';
    }
}
```

Man skapar en vektor med objekt av klasserna `LetterSupplierA`, `LetterSupplierB` och `LetterSupplierC`:

```
LetterSupplier[]    ls = new LetterSupplier[3];
ls[0] = new LetterSupplierA ();
ls[1] = new LetterSupplierB ();
ls[2] = new LetterSupplierC ();
```

a) Vektorn (som refereras med referensen) `ls` innehåller objekt av olika klasser. Varför är det möjligt?

b) Vilken utskrift skapas när följande kodavsnitt exekveras?

```
for (int pos = 0; pos < ls.length; pos++)
{
     System.out.print (ls[pos].lowerCaseLetter () + " | ");
     System.out.println (ls[pos].upperCaseLetter ());
}
```

c)

```
LetterSupplierA   lsa1 = new LetterSupplierC ();
System.out.println (lsa1.upperCaseLetter ());
// System.out.println (lsa1.letter ());    // (1)
```

Vad händer när detta kodavsnitt exekveras? Varför?
Vad händer om satsen `(1)` inkluderas? Varför?

d)

```
LetterSupplierA   lsa2 = new LetterSupplierB ();
System.out.println (lsa2.upperCaseLetter ());
```

Är det här kodavsnittet korrekt? Varför?

Uppgift 6 (2 poäng + 2 poäng + 5 poäng)

Fibonaccitalen är:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Det heltal som finns på position n i den här heltalssekvensen, kan bestämmas så här:

$$f(n) = 0, \quad \text{om } n = 0$$

$$f(n) = 1, \quad \text{om } n = 1$$

$$f(n) = f(n-2) + f(n-1), \quad \text{om } n > 1$$

För att bestämma det Fibonaccital som finns på en given position n, kan följande metod användas:

```
public static long fibonacciNumber (int n)
{
    long[]    f = new long[n + 1];
    f[0] = 0;
    if (n > 0)
    {
        f[1] = 1;
        for (int pos = 2; pos <= n; pos++)
            f[pos] = f[pos - 2] + f[pos - 1];
    }

    return f[n];
}
```

a) Bestäm minneskomplexitet för den algoritm som används i metoden `fibonacciNumber`. Kategorisera motsvarande komplexitetsfunktion: till vilken Θ–mängd tillhör den?

b) Bestäm tidskomplexitet för den algoritm som används i metoden `fibonacciNumber`. En addition ska betraktas som en elementär operation i algoritmen. Kategorisera motsvarande komplexitetsfunktion.

c) Hitta på en ny, minneseffektivare algoritm, för att bestämma Fibonaccitalet på en given position. Algoritmen ska endast använda några få minnesceller. Presentera algoritmen i form av en Java metod. Bestäm algoritmens minneskomplexitet, och kategorisera motsvarande komplexitetsfunktion.