Fadil Galjic

Programmeringsprinciper i Java

Övningar

COPYRIGHT Författare & Studentlitteratur

Innehållsförteckning

Kapitel 1

Ett Javaprogram

Övning 1

```
Class Flera-Meddelanden
{
    public static main (string[] args)
    {
        System.Out.println ("0");
        system.out.println ("1");
        System.out.println ("2")
        // System.out.println ("3");
        System out.println ("4");
    }
}
```

Markera kompileringsfel i det här programmet.

Övning 2

```
class
Meddelande
{
public    static void
main (String[] args)
{
        System.out.println
        ("Lycka till!");
}
}
```

Hur många kompileringsfel finns i det här programmet?

Övning 3

```
class Meddelande
```

```
{
    public static void main (String[] args)
    {
        System.out.println ("Lycka till!");     // ett meddelande
        // System.out.println ("Lycka till!");  // ett meddelande

        /*
        System.out.println ("Lycka till!");
        System.out.println ("Lycka till!");
        */
        System.out.println ("Lycka till!");
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 4

```
// X.java
class X
{
    public static void main (String[] args)
    {
        System.out.println ("1");
        System.out.println ();
        System.out.println ();
        System.out.print ("2");
        System.out.print (" 3\n");
        System.out.println ("4\n5\n6");
        System.out.println ("\n\t7 8 9\t10");
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Kapitel 2

Datalagring

Övning 1

```
class TalTyper
{
    public static void main (String[] args)
    {
        long    m = 5_000_000_000L;
        // long    n = 5_000_000_000;           // (1)
        System.out.println (m);
        // System.out.println (5_000_000_000);  // (2)
        System.out.println (5E+9);

        float    x = 2.5F;
        // float    y = 2.5;                    // (3)
        System.out.println (x);
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Vad händer när man inkluderar satsen `(1)`, eller satsen `(2)`?

c) Vad händer när man inkluderar satsen `(3)`?

Övning 2

a) Vilket talområde kan man representera med fyra bitar, om man representerar bara icke-negativa heltal?

b) Vilket talområde kan man representera med fyra bitar, om man representerar både negativa och icke-negativa heltal?

c) Vilka heltal kan representeras med åtta bitar?

Övning 3

```
class TypOmvandlingar
{
    public static void main (String[] args)
    {
        byte    b1 = 10;
        int     i1 = b1;
        int     i2 = 10;
        byte    b2 = i2;
        System.out.println (i1 + " " + b2);

        float     f1 = 2.5F;
        double    d1 = f1;
        double    d2 = 2.5;
        float     f2 = d2;
        float     f3 = 2.5;
        System.out.println (d1 + " " + f2);
    }
}
```

Varför kan man inte kompilera det här programmet?

Använd uttryckliga typomvandlingar där de behövs, så att det går att kompilera programmet.

Övning 4

```
class TypOmvandlingar
{
    public static void main (String[] args)
    {
        int      i = 10000;
        byte     b = (byte) i;
        short    s = (short) i;
        System.out.println (b + " " + s);

        double    d1 = 2.5;
        double    d2 = 2.5E40;
        double    d3 = 1.23456789012;
        float     f1 = (float) d1;
        float     f2 = (float) d2;
        float     f3 = (float) d3;
        System.out.println (f1 + " " + f2 + " " + f3);
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 5

```
class TypOmvandlingar
{
    public static void main (String[] args)
    {
        int      i1 = 10;
        float    f1 = i1;

        float    f2 = 10.0F;
        int      i2 = f2;
        long     l  = f2;

        System.out.println (f1 + " " + i2 + " " + l);
    }
}
```

Varför går det inte att kompilera det här programmet?

Använd uttryckliga typomvandlingar där de behövs, så att programmet kan kompileras.

Övning 6

```
class TypOmvandlingar
{
    public static void main (String[] args)
    {
        int       i1 = 1234567890;
        float     f1 = i1;
        double    d1 = i1;

        float     f2 = 2.5F;
        int       i2 = (int) f2;
        double    d2 = 2E50;
        long      l  = (long) d2;

        System.out.println (f1 + " " + d1 + " " + i2 + " " + l);
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?.

Övning 7

```
class TypOmvandlingar
{
```

```
    public static void main (String[] args)
    {
        char    c1 = 'A';
        byte    b1 = c1;
        int     i  = c1;

        byte    b2 = 65;
        char    c2 = b2;

        System.out.println (b1 + " " + i + " " + c2);
        System.out.println ((int) 'A');
        System.out.println ((char) 65);
    }
}
```

Varför kan man inte kompilera det här programmet?

Använd uttryckliga typomvandlingar där de behövs, så att programmet kan kompileras. Vilken utskrift erhålls i så fall?

Kapitel 3

Standardinmatning

Övning 1

```
class Inmatning
{
    public static void main (String[] args)
    {
        java.util.Scanner    in =
                          new java.util.Scanner (System.in);

        System.out.print ("ett heltal: ");
        int    tal1 = in.nextInt ();

        System.out.print ("ett heltal: ");
        int    tal2 = in.nextInt ();

        int    sum = tal1 + tal2;
        System.out.println ("heltalens summa: " + sum);
    }
}
```

a) Vad händer om ett heltal tillförs programmet vid den första inmatningen?

b) Vad händer om två heltal tillförs programmet vid den första inmatningen?

c) Vad händer om tre heltal tillförs programmet vid den första inmatningen?

d) Vad händer om man trycker på returtangenten utan att ha matat in ett heltal?

Övning 2

```
class SuccessivaTeckenInmatningar
{
    public static void main (String[] args) throws Exception
    {
        java.io.InputStreamReader   in =
                     new java.io.InputStreamReader (System.in);
```

```
        System.out.print ("ett tecken: ");
        char    c1 = (char) in.read ();
        // in.read ();
        // in.read ();

        System.out.print ("ett tecken: ");
        char    c2 = (char) in.read ();

        System.out.println ("tecknen: " + c1 + c2);
    }
}
```

a) Vad händer om ett tecken tillförs programmet vid den första inmatningen?

b) Vad händer om två tecken tillförs programmet vid den första inmatningen?

c) Vad händer om tre tecken tillförs programmet vid den första inmatningen?

d) Vad händer om man bara trycker på returtangenten?

e) Vad händer om de bortkommenterade satserna inkluderas – först en av dem, och sedan båda två?

Övning 3

```
class RadEfterOrd
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.println ("två ord: ");
        String    ord1 = in.next ();
        String    ord2 = in.next ();
        // in.nextLine ();

        System.out.println (ord1);
        System.out.println (ord2);
        System.out.println ();

        System.out.println ("en rad:");
        String    rad = in.nextLine ();

        System.out.println (rad);
    }
}
```

a) Vad händer om två ord anges i samma rad?

b) Vad händer om två ord anges i två olika rader?

c) Vad händer om tre ord anges i samma rad vid den första inmatningen?

d) Vad händer om den bortkommenterade satsen inkluderas?

Övning 4

```
class RadEfterTal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.println ("två tal: ");
        int    tal1 = in.nextInt ();
        int    tal2 = in.nextInt ();
        // in.nextLine ();

        System.out.println (tal1);
        System.out.println (tal2);
        System.out.println ();

        System.out.println ("en rad:");
        String    rad = in.nextLine ();

        System.out.println (rad);
    }
}
```

a) Vad händer om två heltal anges i samma rad?

b) Vad händer om två heltal anges i två olika rader?

c) Vad händer om tre heltal anges i samma rad vid den första inmatningen?

d) Vad händer om den bortkommenterade satsen inkluderas?

e) Vad händer om tal med decimaler anges i stället för heltal? Hur ska programmet anpassas så att det går att mata in decimaltal?

Övning 5

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

```
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("förnamn: ");
        String    fn = in.nextLine ();
        System.out.print ("efternamn: ");
        String    en = in.nextLine ();
        System.out.print ("födelseår: ");
        int    ar = in.nextInt ();
        // in.nextLine ();
        System.out.println ();

        System.out.println (fn + " " + en + ", " + ar);
        System.out.println ();

        System.out.print ("förnamn, efternamn och födelseår: ");
        String    pu = in.nextLine ();
        System.out.println (pu);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer när den bortkommenterade satsen inkluderas?

Kapitel 4

Operationer med primitiva värden

Övning 1

```
class OperationerMedHeltal
{
    public static void main (String[] args)
    {
        int    m = 50;
        int    n = 20;
        int    p = m / n + m % n;
        int    q = m % n % (n / 5);
        System.out.println (p);
        System.out.println (q);

        short    s1 = 12_000;
        short    s2 = 10_000;
        short    r = (short) (s1 + s2);
        System.out.println (r);

        char    c = 'a';
        char    s = (char) (c + 4);
        System.out.println (s);
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Varför måste en uttrycklig typomvandling utföras vid beräkningen av variablerna `r` och `s`? Vad händer om man inte gör det?

Övning 2

```
class FelVidBerakningar
{
```

```
    public static void main (String[] args)
    {
        int    m = 2_000_000_000;
        int    n = 1_000_000_000;
        int    r1 = (m + n) / 3;
        System.out.println (r1);

        double    r2 = 1_000_000_000.0 / (m - 2 * n);
        System.out.println (r2);

        int    r3 = 1_000_000_000 / (m - 2 * n);
        System.out.println (r3);

        System.out.println (-2 * -3);
    }
}
```

Vad händer när det här programmet exekveras?

Övning 3

Hur många hela veckor finns i ett helt år (med `365` dagar)?

Hur många dagar finns kvar (resten vid heltalsdivisionen)?

Skapa ett program som utför motsvarande beräkningar, och skriver ut resultat..

Övning 4

```
class OmvandlingarVidBerakningar
{
    public static void main (String[] args)
    {
        byte    b = 4;
        char    c = 65;
        int     i = 10;
        long    l = 20L;

        int    res1 = (int) (b + c + i + l);
        System.out.println (res1);

        // int    res2 = b + c + i + l;
        // int    res3 = (int) b + c + i + l;
    }
}
```

a) Vilka automatiska och uttryckliga typomvandlingar sker under programmets gång?

b) Vad händer när någon av de bortkommenterade satserna inkluderas?

Övning 5

```
class OperationerMedTal
{
    public static void main (String[] args)
    {
        int       m = 5;
        int       n = 2;
        float     u = 5;
        double    v = 2;

        int       i = (int) (m / n + u % v);
        float     f = (float) (m / n + u % v);
        double    d = m / n + u % v;
        System.out.println (i + " " + f + " " + d);

        System.out.println (m / n);
        System.out.println ((double) m / n);
        System.out.println (5.2 / 2 + " " + 5.2 % 2);
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Vilka automatiska typomvandlingar sker under programmets gång?

c) Varför måste uttryckliga typomvandlingar användas vid beräkningen av variablerna `i` och `f`?

Övning 6

```
class FelVidFlyttalsBerakningar
{
    public static void main (String[] args)
    {
        float    f1 = 1.0000000005F;
        float    f2 = 4;
        float    f  = f1 + f2;
        System.out.println (f);

        double    d1 = 1.0000000005;
        double    d2 = 4;
```

```
        double    d  = d1 + d2;
        System.out.println (d);
        System.out.println ();

        System.out.println (2E+200 * 2E200);
        System.out.println (2E+200 * 2E200 - 2E+200 * 2E200);
        System.out.println ();

        System.out.println (5.0 / 0);
        System.out.println (-5.0 / 0);
        System.out.println (0.0 / 0);
        System.out.println (5 / 0);

        System.out.println ("slut");
    }
}
```

Vad händer när det här programmet exekveras?

Övning 7

```
class MojligtFel
{
    public static void main (String[] args)
    {
        int     a = 1_000;
        int     b = 1_000;
        long    c = 10_000_000_000L;

        a = a + c;
        b += c;

        System.out.println (a + " " + b + " " + c);
    }
}
```

a) Varför går inte programmet att kompilera?

b) Använd uttrycklig typomvandling vid ändringen av variabeln `a`, så att programmet kan kompileras. Vilken utskrift skapas i detta fall?

Övning 8

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

```
    {
        int   a = 8;
        int   b = 4;
        int   c = ++a * b--;

        System.out.println (a + " " + b + " " + c);
        System.out.println (a++ + " " + ++b + " " + --c);
        System.out.println (a + " " + b + " " + c);
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 9

```
class Jamforelser
{
    public static void main (String[] args)
    {
        int       a = 4;
        int       b = 6;
        int       c = 10;
        int       d = 12;

        boolean    b1 = a == 4;
        boolean    b2 = a != 4;
        boolean    b3 = a + b > d - c + 8;
        boolean    b4 = a + b <= d - c + 8;
        System.out.println (b1 + " " + b2 + " " + b3 + " " + b4);

        double    u = 4;
        float     v = 4;

        boolean    b5 = a == u;
        boolean    b6 = u == v;
        boolean    b7 = 1.0 / 3 == 3.33333333333333;
        System.out.println (b5 + " " + b6 + " " + b7);

        char    c1 = 'A';
        char    c2 = 'a';
        boolean    b8 = c1 < c2;
        System.out.println (b8);
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 10

```
class LogiskaOperationer
{
    public static void main (String[] args)
    {
        int    m = 2;
        int    n = 5;

        boolean    b1 = m > 2  ^   n >= 5;
        boolean    b2 = m > 2  &&  n >= 5;
        boolean    b3 = m > 2  ||  !(n < 5);
        System.out.println (b1 + " " + b2 + " " + b3);

        boolean    b4 = m == 2  ||  n >= 5  &&  m > n;
        boolean    b5 = (m == 2  ||  n >= 5)  &&  m > n;
        boolean    b6 = m == 2  ||  (n >= 5  &&  m > n);
        System.out.println (b4 + " " + b5 + " " + b6);
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 11

```
class Operatorer
{
    public static void main (String[] args)
    {
        int    m = 2;
        int    n = 5;

        boolean    b1 = m > 2  &&  n++ >= 5;
        boolean    b2 = m > 2  ||  n++ >= 5;
        System.out.println (b1 + " " + b2);
        System.out.println (n);
        System.out.println ();

        n = 5;

        boolean    b3 = m > 2  &  n++ >= 5;
        boolean    b4 = m > 2  |  n++ >= 5;
        System.out.println (b3 + " " + b4);
        System.out.println (n);
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 12

```
class Operatorer
{
    public static void main (String[] args)
    {
        int    m = 4;
        int    n = 4;
        boolean    b1 = m-- < n++;
        boolean    b2 = m > n  &&  n++ > 2  &&  !b1;

        System.out.println (b1 + " " + b2);
        System.out.println (m + " " + n);
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Kapitel 5

Logik

Övning 1

```
class SammaLogik
{
    public static void main (String[] args)
    {
        int    m = 10;
        // m = -10;
        // m = 0;
        // m = 5;

        if (m < 0)
            System.out.println (m + " < 0");
        else if (m == 0)
            System.out.println (m + " == 0");
        else
        {
            if (m == 10)
               System.out.println (m + " == 10");
            else
               System.out.println (m + " != 10");
        }

        if (m < 0)
            System.out.println (m + " < 0");
        else if (m == 0)
           System.out.println (m + " == 0");
        else if (m == 10)
            System.out.println (m + " == 10");
        else
            System.out.println (m + " != 10");
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Vad händer när någon av de bortkommenterade satserna inkluderas?

Övning 2

```
class AntalSiffror
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("ett positivt heltal: ");
        int    tal = in.nextInt ();

        if (tal < 10)
            System.out.println ("en siffra");
        else if (tal >= 10  &&  tal < 100)
            System.out.println ("två siffror");
        else if (tal >= 100  &&  tal < 1000)
            System.out.println ("tre siffror");
        else
            System.out.println ("fler än tre siffror");

        if (tal < 10)
            System.out.println ("en siffra");
        else if (tal < 100)
            System.out.println ("två siffror");
        else if (tal < 1000)
            System.out.println ("tre siffror");
        else
            System.out.println ("fler än tre siffror");
    }
}
```

a) Vad gör det här programmet?

b) Varför skapas samma utskrift två gånger?

c) Vad händer om ett negativt heltal matas in?

Övning 3

```
class MinstaHeltalet
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("tre heltal: ");
        int    tal1 = in.nextInt ();
        int    tal2 = in.nextInt ();
```

```
        int    tal3 = in.nextInt ();


        int    m = 0;
        if (tal1 < tal2)
        {
            if (tal1 < tal3)
                m = tal1;
            else
                m = tal3;
        }
        else
        {
            if (tal2 < tal3)
                m = tal2;
            else
                m = tal3;
        }
        System.out.println (m);
    }
}
```

a) Vad händer i det här programmet?

b) Följ programmet i fallet att `4 7 5` matas in, och i fallet att `7 5 4` matas in.

c) Beskriv logiken i programmet med ord.

d) Hur kan man testa det här programmet?

e) Hur kan man bestämma det minsta heltalet i fall att fyra heltal matas in?

f) Hur ska programmet justeras om tre bokstäver ska matas in, och den minsta av dem ska bestämmas?

Övning 4

```
class MinstaHeltalet
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("tre heltal: ");
        int    tal1 = in.nextInt ();
        int    tal2 = in.nextInt ();
        int    tal3 = in.nextInt ();

        int    m = tal1;
```

```
        if (tal2 < m)
            m = tal2;
        if (tal3 < m)
            m = tal3;
        System.out.println (m);

        int    m1 = (tal1 < tal2)? tal1 : tal2;
        m = (m1 < tal3)? m1 : tal3;
        System.out.println (m);
    }
}
```

a) Vad händer i det här programmet?

b) Följ programmet i fallet att `4 7 5` matas in, och i fallet att `7 5 4` matas in.

c) Beskriv logiken i programmet med ord.

d) Hur kan man bestämma det minsta heltalet i fall att fyra heltal matas in?

Övning 5

```
class SorteraHeltal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("tre heltal: ");
        int    tal1 = in.nextInt ();
        int    tal2 = in.nextInt ();
        int    tal3 = in.nextInt ();

        int    u = 0, v = 0, w = 0;
        if (tal1 <= tal2)
        {
            if (tal2 <= tal3)
            {
                u = tal1;
                v = tal2;
                w = tal3;
            }
            else if (tal1 <= tal3)
            {
                u = tal1;
                v = tal3;
                w = tal2;
            }
            else
```

```
            {
                u = tal3;
                v = tal1;
                w = tal2;
            }
        }
        else
        {
            if (tal1 <= tal3)
            {
                u = tal2;
                v = tal1;
                w = tal3;
            }
            else if (tal2 <= tal3)
            {
                u = tal2;
                v = tal3;
                w = tal1;
            }
            else
            {
                u = tal3;
                v = tal2;
                w = tal1;
            }
        }

        System.out.println (u + " " + v + " " + w);
    }
}
```

a) Vad händer i det här programmet?

b) Följ programmet i fallet att `4 7 5` matas in, och i fallet att `7 5 4` matas in.

c) Beskriv logiken i programmet med ord.

d) Hur kan man testa det här programmet?

e) Hur kan man utföra sorteringen i fall att fyra heltal matas in?

f) Hur ska programmet justeras om tre bokstäver ska matas in och sorteras?

Övning 6

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

```
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("två heltal: ");
        int    m = in.nextInt ();
        int    n = in.nextInt ();
        in.nextLine ();

        System.out.print ("s, d, p eller k: ");
        String    val = in.nextLine ();

        int   r = 0;
        switch (val)
        {
        case "s":
            r = m + n;
            break;
        case "d":
            r = m - n;
            break;
        case "p":
            r = m * n;
            break;
        case "k":
            r = m / n;
            break;
        }
        System.out.println (r);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer om någon av `break`-satserna bortkommenteras?

c) Hur ska programmet justeras om apostroftecken ska användas i stället för citattecken i `switch`-satsen?

Övning 7

```
class SummanPrimtal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("ett heltal > 1: ");
        int    n = in.nextInt ();

        int    sum = 0;
```

```
        for (int i = 1; i <= n; i++)
            sum += i;
        System.out.println ("summan: " + sum);

        boolean    arPrim = true;
        for (int i = 2; i <= n / 2; i++)
            if (n % i == 0)
            {
                arPrim = false;
                break;
            }
        System.out.println ("ett primtal: " + arPrim);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Hur många gånger utförs den första loopen?

c) Hur många gånger utförs den andra loopen?

Övning 8

```
class Bokstaver
{
    public static void main (String[] args) throws java.io.IOException
    {
        java.io.InputStreamReader    in =
                  new java.io.InputStreamReader (System.in);

        System.out.print (
                     "en liten bokstav, något annat att avsluta: ");
        char    lb = (char) in.read ();
        in.read ();
        in.read ();  // på visa plattformar behövs det inte
        char    sb = 0;
        while (lb >= 'a'  &&  lb <= 'z')
        {
            sb = (char) (lb - 32);
            System.out.println (sb);
            for (char c = 'a'; c <= lb; c++)
                System.out.print (c);
            System.out.println ("\n");

            System.out.print (
               "en liten bokstav, något annat att avsluta: ");
            lb = (char) in.read ();
            in.read ();
            in.read (); // på visa plattformar behövs det inte
```

```
        }
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Hur många gånger utförs den yttre loopen?

c) Hur många gånger utförs den inre loopen?

Övning 9

```
class GissaHeltal
{
    private static final int    FAST_HELTAL = 7;

    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        int    tal = 0;
        do
        {
            System.out.print ("ett ensifrigt heltal: ");
            tal = in.nextInt ();
            if (tal == FAST_HELTAL)
                System.out.println ("rätt gissning: " + tal);
            else if (tal < FAST_HELTAL)
                System.out.println ("högre");
            else if (tal > FAST_HELTAL)
                System.out.println ("lägre");
        } while (tal != FAST_HELTAL);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Hur många gånger utförs loopen?

Övning 10

```
class TriangelHeltal
{
    public static void main (String[] args)
    {
        for (int i = 1; i <= 12; i++)
```

```
        {
            for (int j = 1; j <= i; j++)
                System.out.print (j % 10 + " ");
            System.out.println ();
        }
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Hur många gånger utförs den yttre loopen?

c) Hur många gånger utförs den inre loopen?

Övning 11

```
class TriangelHeltal
{
    public static void main (String[] args)
    {
        int    k = 1;
        int    m = 0;
        for (int i = 1; i <= 5; i++)
        {
            m = 5;
            for (int j = 1; j <= k; j++)
                System.out.print (m-- + " ");
            System.out.println ();

            k++;
        }
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Hur många gånger utförs den yttre loopen?

c) Hur många gånger utförs den inre loopen?

Övning 12

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

```
        int    k = 0;
        for (int i = 1; i <= 5; i++)
        {
            k = 10 * i;
            for (int j = 0; j <= 10; j++)
                System.out.print ((k + j * i) + " ");
            System.out.println ();
        }
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Kapitel 6

Vektorer

Övning 1

```
class VektorerReferenserIndex
{
    public static void main (String[] args)
    {
        int[]    u = new int[5];
        u[0] = 4;
        u[u.length - 1] = 12;
        // u = null;                          // (1)
        for (int i = 0; i < u.length; i++)
            System.out.print (u[i] + " ");
        System.out.println ();

        int[]    v = new int[0];
        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();

        int[]    w = {4, 9, 8, 1, 7};
        for (int i = 1; i <= w.length; i++)
            System.out.print (w[i] + " ");

        int[]    x = null;
        for (int i = 0; i < x.length; i++)
            System.out.print (x[i] + " ");
    }
}
```

a) Rita vektorn `u` och ange index för varje cell.

b) Vad händer när det här programmet exekveras?

c) Vad händer med vektorn (som refereras av referensen) `u` när satsen `(1)` inkluderas?

d) Vilka fel finns i det här programmet?

Övning 2

```
class VandOmVektor
{
    public static void main (String[] args)
    {
        int    n = 6;
        // n = 5;                  // (1)
        int[]    v = new int[n];
        for (int i = 0; i < v.length; i++)
            v[i] = i + 1;

        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();

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

        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Åskådliggör stegen i den loop som vänder om vektorn `v`. På vilket sätt ändras situationen om satsen `(1)` inkluderas?

Övning 3

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.

Övning 4

```
class ReferenskopieringVektorkopiering
{
    public static void main (String[] args)
    {
        int[]    v = new int[5];
        for (int i = 0; i < 5; i++)
            v[i] = i + 1;
        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();

        int[]    u = v;
        u[0] = 5;
        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();

        int[]    w = new int[v.length];
        for (int i = 0; i < v.length; i++)
            w[i] = v[i];

        w[0] = 99;
        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Åskådliggör alla vektorer och referenser som skapas. Vart refererar referenserna?

Övning 5

```
class Alfabet
{
    public static void main (String[] args)
    {
        char[]    b = new char[26];
        for (int i = 0; i < b.length; i++)
            b[i] = (char) (65 + i);

        for (char c : b)
            System.out.print (c);
        System.out.println ();

        for (int i = b.length - 1; i >= b.length / 2; i--)
            System.out.print (b[i]);
        System.out.println ();

        for (int i = b.length / 2 - 1; i >= 0; i--)
            System.out.print (b[i]);
        System.out.println ();
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 6

```
class MaxMedel
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("antalet heltal: ");
        int    antal = in.nextInt ();
        int[]    tal = new int[antal];

        System.out.print ("Heltalen: ");
        for (int i = 0; i < tal.length; i++)
            tal[i] = in.nextInt ();
        System.out.println ();

        int    max = tal[0];
        // max = 0;                // (1)
        for (int n : tal)
            if (n > max)
                max = n;
```

```
        System.out.println (max);

        int    sum = 0;
        for (int n : tal)
            sum += n;
        double    medel = (double) sum / antal;
        System.out.println (medel);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Följ stegen i de olika looparna.

c) Vad händer om satsen `(1)` inkluderas? Mata in bara negativa heltal, och följ utvecklingen.

Övning 7

```
class MataInOlikaHeltal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("antalet heltal: ");
        int    antal = in.nextInt ();
        int[]    tal = new int[antal];

        System.out.println (antal + " olika heltal:");
        int    pos = 0;
        boolean    redanInmatat = false;
        while (pos < tal.length)
        {
            tal[pos] = in.nextInt ();

            redanInmatat = false;    // (1)
            for (int i = 0; i < pos; i++)
                if (tal[pos] == tal[i])
                {
                    redanInmatat = true;
                    break;
                }

            if (!redanInmatat)
                pos++;
        }
        System.out.println ();
```

```
        for (int n : tal)
            System.out.print (n + " ");
        System.out.println ();
    }
}
```

a) Vad händer när det här programmet exekveras? Följ programmet för olika inmatningar.

b) Vad händer om flera likadana heltal anges vid inmatningen? Hur påverkas det om satsen `(1)` bortkommenteras?

Övning 8

```
class GemensammaHeltal
{
    public static void main (String[] args)
    {
        int[]    u = {10, 5, 1, 12, 4, 7
                  // , 4
                     };

        java.util.Scanner    in = new java.util.Scanner (System.in);
        System.out.print ("antalet heltal: ");
        int    antal = in.nextInt ();
        int[]    v = new int[antal];
        System.out.print ("Heltalen: ");
        for (int i = 0; i < v.length; i++)
            v[i] = in.nextInt ();
        System.out.println ();

        int    antalGemensammaHeltal = 0;
        for (int i = 0; i < u.length; i++)
            for (int j = 0; j < v.length; j++)
            {
                if (u[i] == v[j])
                {
                    antalGemensammaHeltal++;
                    break;
                }
            }

        int[]    w = new int[antalGemensammaHeltal];
        int    pos = 0;
        for (int i = 0; i < u.length; i++)
        {
            for (int j = 0; j < v.length; j++)
            {
                if (u[i] == v[j])
                {
```

```
                    w[pos++] = u[i];
                    break;
                }
            }

            if (pos == w.length)    // (1)
                break;
        }

        for (int n : w)
            System.out.print (n + " ");
        System.out.println ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Följ stegen i de två mellersta looparna.

c) På vilket sätt bidrar satsen `(1)`?

Övning 9

```
class SorteraHeltal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("antalet heltal > 2: ");
        int    antal = in.nextInt ();
        int[]    v = new int[antal];

        System.out.print ("Heltalen: ");
        for (int i = 0; i < v.length; i++)
            v[i] = in.nextInt ();
        System.out.println ();

        int   n = 0;
        int    pos = 0;
        for (int p = pos + 1; p < v.length; p++)
            if (v[p] < v[pos])
            {
                n = v[pos];
                v[pos] = v[p];
                v[p] = n;
            }

        for (int k : v)
```

```
            System.out.print (k + " ");
        System.out.println ();

        pos = 1;
        for (int p = pos + 1; p < v.length; p++)
            if (v[p] < v[pos])
            {
                n = v[pos];
                v[pos] = v[p];
                v[p] = n;
            }

        for (int k : v)
            System.out.print (k + " ");
        System.out.println ();

        for (pos = 2; pos < v.length - 1; pos++)
            for (int p = pos + 1; p < v.length; p++)
                if (v[p] < v[pos])
                {
                    n = v[pos];
                    v[pos] = v[p];
                    v[p] = n;
                }

        for (int k : v)
            System.out.print (k + " ");
        System.out.println ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Följ stegen i de olika looparna.

c) Hur kan man utföra sorteringen med mindre kod?

Övning 10

```
class TvadimensionellVektor
{
    public static void main (String[] args)
    {
        int[]    u = new int[4];
        u[0] = 1;
        u[1] = 2;
        u[2] = 3;
        u[3] = 4;
        int[]    v = {5, 6, 7, 8, 9};
```

```
        int[]    w = {10, 11};

        int[][]    n = new int[3][];
        n[0] = u;
        n[1] = v;
        n[2] = w;
        for (int i = 0; i < n.length; i++)
        {
            for (int j = 0; j < n[i].length; j++)
                System.out.print (n[i][j] + " ");
            System.out.println ();
        }
        System.out.println ();

        int    antalElement = 0;
        for (int i = 0; i < n.length; i++)
            antalElement += n[i].length;
        System.out.println (antalElement);

        int[][]    m = n;
        m[0][0] = 100;
        for (int i = 0; i < n.length; i++)
        {
            for (int j = 0; j < n[i].length; j++)
                System.out.print (n[i][j] + " ");
             System.out.println ();
        }
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Åskådliggör de olika vektorerna och referenserna.

Övning 11

```
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.

Övning 12

```
class MaxMedel
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);
        System.out.println ("antalet rader:");
        int    m = in.nextInt ();
        System.out.println ("antalet kolumner:");
        int    n = in.nextInt ();
        double[][]    v = new double[m][n];
        for (int i = 0; i < m; i++)
        {
            System.out.println  ("talen i rad " + i + ":");
            for (int j = 0; j < n; j++)
                v[i][j] = in.nextDouble ();
        }
        System.out.println ();

        for (double[] w : v)
        {
            for (double d : w)
                System.out.print (d + " ");
            System.out.println ();
        }
        System.out.println ();

        double    max = v[0][0];
        for (double[] w : v)
            for (double d : w)
                if (d > max)
                    max = d;
        System.out.println (max);

        double    sum = 0;
        for (double[] w : v)
            for (double d : w)
                sum = sum + d;
        double    medel = sum / (v.length * v[0].length);
        System.out.println (medel);
```

```
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Åskådliggör vektorn `v`.

c) Följ stegen i de olika looparna.

Övning 13

```
class SummaKolumnvis
{
    public static void main (String[] args)
    {
        int[][]   v = new int[4][5];
        int    k = 1;
        for (int i = 0; i < v.length; i++)
        {
            for (int j = 0; j < v[i].length; j++)
                v[i][j] = k + j;

            k++;
        }

        for (int[] w : v)
        {
            for (int n : w)
                System.out.print (n + " ");
            System.out.println ();
        }
        System.out.println ();

        int[]    sum = new int[5];
        for (int j = 0; j < sum.length; j++)
        {
            sum[j] = 0;
            for (int i = 0; i < v.length; i++)
                sum[j] += v[i][j];
        }

        for (int i = 0; i < sum.length; i++)
            System.out.print (sum[i] + " ");
        System.out.println ();
    }
}
```

Vad händer när det här programmet exekveras?

Problem 1

Temperaturmätningar

Ett problem: bearbeta mätresultat

Man gör temperaturmätningar på ett och samma ställe under ett antal veckor. Mätningarna görs ett bestämt antal gånger – lika många mätningar i varje vecka.

På slutet av mätperioden ska de samlade uppgifterna bearbetas: för varje vecka ska den minsta, den största och medeltemperaturen bestämmas. Den minsta, den största och medeltemperaturen ska bestämmas även för hela mätperioden.

En lösning till problemet – ej fullständig

Det här programmet matar in temperaturerna och visar dem. Därefter bestäms och lagras den minsta, den största och medeltemperaturen för varje vecka. Dessa temperaturer skrivs sedan ut till standardutmatningsenheten. Till sist bestäms och lagras den minsta, den största och medeltemperaturen för hela mätperioden. Även dessa temperaturer skrivs ut till standardutmatningsenheten.

```
import java.util.*;     // Scanner, Locale

class Temperaturer
{
    public static void main (String[] args)
    {
        System.out.println ("TEMPERATURER\n");

        // inmatningsverktyg
        Scanner    in = new Scanner (System.in);
        in.useLocale (Locale.US);

        // mata in uppgifter om antalet veckor och antalet mätningar
        System.out.print ("antalet veckor: ");
        int    antalVeckor = in.nextInt ();
        System.out.print ("antalet mätningar per vecka: ");
        int    antalMatningarPerVecka = in.nextInt ();

        // plats att lagra temperaturer
        double[][]    t =
             new double[antalVeckor + 1][antalMatningarPerVecka + 1];

        // mata in temperaturerna
        for (int vecka = 1; vecka <= antalVeckor; vecka++)
        {
```

```
            System.out.println (
                      "temperaturer - vecka " + vecka + ":");
            for (int matning = 1;
                 matning <= antalMatningarPerVecka; matning++)
                t[vecka][matning] = in.nextDouble ();
        }
        System.out.println ();

        // visa temperaturerna
        System.out.println ("temperaturerna:");
        for (int vecka = 1; vecka <= antalVeckor; vecka++)
        {
            for (int matning = 1;
                 matning <= antalMatningarPerVecka; matning++)
                System.out.print (t[vecka][matning] + " ");
            System.out.println ();
        }
        System.out.println ();

        // den minsta, den största och medeltemperaturen - veckovis
        double[]    minT = new double[antalVeckor + 1];
        double[]    maxT = new double[antalVeckor + 1];
        double[]    sumT = new double[antalVeckor + 1];
        double[]    medelT = new double[antalVeckor + 1];
        // koden ska skrivas här

        // visa den minsta, den största och medeltemperaturen
        // för varje vecka
        // koden ska skrivas här

        // den minsta, den största och medeltemperaturen - hela
        // mätperioden
        double    minTemp = minT[1];
        double    maxTemp = maxT[1];
        double    sumTemp = sumT[1];
        double    medelTemp = 0;
        // koden ska skrivas här

        // visa den minsta, den största och medeltemperaturen i
        // hela mätperioden
        // koden ska skrivas här
    }
}
```

Uppgifter i samband med temperaturmätningar

1. Skapa en tabell som innehåller möjliga temperaturer, både de som erhålls genom mätningarna och de som beräknas. Tabellen ska vara av följande form:

vecka	*mätning 1*	*mätning 2*	*mätning 3*	*minT*	*maxT*	*sumT*	*medT*
1							
2							
				min-Temp	*max-Temp*	*sum-Temp*	*med-Temp*

2. Komplettera programmet `Temperaturer`: lägg till den kod som bestämmer och visar de minsta, största och medeltemperaturerna. Kör programmet flera gånger med olika data, och kontrollera om de erhållna resultaten är riktiga.

3. Rita den vektor där de temperaturer som erhålls genom mätningarna lagras. Hur kommer man åt en viss uppgift i denna vektor? Rita även de vektorer och variabler där de erhållna resultaten lagras.

När en vektor ritas ska både dess referenser, minnesceller och de lagrade uppgifterna finnas med. Det ska framgå vad de enskilda referenserna heter. När en variabel ritas ska den data som lagras och variabelns namn finnas med.

4. Vilken strategi använder man för att bestämma den minsta temperaturen? Illustrera denna strategi: rita en serie bilder som visar hur man kommer fram till den minsta temperaturen.

Kapitel 7
Metoder

Övning 1

```
class TeckenIntervall
{
    public static void main (String[] args)
    {
        teckenIntervall ('a', 'm');
        teckenIntervall ('m', 'a');
        teckenIntervall ('a', 'a');
        char    c1 = 'n';
        char    c2 = 'z';
        teckenIntervall (c1, c2);
        System.out.println ();

        for (char c = 'a'; c <= 'e'; c++)
            teckenIntervall ('a', c);
        System.out.println ();

        teckenIntervall ('0', '9');
    }

    public static void teckenIntervall (char c1, char c2)
    {
        System.out.print ("[");
        for (char c = c1; c < c2; c++)
            System.out.print (c + ", ");
        if (c1 <= c2)
            System.out.print (c2);
        System.out.println ("]");
    }
}
```

a) Vilken utskrift ger det här programmet?

b) Är variabeln `c1` i metoden `main` och parametern `c1` i metoden `teckenInterval` egentligen en och samma variabel?

Övning 2

```
class Primtal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("Ett heltal > 1: ");
        int    tal = in.nextInt ();
        primtal (tal);
    }

    public static void primtal (int n)
    {
        if (n < 2)
        {
            System.out.println ();
            return;
        }

        boolean    arPrim = true;
        for (int k = 2; k <= n; k++)
        {
            arPrim = true;
            for (int i = 2; i <= k/2; i++)
            {
                if (k % i == 0)
                {
                    arPrim = false;
                    break;
                }
            }

            if (arPrim)
                System.out.print (k + " ");
        }
        System.out.println ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vilket är samband mellan argumentet `tal` och parametern `n`?

c) Följ noggrant stegen i metoden `primtal`.

Övning 3

```
class Primtal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("Ett heltal > 1: ");
        int    tal = in.nextInt ();
        primtal (tal);
    }

    public static void primtal (int n)
    {
        boolean    arPrimTal = true;
        for (int k = 2; k <= n; k++)
        {
            arPrimTal = arPrim (k);
            if (arPrimTal)
                System.out.print (k + " ");
        }
        System.out.println ();
    }

    public static boolean arPrim (int n)
    {
        if (n < 2)
            return false;

        boolean    arPrim = true;
        for (int k = 2; k <= n/2; k++)
        {
            if (n % k == 0)
            {
                arPrim = false;
                break;
            }
        }

        return arPrim;
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Hur många gånger anropas metoden `arPrim`?

c) Vad händer när `return`.satsen i metoden `arPrim` exekveras? Vart går det värde som returneras?

Övning 4

```
class GemenerVersaler
{
    public static void main (String[] args)
    {
        char[]    u = {'a', 'B', 'c', 'D', 'e', '1', '5', '+' , 'k'};
        System.out.println (u);
        System.out.println ();

        char[]    g = baraGemener (u);
        System.out.println (g);
        char[]    v = tillVersaler (u);
        System.out.println (v);
        System.out.println (u);
    }

    public static boolean arGemen (char c)
    {
        boolean    b = false;
        if (c >= 'a'  &&  c <= 'z')
            b = true;

        b = b  ||  c == 'å'  ||  c == 'ä'  ||  c == 'ö';

        return b;
    }

    public static char[] baraGemener (char[] v)
    {
        int    antalGemener = 0;
        for (int i = 0; i < v.length; i++)
            if (arGemen (v[i]))
                antalGemener++;

        char[]    gem = new char [antalGemener];
        int    pos = 0;
        for (int i = 0; i < v.length; i++)
            if (arGemen (v[i]))
                gem[pos++] = v[i];

        return gem;
    }

    public static char tillVersal (char c)
    {
        char    ver = c;
        if (c >= 'a'  &&  c <= 'z')
            ver = (char) (c - 32);
        else if (c == 'å')
```

```
            ver = 'Å';
        else if (c == 'ä')
            ver = 'Ä';
        else if (c == 'ö')
            ver = 'Ö';

        return ver;
    }

    public static char[] tillVersaler (char[] v)
    {
        char[]    ver = new char [v.length];
        for (int i = 0; i < v.length; i++)
            ver[i] = tillVersal (v[i]);

        return ver;
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Följ noggrant stegen i metoden `tillVersaler`.

c) Följ noggrant stegen i metoden `baraGemener`.

d) Är referensen `u` i metoden `main` och parametern `v` i metoden `baraGemener` egentligen en och samma referens? Vart pekar dessa referenser? Rita motsvarande bild.

e) Är referensen `gem` i metoden `baraGemener` och variabeln `g` i metoden `main` egentligen en och samma referens? Vart pekar dessa referenser? Rita motsvarande bild.

Övning 5

```
class MaxMedel
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("antalet heltal: ");
        int    antal = in.nextInt ();
        int[]    tal = new int[antal];

        System.out.print ("Heltalen: ");
        for (int i = 0; i < tal.length; i++)
            tal[i] = in.nextInt ();
        System.out.println ();

        int    mx = max (tal);
```

```
        System.out.println (mx);

        double    md = medel (tal);
        System.out.println (md);
        System.out.println ();

        System.out.print ("tre heltal: ");
        int    n1 = in.nextInt ();
        int    n2 = in.nextInt ();
        int    n3 = in.nextInt ();

        mx = max (n1, n2, n3);
        System.out.println (mx);

        md = medel (n1, n2, n3);
        System.out.println (md);
    }

    public static int max (int... v)
    {
        int    max = v[0];
        for (int n : v)
            if (n > max)
                max = n;

        return max;
    }

    public static double medel (int... v)
    {
        int    sum = 0;
        for (int n : v)
            sum += n;
        double    med = (double) sum / v.length;

        return med;
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer när metoden `max` anropas?

c) Vad händer när metoden `medel` anropas?

d) Vad händer i fall att `int[] v` används som parameter i metoderna `max` och `medel`?

Övning 6

```
class SorteraTal
{
    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("antalet heltal: ");
        int    antal = in.nextInt ();
        int[]    tal = new int[antal];

        System.out.print ("Heltalen: ");
        for (int i = 0; i < tal.length; i++)
            tal[i] = in.nextInt ();
        System.out.println ();

        sortera (tal);

        for (int n : tal)
            System.out.print (n + " ");
        System.out.println ();
    }

    public static void sortera (int[] v)
    {
        int    n = 0;
        for (int pos = 0; pos < v.length - 1; pos++)
            for (int p = pos + 1; p < v.length; p++)
                if (v[p] < v[pos])
                {
                    n = v[pos];
                    v[pos] = v[p];
                    v[p] = n;
                }
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Åskådliggör stegen i metoden `sortera`.

c) Behöver metoden `sortera` returnera vektorn `v`? Ska returtypen vara `void` eller `int[]`?

Övning 7

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 och c = 3.5.

Övning 8

```
class ReferenserVektorer
{
    public static void main (String[] args)
    {
        int[]    v = {10, 20, 30, 40};

        int[]    u = paverka (v);
        System.out.println ();

        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();

        for (int i = 0; i < u.length; i++)
            System.out.print (u[i] + " ");
        System.out.println ();
    }

    public static int[] paverka (int[] v)
    {
        v[0] = 11;
        v = new int[5];
        v[1] = 22;

        for (int i = 0; i < v.length; i++)
            System.out.print (v[i] + " ");
        System.out.println ();

        return v;
    }
}
```

Vilken utskrift skapas när det här programmet exekveras?

Övning 9

a)

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

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

b)

En metod är given:

```
public static void storreElementTillHoger (int[] v)
{
    int    e = 0;
    for (int pos = 0; pos < v.length - 1; pos++)
    {
        if (v[pos] > v[pos + 1])
        {
            e = v[pos];
            v[pos] = v[pos + 1];
            v[pos + 1] = e;
        }
    }
}
```

Metoden anropas så här:

```
int[]    u = {4, 2, 3, 0, 5, 6, 1};
storreElementTillHoger (u);
```

Hur ser den skapade vektorn ut efter metodanropet: rita både vektorn och motsvarande referens.

Övning 10

a)

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

```
v[0][0] = 1;
int    m = v.length - 1;
int    n = v[m].length - 1;
v[m/2][n/2] = 2;
v[m][n] = 3;
```

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.

b)

En metod är given:

```
public static void utbytRader(int[][] v)
{
    int[]    p = v[0];
    v[0] = v[v.length - 1];
    v[v.length - 1] = p;
}
```

Metoden anropas så här:

```
int[][]    v = new int[3][5];
v[0][0] = 1;
v[1][1] = 2;
v[2][2] = 3;
utbytRader (v);
```

Hur ser den skapade vektorn ut efter metodanropet: rita både vektorn och motsvarande referenser.

Övning 11

```
class Klot
{
    public static double    PI = 3.14;

    public static void main (String[] args)
    {
        java.util.Scanner    in = new java.util.Scanner (System.in);

        System.out.print ("klotets radie: ");
        double    r = in.nextDouble ();
        double    ar = area (r);
        double    vol = volym (r);
        System.out.println ();

        System.out.println ("area: " + ar);
        System.out.println ("volym: " + vol);
```

```
    }

    public static double area (double r)
    {
        return 4 * PI * r * r;
    }

    public static double volym (double r)
    {
        return 4 * PI * r * r * r / 3;
    }
}
```

Vad händer när det här programmet exekveras?

Övning 12

```
class NerOchUpp
{
    public static void main (String[] args)
    {
        nerOchUpp (5);
        System.out.println ();
    }

    public static void nerOchUpp (int n)
    {
        if (n < 1)
            return;                    // (1)

        System.out.print (n + " ");    // (2)
        nerOchUpp (n - 1);
        System.out.print (n + " ");    // (3)
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer om satsen `(3)` bortkommenteras?

c) Vad händer om satsen `(2)` bortkommenteras?

d) Vad händer om satsen `(1)` bortkommenteras?

e) Följ anropskedjan: vad händer i varje enskilt anrop av metoden `nerOchUpp`?

Övning 13

```
class SummaRekursivt
{
    public static void main (String[] args)
    {
        java.util.Scanner  in = new java.util.Scanner (System.in);

        System.out.print ("antalet heltal att summeras: ");
        int    n = in.nextInt ();

        int    s = sum (n);
        for (int i = 1; i < n; i++)
            System.out.print (i + " + ");
        System.out.println (n + " = " + s);
    }

    public static int sum (int n)
    {
        int   s = 0;
        if (n == 1)
            s = 1;
        else
            s = sum (n - 1) + n;

        return s;
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Följ anropskedjan i fall att heltalet `5` matas in.

c) Implementera metoden `sum` utan rekursion: iterativt.

Kapitel 8

Ett klassbibliotek

Övning 1

```
// Sorterare.java

public class Sorterare
{
    public static void sortera (int[] element)
    {
        int    sist = element.length - 1;
        int    pos = 0;
        int    e = 0;
        while (pos < sist)
        {
            for (int p = sist; p > pos; p--)
                if (element[p] < element[p - 1])
                {
                    e = element[p - 1];
                    element[p - 1] = element[p];
                    element[p] = e;
                }

            pos++;
        }
    }

    public static void sortera (double[] element)
    {
        int    sist = element.length - 1;
        int    pos = 0;
        double    e = 0;
        while (pos < sist)
        {
            for (int p = sist; p > pos; p--)
                if (element[p] < element[p - 1])
                {
                    e = element[p - 1];
                    element[p - 1] = element[p];
                    element[p] = e;
```

```
                }

            pos++;
        }
    }
}


// AnvandSorterare.java

import java.util.Scanner;

class AnvandSorterare
{
    public static void main (String[] args)
    {
        Scanner  in = new Scanner (System.in);

        System.out.print ("antalet tal: ");
        int    antal = in.nextInt ();
        double[]    tal = new double[antal];

        System.out.print ("Talen: ");
        for (int i = 0; i < tal.length; i++)
            tal[i] = in.nextDouble ();
        System.out.println ();

        Sorterare.sortera (tal);

        for (double n : tal)
            System.out.print (n + " ");
        System.out.println ();
    }
}
```

a) Vad händer när programmet `AnvandSorterare` exekveras?

b) Vilka klasser och metoder används i programmet `AnvandSorterare`? Använder man de båda två metoderna i klassen `Sorterare`?

c) Åskådliggör stegen i metoden `sortera`.

Övning 2

```
// Letter.java

/*********************************************************
```

```
Letter hanterar bokstäver i det engelska alfabetet.

Det går att kontrollera om ett tecken är en bokstav. Det går
även att kontrollera huruvida ett tecken är en liten bokstav
eller en stor bokstav.

Om ett tecken är en liten bokstav kan den motsvarande stora
bokstaven erhållas, och tvärtom. Det går även att erhålla
en bokstavs föregångare och efterföljare.

*********************************************************/

package fjava.edu;

import java.util.*;  // NoSuchElementException

public class Letter
{
    // isLetter returnerar true om det givna tecknet
    // representerar en bokstav
    public static boolean isLetter (char c)
    {
        return c >= 'A'  &&  c <= 'Z'
           ||  c >= 'a'  &&  c <= 'z';
    }

    // isUpperCase returnerar true om det givna tecknet
    // representerar en stor bokstav
    public static boolean isUpperCase (char c)
    {
        return c >= 'A'  &&  c <= 'Z';
    }

    // isLowerCase returnerar true om det givna tecknet
    // representerar en liten bokstav
    public static boolean isLowerCase (char c)
    {
        return c >= 'a'  &&  c <= 'z';
    }

    // toUpperCase tar emot ett tecken. Om detta tecken är
    // en liten bokstav returneras motsvarande stor bokstav,
    // annars returneras det givna tecknet.
    public static char toUpperCase (char c)
    {
        char    ch = c;
        if (isLowerCase (c))
            ch = (char) (c - 32);

        return ch;
    }
```

```
    // toLowerCase tar emot ett tecken. Om detta tecken är
    // en stor bokstav returneras motsvarande liten bokstav,
    // annars returneras det givna tecknet.
    public static char toLowerCase (char c)
    {
        char    ch = c;
        if (isUpperCase (c))
            ch = (char) (c + 32);

        return ch;
    }

    // previousLetter tar emot ett tecken. Om detta tecken är
    // en bokstav, returneras den bokstav som kommer direkt
    // före denna. I fallet att det inte finns en sådan bokstav,
    // och i fallet att tecknet inte är en bokstav, kastas
    // ett undantag av typen java.util.NoSuchElementException.
    public static char previousLetter (char c)
                          throws NoSuchElementException
    {
        boolean    previousLetterExists = c >= 'B'  &&  c <= 'Z'
                                       ||  c >= 'b'  &&  c <= 'z';
        if (!previousLetterExists)
             throw new NoSuchElementException ("no such letter");

        return --c;
    }

    // nextLetter tar emot ett tecken. Om detta tecken är
    // en bokstav, returneras den bokstav som kommer direkt
    // efter denna. I fallet att det inte finns en sådan bokstav,
    // och i fallet att tecknet inte är en bokstav, kastas
    // ett undantag av typen java.util.NoSuchElementException.
    public static char nextLetter (char c)
                          throws NoSuchElementException
    {
        boolean    nextLetterExists = c >= 'A'  &&  c <= 'Y'
                                   ||  c >= 'a'  &&  c <= 'y';
        if (!nextLetterExists)
            throw new NoSuchElementException ("no such letter");

        return ++c;
    }
}


// AnvandLetter.java

import java.io.*;      // InputStreamReader
import fjava.edu.*;    // Letter
```

```
import static java.lang.System.out;

class AnvandLetter
{
    public static void main (String[] args) throws IOException
    {
        InputStreamReader    in = new InputStreamReader (System.in);
        System.out.print ("ett tecken: ");
        char    c = (char) in.read ();

        boolean    arBokstav = Letter.isLetter (c);
        boolean    arLitenBokstav = Letter.isLowerCase (c);
        boolean    arStorBokstav = Letter.isUpperCase (c);
        out.println ("är bokstav: " + arBokstav);
        out.println ("är liten bokstav: " + arLitenBokstav);
        out.println ("är stor bokstav: " + arStorBokstav);
        out.println ();

        if (arBokstav)
        {
            out.print (Letter.previousLetter (c) + " ");
            if (arLitenBokstav)
                out.print (Letter.toUpperCase (c) + " ");
            else
                out.print (Letter.toLowerCase (c) + " ");
            out.println (Letter.nextLetter (c));
        }
    }
}
```

a) Vilka klasser och metoder används i programmet `AnvandLetter`? Var finns dessa klasser: i vilka paket?

b) Vad händer om tecknet `a` matas in när det här programmet exekveras? Är det någon skillnad om tecknet `z`, eller tecknet `M`, matas in? Vad händer om tecknet `+` matas in?

c) Hur ska programmet anpassas om `import`-satserna inte ska användas?

Övning 3

```
import java.util.Scanner;
import static java.lang.System.out;

class Triangeln
{
    public static void main (String[] args)
    {
        Scanner  in = new Scanner (System.in);
```

```
        out.print ("två sidor i en triangel med spetsiga vinklar: ");
        double    a = in.nextDouble ();
        double    b = in.nextDouble ();
        out.print ("vinkeln (i grader) mellan sidorna: ");
        double    gamma = in.nextDouble ();
        out.println ();

        double    c = Math.sqrt (
                Math.pow (a, 2) + Math.pow (b, 2)
                - 2 * a * b * Math.cos (Math.toRadians (gamma)));
        out.println ("den tredje sidan är: " + c);
        out.println ("triangelns omkrets: " + (a + b + c));

        double    h = a * Math.sin (Math.toRadians (gamma));
        double    area = b * h / 2;
        out.println ("triangelns area: " + area);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vilka klasser och metoder används i programmet?

Övning 4

```
import java.util.Arrays;
import static java.lang.System.out;

class Siffror
{
    public static void main (String[] args)
    {
        int    m = (int) (5 * Math.random () + 6);
        char[]    v = new char[m];
        for (int pos = 0; pos < v.length; pos++)
            v[pos] = (char) ((int) (48 + 10 * Math.random ()));
        String    s1 = java.util.Arrays.toString (v);
        out.println (s1);
        out.println ();

        String    s2 = String.valueOf (v);
        out.println (s2);
        long    n = Long.parseLong (s2);
        double    r = Math.cbrt (n);
        out.println (r);
        out.println ();

        java.util.Arrays.sort (v);
```

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

a) Vad händer när det här programmet exekveras?

b) Vilka klasser och metoder används i programmet?

Övning 5

```
// Mat.java

public class Mat
{
    public static double sqrt (double d)
                          throws ArithmeticException
    {
        if (d < 0)
            throw new ArithmeticException ("no root");

        return Math.sqrt (d);
    }
}


// KvadratRot.java

import java.util.Scanner;
import static java.lang.System.out;

class KvadratRot
{
    public static void main (String[] args)
    {
        Scanner  in = new Scanner (System.in);
        out.println ("ett tal: ");
        double    tal = in.nextDouble ();
        out.println ();

        double    rot1 = Math.sqrt (tal);
        out.println (rot1);
        double    rot2 = Mat.sqrt (tal);
        out.println (rot2);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer när ett negativt tal matas in?

Problem 1

Beräkningar i samband med en triangel

Bestämma olika uppgifter om en triangel

I olika sammanhang kan olika uppgifter om en triangel behöva bestämmas. Man kan behöva bestämma längder av en triangels sidor och höjder, eller dess vinklar, eller dess omkrets och area. I andra situationer kan längder av en triangels bisektriser och medianer behöva bestämmas, eller radien av den cirkel som är inskriven i triangeln eller omskriven kring den, eller något annat.

För att kunna bestämma en uppgift behöver man känna till andra uppgifter i samband med en triangel. Om till exempel längderna av en triangels sidor är kända, kan triangelns omkrets och area bestämmas. Det går även att bestämma triangelns vinklar, längder av höjder och andra uppgifter. Ofta går det att använda en formel, stoppa in de nödvändiga uppgifterna, och beräkna det som behövs.

Man kan skapa en samling formler som gäller en triangel. Så snart man behöver beräkna någonting, ska man välja rätt formel och använda den. På så sätt underlättas beräkningar i samband med trianglar betydligt. Man tillför vissa uppgifter till formeln, och erhåller tillbaka det som önskas. Man kan till och med skapa en programenhet med flera metoder. En metod motsvarar till en formel: de nödvändiga uppgifterna tillförs metoden, och den önskade uppgiften erhålls från metoden. Så snart man behöver beräkna någonting, väljer man rätt metod och anropar den, och tar emot den uppgift som returneras. Man bygger in formlerna i motsvarande metoder.

De olika formlerna som gäller en triangel kan hittas på:

http://sv.wikipedia.org/wiki/Triangel

Uppgifter i samband med en triangel

1. En klass `Triangel` ska skapas: klassen ska innehålla flera statiska metoder som utför olika beräkningar i samband med en triangel. En metod tar emot vissa uppgifter, och beräknar och returnerar någon annan uppgift. Till exempel kan en metod ta emot längden av en triangels sida och längden av motsvarande höjd, och returnera triangelns area. En annan metod kan ta emot längderna av alla tre sidorna, och returnera antingen triangelns omkrets, dess area, en av triangelns medianer eller bisektriser, eller något annat.

2. I fall att längder av två sidor i en triangel och vinkeln mellan dessa sidor är givna, kan längden av motsvarande bisektris beräknas med följande formel:

$bis = (2bc \cos(\alpha/2)) / (b + c)$

I denna formel är b och c längder av två sidor i en triangel, α är vinkeln mellan dessa sidor, och bis längden av den bisektris som delar vinkeln mellan sidorna i två lika delar.

Klassen `Triangel` kan innehålla en metod som beräknar längden av en bisektris:

```
// bisektris tar emot två sidor i en triangel och vinkeln (i radianer)
// mellan dessa sidor. Metoden returnerar längden av den motsvarande
// bisektrisen - den som delar den givna vinkeln i två lika delar.
public static double bisektris (double b, double c, double alfa)
{
    double    p = 2 * b * c * Math.cos (alfa / 2);
    double    bis = p / (b + c);

    return bis;
}
```

Anta att man känner till längder av alla tre sidor i en triangel och storlek av alla tre vinklar. Kan man i så fall bestämma längder av alla tre bisektriser med metoden `bisektris`? Kanske ska tre separata metoder finnas i klassen `Triangel` – en metod per bisektris?

3. Ett program `EnTriangelOchDessCirklar` ska skapas: programmet ska mata in längderna av en triangels sidor, och bestämma radien av den cirkel som är omskriven kring triangeln, samt radien av den cirkel som är inskriven i triangeln. För dessa beräkningar ska passande metoder i klassen `Triangel` användas: man ska anropa den metod som bestämmer radien för den omskrivna cirkeln och den metod som bestämmer radien för den inskrivna cirkeln.

4. Rita en triangel med de sidolängder som används vid en exekvering av programmet `EnTriangelOchDessCirklar`. Rita även de två cirklarna: den som är omskriven och den som är inskriven. Mät cirklarnas radie, och kontrollera om de erhållna uppgifterna motsvarar de uppgifter som man får med programmet `EnTriangelOchDessCirklar`.

Kapitel 9
Algoritmer

Övning 1

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)
{
    m = 1
    i = 2
    while i <= n
    {
        if (x_i > x_m)
            m = i
        i++
    }

    return x_m
}
```

Spåra denna algoritm i samband med följande mängd:

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

Relevanta data ska samlas i en tabell av följande form:

i	x_i	m	x_m
–	–	1	4
2			

Övning 2

En metod `sok` söker ett givet element i en given sekvens:

```
public static int sok (int[] element, int e)
{
    int    elementPos = -1;

    int    pos = 0;
    while (pos < element.length)
    {
        if (e == element[pos])
        {
            elementPos = pos;
            break;
        }

        pos++;
    }

    return elementPos;
}
```

En elementjämförelse kan betraktas som en elementär operation i den algoritm som används i metoden `sok`. Man kan anta att det finns n $(n \in N, n > 0)$ element i sekvensen. Bestäm i så fall algoritmens tidskomplexitet i följande fall:

a) i bästa fall

b) i värsta fall

c) i ett genomsnittligt fall.

För att bestämma komplexiteten i ett genomsnittligt fall, anta att följande villkor är uppfyllda: det sökta elementet finns i sekvensen, alla element i sekvensen är olika, och sannolikheten att elementet finns på en given position är lika för all positioner.

Tidskomplexiteten ska anges med en komplexitetsfunktion, och det ska framgå hur denna komplexitetsfunktion erhålls.

Övning 3

ETT PROBLEM

Det finns i ett trafiksystem en startstation, en destinationsstation, och n mellanstationer. Mellanstationerna är numrerade från och med 1 till och med n, där n är ett positivt heltal.

Det går att ta sig från startstationen till destinationsstationen genom vilken som helst av mellanstationerna. Längden av vägen mellan startstationen och mellanstationen nummer i ($i = 1, 2, ..., n$) är a_i. Längden av vägen mellan mellanstationen nummer i ($i = 1, 2, ..., n$) och destinationsstationen är b_i.

En mellanstation ska väljas, så att vägen mellan startstationen och destinationsstationen blir så kort som möjligt.

EN ALGORITM

a) Bestäm en instans av problemet, och åskådliggör den.

b) Hitta en minneseffektiv algoritm som löser det givna problemet. Beskriv denna algoritm med pseudokod. Bestäm algoritmens minneskomplexitet.

c) Hur kan man få en algoritm, som löser givna problemet, vars minneskomplexitet är $\theta(n)$?

Övning 4

PROBLEMET

En sekvens med element är given. Invertera sekvensen: utbyt plats på det första och det sista elementet, på det andra och det näst sista elementet, och så vidare.

ALGORITMEN INVERTERING

FÖRVILLKOR

En sekvens med element är given.

EFTERVILLKOR

Sekvensen är inverterad.

ALGORITMEN

Sätt en pekare som heter `framP` att peka till första positionen i sekvensen, och en pekare `bakP` att peka till sista positionen i sekvensen.

Upprepa följande så länge pekaren `framP` ligger till vänster av pekaren `bakP`.

Utbyt plats på de utpekade elementen: flytta det element som pekas med pekaren `framP` till en extra plats som heter `e`, flytta det element som pekas med pekaren `bakP` till den plats som pekas med `framP`, och till sist flytta elementet på platsen `e` till den plats som pekas med `bakP`.

Flytta pekaren `framP` ett steg framåt och pekaren `bakP` ett steg bakåt.

UPPGIFTER

a) Formulera två instanser av problemet: i den ena ska sekvensen innehålla ett udda antal element, och i den andra ett jämnt antal element. Åskådliggör algoritmen i samband med dessa instanser.

b) Formulera algoritmen med pseudokod.

c) Implementera algoritmen – skapa motsvarande metod i Java. Testa metoden. Sätt in lämpliga spårutskrifter i metoden och följ transformationer av sekvensen steg för steg.

d) Bestäm algoritmens minneskomplexitet. Hur många extra minnesceller används i algoritmen?

e) Bestäm även tidskomplexiteten – ett elementutbyte ska uppfattas som en elementär operation. Hur många gånger utförs den elementära operationen om sekvensen innehåller hundra element? Vad händer med tidskomplexiteten om en elementförflyttning (tilldelning i den motsvarande metoden) väljs som en elementär operation?

f) Kategorisera motsvarande komplexitetsfunktioner: till vilka Θ–mängder hör funktionerna?

g) Bevisa att följande påstående är en loopinvariant till algoritmen: de element som finns framför pekaren `framP` har utbytt plats med motsvarande element i sekvensen.

h) Utgå ifrån loopinvarianten och bevisa algoritmen.

Övning 5

PROBLEMET

En icke-tom tabell med element är given. Invertera tabellen: utbyt plats på den första och den sista raden, på den andra och den näst sista raden, och så vidare.

ALGORITMEN INVERTERING

FÖRVILLKOR

En icke-tom tabell med element är given.

EFTERVILLKOR

Tabellen är inverterad: den första och den sista raden har utbytt plats, den andra och den näst sista raden utbytte plats, och så vidare.

ALGORITMEN IMPLEMENTERAD SOM EN METOD I JAVA

```
public static void invertera (int[][] tal)
{
    int    antalRader = tal.length;
    int    antalKolumner = tal[0].length;
    int[]    t = new int[antalKolumner];
```

```
    int    rad = 0;
    while (rad <= antalRader / 2)
    {
        for (int kolumn = 0; kolumn < antalKolumner; kolumn++)
            t[kolumn] = tal[rad][kolumn];

        for (int kolumn = 0; kolumn < antalKolumner; kolumn++)
            tal[rad][kolumn] = tal[antalRader - 1 - rad][kolumn];

        for (int kolumn = 0; kolumn < antalKolumner; kolumn++)
            tal[antalRader - 1 - rad][kolumn] = t[kolumn];

        rad++;
    }
}
```

UPPGIFTER

a) Inverteringsalgoritmen innehåller ett fel. Spåra algoritmen, och hitta och korrigera felet. Spåra algoritmen både med papper och penna, och med datorns hjälp.

b) Formulera inverteringsalgoritmen med ord.

c) Bestäm algoritmens minneskomplexitet. Bestäm även tidskomplexiteten – en tilldelning ska uppfattas som en elementär operation. Kategorisera motsvarande komplexitetsfunktioner.

d) Modifiera algoritmen – utför elementutbyten på ett minneseffektivare sätt. Bestäm minneskomplexiteten och tidskomplexiteten även i detta fall.

e) Hur mycket minne sparas med den nya algoritmen i fall att tabellen innehåller hundra rader och hundra kolumner?

f) Bevisa algoritmen.

g) Skapa en ny algoritm som inverterar tabellen kolumnvis.

Övning 6

PROBLEMET

En kvadratisk tabell med tal är given. Beräkna summan av talen på huvuddiagonalen.

ALGORITMEN SUMMA

FÖRVILLKOR

En kvadratisk tabell med tal är given.

EFTERVILLKOR

Summan av talen på huvuddiagonalen är beräknad.

TVÅ OLIKA SUMMERINGSALGORITMER

ALGORITM 1

```
huvudDiagonalSumma (tabell med tal)
{
    summa = 0
    för varje tal i tabellen
       om talet ligger på huvuddiagonalen
           summa = summa + tal

    return summa
}
```

ALGORITM 2

```
huvudDiagonalSumma (tabell med tal)
{
    summa = 0
    för varje tal på huvuddiagonalen i tabellen
       summa = summa + tal

    return summa
}
```

UPPGIFTER

a) Använd de båda två algoritmerna i samband med en instans av problemet.

b) Implementera dessa algoritmer som metoder i Java, och testa metoderna.

c) Bestäm tidskomplexiteten för algoritmerna om en summering uppfattas som en elementär operation. Kategorisera motsvarande komplexitetsfunktioner. Hur många gånger utförs den elementära operationen om tabellen innehåller hundra rader?

d) Bestäm tidskomplexiteten för algoritmerna i fall att en kontroll huruvida ett tal ligger på huvuddiagonalen uppfattas som en elementär operation. Kategorisera motsvarande komplexitetsfunktioner. Hur många gånger utförs den elementära operationen om tabellen innehåller hundra rader?

e) Formulera lämpliga loopinvarianter och bevisa algoritmerna.

Övning 7

PROBLEMET

Två positiva heltal är givna. Bestäm heltalens minsta gemensamma multipel.

ALGORITMEN MINSTA GEMENSAMMA MULTIPEL

FÖRVILLKOR

Två positiva heltal är givna.

EFTERVILLKOR

Heltalens minsta gemensamma multipel är bestämd.

EN ALGORITM SOM BESTÄMMER DEN MINSTA GEMENSAMMA MULTIPELN

```
minstaGemensammaMultipel (tal m, tal n)
{
    mgm = (m > n)? m : n
    while (!(mgm delbar med m  &&  mgm delbar med n))
        mgm++;

    return mgm;
}
```

UPPGIFTER

a) Använd algoritmen i samband med flera instanser av problemet.

b) Implementera algoritmen som en metod i Java, och testa metoden.

c) Delbarhetskontrollen i loopen kan uppfattas som en elementär operation i algoritmen. Hur många gånger utförs denna operation i bästa fallet, och hur många gånger i värsta fallet? Vad händer om heltalen är 35 och 14, och i fallet att heltalen är 5 och 7?

d) Hitta en annan algoritm som har bättre tidskomplexitet. Vad händer i så fall om heltalen är 35 och 14?

e) Bevisa de båda två algoritmerna.

Övning 8

PROBLEMET

Två sekvenser med heltal är givna.

Markera de heltal i den första sekvensen som är unika för den sekvensen: de finns inte i den andra sekvensen.

ALGORITMEN MARKERING

FÖRVILLKOR

Två sekvenser med heltal är givna.

EFTERVILLKOR

De element i den första sekvensen som är unika för den sekvensen är markerade.

EN ALGORITM IMPLEMENTERAD SOM EN METOD I JAVA

```
public static char[] unikaElement (int[] u, int[] v)
{
    char[]    unik = new char[u.length];
    for (int i = 0; i < u.length; i++)
        unik[i] = 'g';  // 'g' för gemensam
    for (int i = 0; i < u.length; i++)
    {
        int    j = 0;
        while (j < v.length  &&  v[j] != u[i])
            j++;
        if (j < v.length)
            unik[i] = 'u';  // 'u' för unik
```

```
    }

    return unik;
}
```

UPPGIFTER

a) Algoritmen innehåller ett fel. Spåra algoritmen både med penna och paper, och med datorn. Hitta felet och rätta det.

b) Formulera algoritmen med pseudokod.

c) Bestäm algoritmens tidskomplexitet i bästa fallet och i värsta fallet.

d) Bevisa algoritmen.

Övning 9

PROBLEMET

En tabell med jämförbara element är given.

Markera de elemnt i tabellen som är unika – ett sådant element förekommer bara på ett ställe i tabellen.

UPPGIFTER

a) Hitta en algoritm som löser problemet.

b) Spåra algoritmen i samband med olika instanser av problemet.

c) Formulera algoritmen med pseudokod.

d) Implementera algoritmen som en metod i Java, och testa den metoden.

e) Bestäm algoritmens tidskomplexitet i bästa fallet och i värsta fallet. Kan man utföra markeringen på ett effektivare sätt?

f) Bevisa algoritmen.

Övning 10

PROBLEMET

En icke-tom sekvens med element är given. Rotera sekvensen: flytta varje element ett givet antal steg framåt. När elementet på sista positionen flyttas ett steg fram, dyker det upp i början av sekvensen – det är en cirkulär förflyttning (en rotation).

ALGORITMEN ROTATION

FÖRVILLKOR

En icke-tom sekvens med element är given.

EFTERVILLKOR

Sekvensen är roterad det angivna antalet steg.

TVÅ OLIKA ALGORITMER IMPLEMENTERADE SOM METODER I JAVA

```
public static void rotera (int[] tal, int n)
{
    for (int i = 1; i <= n; i++)
        rotera (tal);
}

public static void rotera (int[] tal)
{
    int    sistaPos = tal.length - 1;
    int    t = tal[sistaPos];
    int    pos = sistaPos - 1;
    while (pos >= 0)
    {
        tal[pos + 1] = tal[pos];
        pos--;
    }
    tal[0] = t;
}

public static void rotera (int[] tal, int n)
{
    n = n % tal.length;

    int[]  t = new int[n];
```

```
    int    j = 0;
    int    pos = tal.length - n;
    for (int i = pos; i < tal.length; i++)
        t[j++] = tal[i];

    pos--;
    while (pos >= 0)
    {
        tal[pos + n] = tal[pos];
        pos--;
    }
    for (int i = 0; i < n; i++)
        tal[i] = t[i];
}
```

UPPGIFTER

a) Spåra de två givna algoritmerna både med papper och penna, och med datorns hjälp.

b) Formulera algoritmerna med ord.

c) Bestäm både minneskomplexiteten och tidskomplexiteten för algoritmerna. Kategorisera motsvarande komplexitetsfunktioner.

d) Jämför algoritmerna: vilken algoritm är minneseffektivare, vilken är tidseffektivare och vilken är enklare.

e) Bevisa algoritmerna.

Övning 11

PROBLEMET

En icke-tom tabell med element är given. Rotera tabellen radvis: flytta varje rad ett givet antal steg neråt. När den sista raden flyttas ett steg neråt, dyker den upp i början av tabellen – det är en cirkulär förflyttning (en rotation).

UPPGIFTER

a) Hitta en algoritm för tabellrotation, och beskriv den med ord.

b) Bestäm en instans av problemet, och spåra algoritmen med papper och penna i samband med den instansen.

c) Implementera algoritmen som en metod i Java, och testa den metoden.

d) Bestäm både minneskomplexiteten och tidskomplexiteten för algoritmen. Kategorisera motsvarande komplexitetsfunktioner.

e) Bevisa algoritmen.

f) Kan man lösa samma problem på ett tidseffektivare eller minneseffektivare sätt?

Övning 12

PROBLEMET

En icke-tom, kvadratisk tabell med element är given. Rotera tabellen kring huvuddiagonalen: de element som är ovanför diagonalen ska hamna på motsvarande platser under diagonalen, och tvärtom.

UPPGIFTER

a) Hitta en algoritm för tabellrotation och beskriv den med ord.

b) Bestäm en instans av problemet, och spåra algoritmen med papper och penna i samband med den instansen.

c) Implementera algoritmen som en metod i Java, och testa den metoden.

d) Bestäm både minneskomplexiteten och tidskomplexiteten för algoritmen. Kategorisera motsvarande komplexitetsfunktioner. Hur många gånger utförs ett elementutbyte om tabellen innehåller hundra rader?

e) Bevisa algoritmen.

f) Kan man lösa samma problem på ett mindre effektivt sätt när det gäller minne? Hur många extra minnesceller krävs i så fall?

Övning 13

PROBLEMET

En sekvens med jämförbara element är given. Sortera den sekvensen i icke-avtagande ordning.

ALGORITMEN BUBBELSORTERING

FÖRVILLKOR

En sekvens med jämförbara element är given.

EFTERVILLKOR

Sekvensen är sorterad i icke-avtagande ordning.

ALGORITMEN

```
sortera (sekvens med element)
{
    pos → första position
    sist → sista position
    while pos < sist
    {
        p = sist
        while p > pos
        {
            if element(p) < element(p - 1)
                utbyt plats på elementen

            p--
        }

        pos++
    }
}
```

UPPGIFTER

a) Använd algoritmen i samband med flera instanser av problemet.

b) Åskådliggör algoritmen.

c) Beskriv algoritmen med ord.

d) Implementera algoritmen som en metod i Java, och testa metoden.

e) Bestäm både minneskomplexiteten och tidskomplexiteten för algoritmen. Kategorisera motsvarande komplexitetsfunktioner. Hur många gånger utförs en elementjämförelse om sekvensen innehåller hundra element?

f) Hut många elementutbyten utförs i bästa fallet och hur många i värsta fallet?

g) Jämför bubbelsorteringen med urvalssorteringen.

h) Bevisa algoritmen.

Övning 14

PROBLEMET

En icke-tom tabell med jämförbara element och ordningsnummer av en kolumn i tabellen är givna. Sortera den givna kolumnen i icke-avtagande ordning.

UPPGIFTER

a) Hitta en algoritm för kolumnsortering och beskriv den med ord.

b) Beskriv algoritmen med pseudokod.

c) Bestäm en instans av problemet, och spåra algoritmen med papper och penna i samband med den instansen.

d) Implementera algoritmen som en metod i Java, och testa den metoden.

e) Bestäm både minneskomplexiteten och tidskomplexiteten för algoritmen. Kategorisera motsvarande komplexitetsfunktioner.

f) Bevisa algoritmen.

Övning 15

PROBLEMET

Ett schackbräde med jämförbara element på varje fält är given. Enligt den algebraiska schacknotationen är varje fält entydigt bestämt med en bukstav och ett nummer: a3, b7, c4, och så vidare.

Man anger ett fält på schackbrädet, och betraktar alla de fält som springaren kan nå från det givna fältet med ett enda steg. Man uppfattar element på dessa fält, sedda medsols, som en sekvens. Elementet på det fält som har minsta radnummer (minsta bokstav i beteckningen) utgör början av den sekvensen. Om det finns två sådana fält, väljs det som har mindre kolumnnummer (mindre nummer i beteckningen).

Elementen i sekvensen ska omplaceras: de ska sorteras i icke-avtagande ordning.

UPPGIFTER

a) Hitta en algoritm för elementsortering och beskriv den med ord.

b) Bestäm en instans av problemet, och spåra algoritmen med papper och penna i samband med den instansen.

c) Implementera algoritmen som en metod i Java, och testa den metoden.

d) Bestäm både minneskomplexiteten och tidskomplexiteten för algoritmen. Kategorisera motsvarande komplexitetsfunktioner.

e) Bevisa algoritmen.

Övning 16

En metod `min` bestämmer det minsta heltalet i en sekvens:

```
// min returnerar det minsta heltalet i en sekvens med heltal
public static int min (int[] element)
{
    if (element.length == 0)
        throw new IllegalArgumentException ("tom sekvens");

    int[]    sekvens = element;
    int      antaletPar = sekvens.length / 2;
    int[]    delsekvens = new int[antaletPar];
    int      i = 0;
    int      j = 0;
    while (antaletPar >= 1)
    {
        // urskilj en delsekvens med de tänkbara heltalen
        i = 0;
        j = 0;
        while (j < antaletPar)
        {
            delsekvens[j++] = (sekvens[i] < sekvens[i + 1])?
```

```
                                    sekvens[i] : sekvens[i + 1];
                i += 2;
            }

        // utgå nu ifrån delsekvensen
        sekvens = delsekvens;
        antaletPar = antaletPar / 2;
    }

    // sekvens[0] är det enda återstående tänkbart heltal -
    // det är det minsta heltalet
    return sekvens[0];
}
```

a) Bevisa att metoden inte är korrekt: ange en heltalssekvens då metoden ger ett felaktigt svar.

b) Vad är otillräckligt i den strategi som metoden använder?

c) För vissa sekvenslängder fungerar metoden bra: man kan variera både heltal och deras ordning – svaret blir alltid korrekt. Vilka är dessa sekvenslängder?

Problem 1

Den kortaste vägen

Ett problem: bestäm mellanstationer för den kortaste vägen

Det finns i ett trafiksystem fyra zoner: Z_1, Z_2, Z_3 och Z_4. I zonen Z_1 finns endast stationen X, och zonen Z_4 omfattar bara stationen Y. I zonen Z_2 finns stationerna U_1, U_2, …, U_m, (m är ett positivt heltal), och zonen Z_3 omfattar stationerna V_1, V_2, …, V_n (n är ett positivt heltal).

Det finns direkta vägar mellan stationen X och alla stationer i zonen Z_2. Zonerna Z_2 och Z_3 är väl kopplade med varandra: det finns en direkt väg mellan vilken station som helst i den ena zonen och en godtycklig station i den andra zonen. Det finns även en direkt väg mellan vilken station som helst i zonen Z_3 och stationen Y. Det finns inga andra vägar mellan givna stationer..

För ett godtyckligt heltal i, $1 \leq i \leq m$, gäller: längden av vägen mellan stationen X och stationen U_i är a_i.
För ett godtyckligt heltal i, $1 \leq i \leq m$, och för ett godtyckligt heltal j, $1 \leq j \leq n$, gäller: längden av vägen mellan stationen U_i och stationen V_j är b_{ij}.

För ett godtyckligt heltal j, $1 \leq j \leq n$, gäller: längden av vägen mellan stationen V_j och stationen Y är c_j

En väg mellan stationerna X och Y går genom en station i zonen Z_2 och en station i zonen Z_3. En mellanstation i var och en av zonerna Z_2 och Z_3 ska väljas, så att vägen mellan stationen X och stationen Y blir så kort som möjligt.

Det kan hända att det finns flera vägar som har den kortaste längden. I så fall ska mellanstationer på en av dessa vägar bestämmas.

Uppgifter i samband med problemet

1. Bestäm en instans av det här problemet i fallet att $m = 3$ och $n = 4$ – välj väglängderna. Specificera den instansen med en bild. Det ska framgå vilka stationer och vägar som finns, och hur långa vägarna är.

2. Specificera den valda instansen även med en tabell. Tabellen ska vara av följande form:

Z1	*ai*	*Z2*	*bij*	*Z3*	*cj*	*Z4*	*Längd*
X		*U1*		*V1*		*Y*	
X		*U1*		*V2*		*Y*	
X		*U1*		*V3*		*Y*	
X		*U1*		*V4*		*Y*	
X		*U2*		*V1*		*Y*	
X		*U2*		*V2*		*Y*	
X		*U2*		*V3*		*Y*	
X		*U2*		*V4*		*Y*	
X		*U3*		*V1*		*Y*	
X		*U3*		*V2*		*Y*	
X		*U3*		*V3*		*Y*	
X		*U3*		*V4*		*Y*	

Lös den aktuella instansen av problemet med papper och penna: undersök alla möjliga vägar och bestäm mellanstationerna för den kortaste vägen (för in längderna i kolumnen ”Längd” och välj de mellanstationer som motsvarar den minsta längden).

3. Hitta en minneseffektiv algoritm som löser det här problemet i ett allmänt fall – använd uppdateringsstrategi. Beskriv den algoritmen på två olika sätt: med ord och med pseudokod.

Beskrivningen ska vara på följande form:

PROBLEM

Problembeskrivning

ALGORITM

FÖRVILLKOR

Precisera algoritmens förvillkor

EFTERVILLKOR

Precisera algoritmens eftervillkor

STEG I ALGORITMEN

Beskriv steg i algoritmen med ord

STEG I ALGORITMEN – PSEUDOKOD

Beskriv steg i algoritmen med symboler

4. Skapa ett Javaprogram som kan lösa olika instanser av det här problemet. Använd programmet i samband med två instanser, och förklara de resultat som erhålls.

Det ska finnas två klasser: `DenKortasteVagen` och `BestamDenKortasteVagen`. Den första klassen ska se ut så här:

```
class DenKortasteVagen
{
    // mellanstationer returnerar en vektor med de mellanstationer
```

```
    // som finns på den kortaste vägen. Ordningsnummer av den första
    // mellanstationen finns på index 1, och ordningsnummer av den
    // andra mellanstationen på index 2 i vektorn.
    public static int[] mellanstationer (
                           double[] a, double[][] b, double[] c)
    {
        // koden här
    }


    // langd returnerar längden av den kortaste vägen.
    public static double langd (double[] a, double[][] b, double[] c)
    {
        // koden här
    }
}
```

Klassen `BestamDenKortasteVagen` ska innehålla metoden `main`, där instansspecifika uppgifter matas in, och metoder i klassen `DenKortasteVagen` anropas.

Problem 2

Det minsta heltalet – hitta fel i lösningen

Ett problem att lösa: bestäm det minsta heltalet

Det finns en sekventiell samling med heltal. Bestäm det minsta heltalet i samlingen.

Lösningen

Följande lösning innehåller två fel, utplacerade i två olika rader. Det första felet gör att algoritmen blir oändlig. Det andra felet leder till ett felaktigt svar för vissa talsekvenser.

Felen i lösningen ska hittas och korrigeras. När ett fel korrigeras, ska bara en del av den motsvarande raden ändras.

En lösning till problemet – innehåller två fel

```
// min returnerar det minsta elementet i en sekventiell samling.
// Om samlingen är tom, kastas ett undantag av typen
// IllegalArgumentException.
public static int min (int[] element) throws IllegalArgumentException
{
```

```
    if (element.length == 0)
        throw new IllegalArgumentException ("tom samling");

    // hör ihop med spårutskriften 2:
    // int    antalVarv = 1;

    int[]    sekvens = element;
    int      antaletPar = sekvens.length / 2;
    int      antaletOparadeElement = sekvens.length % 2;
    int      antaletTankbaraElement =
                       antaletPar + antaletOparadeElement;
    int[]    delsekvens = new int[antaletTankbaraElement];
    int      i = 0;
    int      j = 0;
    while (sekvens.length > 1)
    {
        // skilj ur en delsekvens med de tänkbara elementen
        i = 0;
        j = 0;
        while (j < antaletPar)
        {
            delsekvens[j++] = (sekvens[i] < sekvens[i + 1])?
                              sekvens[i] : sekvens[i + 1];
            i += 2;
        }
        if (antaletOparadeElement == 1)
            delsekvens[j] = sekvens[sekvens.length - 1];

        // utgå nu ifrån delsekvensen
        sekvens = delsekvens;
        antaletPar = antaletTankbaraElement / 2;
        antaletOparadeElement = antaletTankbaraElement % 2;
        antaletTankbaraElement = antaletPar + antaletOparadeElement;

        // spårutskrift 1 – för att följa sekvensen
        // System.out.println (java.util.Arrays.toString (sekvens));

        // spårutskrift 2 - för att avsluta loopen i förväg
        // (för att kunna se vad som händer i början)
        // if (antalVarv++ == 10)
        //    System.exit (0);
    }

    // sekvens[0] är det enda återstående tänkbara elementet
    // - det är det minsta elementet
    return sekvens[0];
}
```

Uppgifter i samband med problemet och lösningen

1. . Spåra metoden `min` i fallet att sekvensen innehåller sexton element. Använd papper och penna: rita en serie bilder som visar hur talsekvensen transformeras. Analysera transformationer av talsekvensen, och hitta det första felet i metoden. Rätta detta fel.

2. För att hitta det andra felet spåra den korrigerade varianten av metoden `min`. Använd en sekvens som innehåller nitton element med det minsta elementet på den sjuttonde positionen. Rita en serie bilder som visar hur det minsta heltalet bestäms. Varför erhålls ett felaktigt svar? Hitta det andra felet i metoden och rätta det.

3. Spåra den variant av metoden `min` som innehåller två fel med datorns hjälp. Skapa ett testprogram som anropar metoden `min`. Använd en sekvens med sexton element. Aktivera först spårutskriften `1` och följ utvecklingen. Aktivera sedan även spårutskriften `2`. Analysera de uppgifter som erhålls och hitta det första felet. Korrigera detta fel.

4. Spåra med datorns hjälp den variant av metoden `min` som enbart innehåller det andra felet. Använd en sekvens som innehåller nitton element med det minsta elementet på den sjuttonde positionen. Man ska aktivera bara spårutskriften `1`. Analysera de data som erhålls och hitta felet. Korrigera felet.

5. Lös problemet på ett annat sätt: skapa en ny metod `min` som bestämmer det minsta elementet i samlingen. I denna metod ska inte urskiljningsstrategin användas, utan en minneseffektivare och enklare strategi: uppdateringsstrategin.

Problem 3

Sortera en mängd

Ett problem att lösa: sortera element i en mängd

Det finns en mängd U, vars element kan jämföras med varandra med operatorn $<$. Det går att bestämma det mindre av två godtyckliga element i mängden. Mängden X är en ändlig, icke-tom delmängd i mängden U.

Sortera element i mängden X i stigande ordning.

En algoritm som löser problemet – utbytessortering

Algoritm: *sortera*

Förvillkor:

U är en mängd vars element kan jämföras med operatorn $<$,

N är mängden av alla naturliga heltal,

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

för ett godtyckligt heltal i, $1 \leq i \leq n$, betecknar x_i det element som finns på positionen i

Eftervillkor:

$x_1 < x_2 < ... < x_n$

Stegen i algoritmen:

```
sortera (n, X)
{
    i = 1
    while i < n
    {
        j = i + 1
        while j ≤ n
        {
            if (xj < xi)
                utbyt xj och xi
            j++
        }

        i++
    }
}
```

Uppgifter i samband med problemet och algoritmen

1. Åskådliggör algoritmen: rita en serie bilder som visar hur en mängd sorteras.

2. Bestäm algoritmens tidskomplexitet när det gäller antalet elementjämförelser: bestäm motsvarande komplexitetsfunktion. Till vilken θ–mängd tillhör denna komplexitetsfunktion?

3. Bestäm algoritmens tidskomplexitet när det gäller antalet elementutbyten. Bestäm komplexiteten i bästa fall, i värsta fall och i ett genomsnittligt fall. Anta att sannolikheten för ett utbyte i ett genomsnittligt fall är 0.5.

Kategorisera de motsvarande komplexitetsfunktionerna: till vilken θ–mängd tillhör dem?

4. Jämför tidskomplexiteten mellan utbytessorteringen och urvalssorteringen. Både antalet jämförelser och antalet utbyten ska betraktas.

5. Bevisa algoritmen.

Beviset ska ha följande struktur:

A) INRE LOOPEN

ETT PÅSTÅENDE OM INRE LOOPEN

När den inre loopen har utförts, gäller följande:

$x_i = minimum\ \{x_i, x_{i+1}, ..., x_n\}$

BEVIS

Här ska beviset finnas. Man ska fastställa ett påstående om variablerna, och bevisa att detta påstående är en loopinvariant för den inre loopen. Med hjälp av denna loopinvariant ska beviset sedan härledas.

B) HUVUDLOOPEN

ETT PÅSTÅENDE OM HUVUDLOOPEN

När huvudloopen har utförts, gäller följande:

$x_1 < x_2 < ... < x_n$

BEVIS

Här ska beviset, som utnyttjar påståendet om den inre loopen, finnas. Man ska fastställa ett påstående om variablerna, och bevisa att detta påstående är en loopinvariant för huvudloopen. Med hjälp av denna loopinvariant ska beviset sedan härledas.

Kapitel 10

Objekt

Övning 1

```
import java.awt.Point;
import static java.lang.System.out;

class Punkter
{
    public static void main (String[] args)
    {
        Point    p1 = null;
        Point    p2 = null;

        p1 = new Point ();
        p2 = new Point (3, 4);
        out.println (p1.toString ());
        out.println (p2.toString ());
        double    d = p1.distance (p2);
        out.println (d);
        out.println ();

        p2 = new Point (5, 12);
        out.println (p1);
        out.println (p2);
        d = p1.distance (p2);
        out.println (d);
        out.println ();

        p1 = null;
        p2.move (1, 1);
        d = p1.distance (p2);
        out.println (d);
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vilka klasser, konstruktorer och instansmetoder används i programmet?

c) Rita de objekt och referenser som skapas i programmet. Vart refererar referenserna?

d) Hur många gånger anropas metoden `toString`?

Övning 2

```
import java.awt.Point;
import static java.lang.System.out;

class Punkter
{
    public static void main (String[] args)
    {
        Point    p1 = new Point (3, 4);
        Point    p2 = new Point (5, 6);
        Point    p3 = p1;
        Point    p4 = new Point (p1);

        String    s1 = toString (p1);
        out.println (s1);
        out.println (toString (p2));
        out.println (toString (p3));
        out.println (toString (p4));
        out.println ();

        double    d = p1.distance (p2);
        out.println (d);
        out.println ();

        out.println (p3 == p1);
        out.println (p4 == p1);
        out.println (p4.equals (p1));
        out.println ();

        Class    c = p1.getClass ();
        out.println (c.getName ());
    }

    public static String toString (Point p)
    {
        String    s = p.toString ();
        s = s.substring (s.indexOf ("["));

        return s;
    }
}
```

a) Vad händer när det här programmet exekveras?

b) När är två objekt av typen `java.awt.Point` likadana?

c) Varifrån kommer metoden `getClass`?

d) Vilken roll har metoden `toString`? Vart pekar parameterreferensen `p`? Till vilket objekt refererar referensen `s1`? Ritta detta objekt.

Övning 3

```
import java.util.Arrays;
import static java.lang.System.out;

class PaverkaArgument
{
    public static void main (String[] args)
    {
        StringBuilder    s = new StringBuilder ("abcde");
        out.println (s);

        StringBuilder    s1 = paverka (s);
        out.println (s);
        out.println (s1);
        out.println ();

        StringBuilder[]   v = new StringBuilder[4];
        v[0] = new StringBuilder ("1");
        v[1] = new StringBuilder ("2");
        v[2] = new StringBuilder ("3");
        v[3] = new StringBuilder ("4");
        out.println (Arrays.toString (v));

        paverka (v);
        out.println (Arrays.toString (v));
    }

    public static StringBuilder paverka (StringBuilder sb)
    {
        sb.append ("01234");

        sb = new StringBuilder ();
        sb.append ("uvwxy");

        return sb;
    }

    public static void paverka (StringBuilder[] sb)
    {
        sb[sb.length - 1].insert (0, "D");
```

```
    }
}
```

a) Vilken utskrift skapas när det här programmet exekveras?

b) Vart pekar parameterreferenserna `sb` i de två metoderna som heter `paverka`? Rita motsvarande bilder.

c) Vart pekar referensen `s1`? Rita bilden.

Övning 4

```
// splittra tar emot en teckensträng, och returnerar en
// vektor som innehåller strängens ord som sina element
public static String[] splittra (String s)
{
    // bestäm antalet ord i strängen
    java.util.Scanner    t = new java.util.Scanner (s);
    int    antalOrd = 0;
    String    o;
    while (t.hasNext ())
    {
        o = t.next ();
        antalOrd++;
    }

    // skapa en vektor och lagra strängens ord i den
    // koden här
}
```

a) Komplettera metoden `splittra`: skriv den kod som saknas.

b) Anropa metoden `splittra` med strängen `ett tva tre fyra` som argument. Rita den vektor som returneras i så fall.

Övning 5

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 `"000457635646670043577777"`), 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 teckenströngen till motsvarande teckenvektor.

b) Anropa metoden `taBortInledandeNollor` på något sätt.

Övning 6

```
public static String stringMedSkiljetecken (String s,
                                            char skiljeTecken)
{
    StringBuilder    sMedSkiljetecken = new StringBuilder ("");
    if (!s.equals (""))
    {
        int    langd = s.length ();
        sMedSkiljetecken.append (s.charAt(0));
        for (int pos = 1; pos < langd; pos++)
        {
            sMedSkiljetecken.append (skiljeTecken);
            sMedSkiljetecken.append (s.charAt(pos));
        }
    }

    return sMedSkiljetecken.toString ();
}
```

a) Spåra metoden `stringMedSkiljetecken` i samband med strängen `abcde`, i fall att skiljetecknet `|` används.

b) Formulera den algoritm som används i metoden `stringMedSkiljetecken` med ord.

Övning 7

En metod `calculate` utför en beräkning med heltal:

```
public static int calculate (int m, int n)
{
    int    p = m / n;
    int    q = m % n;
    int    r = p + q;
    int    res = (int) Math.pow (r, 2);

    return res;
}
```

Metoden `calculate` kan anropas, till exempel, så här:

```
int    m = 6405;
int    n = 3200;
int    res1 = calculate (m, n);  // res1 blir 49
```

Men denna metod kan inte utföra beräkningen om heltalen är alltför långa. I så fall kan man representera heltalen med motsvarande teckensträngar, och få resultatet i form av en teckensträng. Motsvarande metod skulle kunna anropas, till exempel, så här:

```
String    n1 = "6400000000000000000005";
String    n2 = "3200000000000000000000";
String    res2 = calculate (n1, n2);  // res2 blir "49"
```

Skapa en sådan metod `calculate`, som kan utföra motsvarande beräkning med långa heltal. Inuti metoden ska objekt av typen `java.math.BigInteger` skapas utifrån givna teckensträngar. Beräkningen ska utföras med motsvarande metoder i klassen `BigInteger`.

Övning 8

```
import java.util.Scanner;
import java.awt.Point;
import static java.lang.System.out;

class Polygon
{
    public static void main (String[] args)
    {
        Scanner    in = new Scanner (System.in);
        out.print ("antalet hörn i polygonen: ");
        int    n = in.nextInt ();
        Point[]    h = new Point[n];

        int    x = 0;
        int    y = 0;
        for (int i = 0; i < n; i++)
        {
            out.print ("hörn " + (i + 1) + ": ");
            x = in.nextInt ();
            y = in.nextInt ();
            h[i] = new Point (x, y);
        }
        out.println ();

        out.println (toString (h));
        out.println (omkrets (h));
    }

    public static double omkrets (Point[] h)
    {
        double    omkrets = 0;
        for (int i = 0; i < h.length; i++)
            omkrets += h[i].distance (h[(i + 1) % h.length]);
```

```
        return omkrets;
    }

    public static String toString (Point[] h)
    {
        StringBuilder    s = new StringBuilder ("[");
        for (int i = 0; i < h.length; i++)
            s.append ("(" + h[i].x + ", " + h[i].y + ")");
        s.append ("]");

        return s.toString ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Rita den vektorn som innehåller polygonens hörn. Vilka referenser refererar till denna vektor när metoden `omkrets` exekveras, och vilka när metoden `toString` exekveras?

c) Följ noggrant stegen i metoderna `omkrets` och `toString`. Beskriv dessa steg med ord.

Övning 9

```
import java.util.Scanner;
import java.awt.Point;
import static java.lang.System.out;

class PunkterIEnOmgivning
{
    public static void main (String[] args)
    {
        Scanner    in = new Scanner (System.in);
        out.print ("antalet punkter: ");
        int    n = in.nextInt ();
        Point[]    p = new Point[n];

        int    x = 0;
        int    y = 0;
        for (int i = 0; i < n; i++)
        {
            out.print ("punkt " + (i + 1) + ": ");
            x = in.nextInt ();
            y = in.nextInt ();
            p[i] = new Point (x, y);
        }
        out.print ("omgivningens radie: ");
        int    r = in.nextInt ();
        out.println ();
```

```
        Point[]   pIOmg = iOrigosOmgivning (p, r);
        out.println (toString (p));
        out.println (toString (pIOmg));
    }

    public static Point[] iOrigosOmgivning (Point[] p, int r)
    {
        boolean[]    iOmg = new boolean[p.length];
        int    antalIOmg = 0;

        Point    origo = new Point (0, 0);
        for (int i = 0; i < p.length; i++)
            if (p[i].distance (origo) <= r)
            {
                iOmg[i] = true;
                antalIOmg++;
            }

        Point[]    pt = new Point[antalIOmg];
        int    pos = 0;
        for (int i = 0; i < iOmg.length; i++)
            if (iOmg[i])
                pt[pos++] = new Point (p[i]);

        return pt;
    }

    public static String toString (Point[] p)
    {
        StringBuilder    s = new StringBuilder ("[");
        for (int i = 0; i < p.length; i++)
            s.append ("(" + p[i].x + ", " + p[i].y + ")");
        s.append ("]");

        return s.toString ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vart pekar referensen `pIOmg`? Rita bilden.

c) Följ noggrant stegen i metoden `iOrigosOmgivning`. Beskriv dessa steg med ord.

Övning 10

```
import java.util.Scanner;
import java.awt.Point;
import static java.lang.System.out;
```

```
class FordelaPunkter
{
    public static void main (String[] args)
    {
        Scanner    in = new Scanner (System.in);
        out.print ("antalet punkter: ");
        int    n = in.nextInt ();
        Point[]    p = new Point[n];

        int    x = 0;
        int    y = 0;
        for (int i = 0; i < n; i++)
        {
            out.print ("punkt " + (i + 1) + ": ");
            x = in.nextInt ();
            y = in.nextInt ();
            p[i] = new Point (x, y);
        }
        out.println ();

        Point[][]   pnt = fordelaPunkter (p);
        out.println (toString (p));
        out.println (toString (pnt));
    }

    public static Point[][] fordelaPunkter (Point[] p)
    {
        int[]    kvadrant = new int[p.length];
        int[]    antalIKv = new int[5];

        for (int i = 0; i < p.length; i++)
        {
            if (p[i].x == 0  ||  p[i].y == 0)
            {
                kvadrant[i] = 0;
                antalIKv[0]++;
            }
            else if (p[i].x > 0)
            {
                if (p[i].y > 0)
                {
                    kvadrant[i] = 1;
                    antalIKv[1]++;
                }
                else
                {
                    kvadrant[i] = 4;
                    antalIKv[4]++;
                }
            }
```

```
            else
            {
                if (p[i].y > 0)
                {
                    kvadrant[i] = 2;
                    antalIKv[2]++;
                }
                else
                {
                    kvadrant[i] = 3;
                    antalIKv[3]++;
                }
            }
        }

        Point[][]    pt = new Point[5][];
        for (int i = 0; i < pt.length; i++)
            pt[i] = new Point[antalIKv[i]];
        int[]    pos = new int[pt.length];
        for (int i = 0; i < p.length; i++)
        {
            pt[kvadrant[i]][pos[kvadrant[i]]] = new Point (p[i]);
            pos[kvadrant[i]]++;
        }

        return pt;
    }

    public static String toString (Point[] p)
    {
        StringBuilder    s = new StringBuilder ("[");
        for (int i = 0; i < p.length; i++)
            s.append ("(" + p[i].x + ", " + p[i].y + ")");
        s.append ("]");

        return s.toString ();
    }

    public static String toString (Point[][] p)
    {
        StringBuilder    s = new StringBuilder ("{");
        for (int i = 0; i < p.length - 1; i++)
            s.append (toString (p[i]) + ",\n");
        if (p.length - 1 >= 0)
            s.append (toString (p[p.length - 1]));
        s.append ("}");

        return s.toString ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vart pekar referensen `pnt`? Rita bilden.

c) På vilka ställen anropas metoden `toString` (den första av de två metoderna med samma namn) och vart pekar dess parameterreferens `p` under exekveringen?

d) Följ noggrant stegen i metoden `fordelaPunkter`. Beskriv dessa steg med ord.

Övning 11

```
import java.util.Scanner;
import static java.lang.System.out;

class Personer
{
    public static void main (String[] args)
    {
        Scanner    in = new Scanner (System.in);
        out.print ("antalet personer: ");
        int    n = in.nextInt ();
        in.nextLine ();
        String[]    p = new String[n];
        out.println ("personerna - efternamn och förnamn: ");
        for (int i = 0; i < n; i++)
            p[i] = in.nextLine ();
        out.println ();

        sortera (p);
        out.println (toString (p));
        out.println ();

        invertera (p);
        sortera (p);
        out.println (toString (p));
    }

    public static void sortera (String[] s)
    {
        int    pos = 0;
        int    p = 0;
        String   str = null;
        while (pos < s.length - 1)
        {
            p = pos + 1;
            while (p < s.length)
            {
                if (s[p].compareToIgnoreCase (s[pos]) < 0)
                {
```

```
                    str = s[pos];
                    s[pos] = s[p];
                    s[p] = str;
                }

                p++;
            }

            pos++;
        }
    }

    public static void invertera (String[] s)
    {
        Scanner    t = null;
        String     ord1 = null;
        String     ord2 = null;
        for (int i = 0; i < s.length; i++)
        {
            t = new Scanner (s[i]);
            ord1 = t.next ();
            ord2 = t.next ();
            s[i] = ord2 + " " + ord1;
        }
    }

    public static String toString (String[] s)
    {
        StringBuilder    sb = new StringBuilder ("");
        for (int i = 0; i < 20; i++)
            sb.append ("-");
        sb.append ("\n");
        for (int i = 0; i < s.length - 1; i++)
            sb.append (s[i] + ",\n");
        if (s.length - 1 >= 0)
            sb.append (s[s.length - 1]);
        sb.append ("\n");
        for (int i = 0; i < 20; i++)
            sb.append ("-");

        return sb.toString ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Åskådliggör den algoritm som används i metoden `sortera`.

c) Åskådliggör den algoritm som används i metoden `invertera`.

Övning 12

```
import java.util.*;    // Scanner, Arrays
import static java.lang.System.out;

class SorteraForstaOrd
{
    public static void main (String[] args)
    {
        Scanner    in = new Scanner (System.in);
        out.print ("antalet ord: ");
        int    n = in.nextInt ();
        String[]    ord = new String[n];
        out.println ("orden: ");
        for (int i = 0; i < n; i++)
            ord[i] = in.next ();
        out.println ();
        out.println (Arrays.toString (ord));

        int    ordPos = sorteraForstaOrd (ord);
        out.println ("plats för det första ordet: " + ordPos);
        out.println (Arrays.toString (ord));
    }

    public static int sorteraForstaOrd (String[] ord)
    {
        int    framPos = 1;
        int    bakPos = ord.length - 1;
        int    ordPos = 0;
        boolean    ordPosBestamd = false;
        String    o = null;
        while (!ordPosBestamd)
        {
            while (framPos <= bakPos
               &&  ord[framPos].compareTo (ord[0]) <= 0)
                framPos++;

            while (framPos <= bakPos
               &&  ord[bakPos].compareTo (ord[0]) > 0)
                bakPos--;

            if (framPos <= bakPos)
            {
                o = ord[framPos];
                ord[framPos] = ord[bakPos];
                ord[bakPos] = o;
            }
            else
            {
                ordPos = bakPos;
```

```
                ordPosBestamd = true;
            }
        }

        o = ord[0];
        ord[0] = ord[ordPos];
        ord[ordPos] = o;

        return ordPos;
    }
}
```

a) När det här programmet exekveras, sorteras det första av de inmatade orden – ordet hamnar på den position där det skulle finnas i fall att alla ord sorterats. Vad händer med de andra orden?

b) Vad händer om metoden `compareToIgnoreCase` används för ordjämförelser?

c) Åskådliggör den algoritm som används i metoden `sorteraForstaOrd`.

d) På vilket sätt utbyter två ord plats? Är det orden eller motsvarande referenser som utbyter plats?

e) Beskriv den algoritm som används i metoden `sorteraForstaOrd` - både med ord och med pseudokod.

f) Bestäm komplexiteten för den algoritm som används i metoden `sorteraForstaOrd`.

g) Bevisa algoritmen.

h) Hur kan man anpassa algoritmen om ordet på en given position ska sorteras?

Övning 13

FÖRVILLKOR

En sekvens med ord och ett ord är givna.

EFTERVILLKOR

En pekare, som pekar till ett av de ord i sekvensen som är likadana som givna ordet, returneras. Om sådana ord inte finns i sekvensen, pekar den returnerade pekaren ingenstans.

ALGORITMEN

```
sok (sekvens med ord, ord)
{
```

```
    forst → första position
    sist → sista position
    aktuell = forst

    while aktuell <= sist  &&  ord(aktuell) != ord
        aktuell++

    return aktuell
}
```

a) Vad gör den här algoritmen? Åskådliggör den.

b) Beskriv algoritmen med ord.

c) Implementera algoritmen som en metod i Java. Använd sedan den metoden.

d) Bestäm algoritmens komplexitet i bästa fallet, i värsta fallet och i ett genomsnittligt fall.

e) Bevisa algoritmen.

Övning 14

FÖRVILLKOR

En sekvens med ord är given.

EFTERVILLKOR

Orden är sorterade i lexikografisk ordning.

ALGORITMEN.

```
sortera (sekvens med ord)
{
    forst → första position
    sist → sista position
    aktuell = forst + 1
    hal → ingenstans
    o  // en plats för ett ord

    while aktuell <= sist
    {
        ord(aktuell) --> platsen o  // flytta ordet till platsen o
        hal = aktuell
        while hal > forst  &&  ord(o) < ord(hal – 1)
        {
            ord(hal - 1) --> plats(hal)
```

```
            hal--
        }

        ord(o) --> plats (hal)

        aktuell++
    }
}
```

a) Vad gör den här algoritmen? Åskådliggör den.

b) Beskriv algoritmen med ord.

c) Implementera algoritmen som en metod i Java. Använd sedan den metoden.

d) Bestäm algoritmens komplexitet i bästa fallet, i värsta fallet och i ett genomsnittligt fall.

e) Bevisa algoritmen.

Problem 1

Räkna med teckensträngar

Ett problem: utför aritmetiska operationer i samband med naturliga heltal givna som teckensträngar

Två naturliga heltal är givna som teckensträngar av godtycklig längd.

Man ska utföra olika aritmetiska operationer i samband med dessa heltal. I en operation utgår man ifrån de siffror som finns i givna teckensträngar, och bestämmer siffrorna i resultatet, en siffra i taget. På så sätt skapas en ny teckensträng, som representerar resultatet av operationen.

En lösning till problemet – ej fullständig

```
import java.util.*;    // Scanner
import static java.lang.System.out;

class OperationerMedNaturligaHeltalGivnaSomTeckenstrangar
{
    public static void main (String[] args)
    {
        out.println ("OPERATIONER MED NATURLIGA HELTAL "
                   + "GIVNA SOM TECKENSTRANGAR\n");
```

```
    // mata in två naturliga heltal
    Scanner    in = new Scanner (System.in);
    out.println ("två naturliga heltal:");
    String    tal1 = in.next ();
    String    tal2 = in.next ();
    out.println ();

    // addera heltalen och visa resultatet
    String    summa = addera (tal1, tal2);
    visa (tal1, tal2, summa, '+');

    // subtrahera heltalen och visa resultatet
    // koden här
}

// addera tar emot två naturliga heltal givna som teckensträngar,
// och returnerar deras summa som en teckensträng.
public static String addera (String tal1, String tal2)
{
    // koden ska skrivas här
}

// subtrahera tar emot två naturliga heltal givna som
// teckensträngar, och returnerar deras differens som
// en teckensträng.
// Det första heltalet är inte mindre än det andra heltalet.
public static String subtrahera (String tal1, String tal2)
{
    // koden ska skrivas här
}

// visa visar två givna naturliga heltal, och resultatet av
// en aritmetisk operation utförd i samband med hetalen
public static void visa (
      String tal1, String tal2, String resultat, char operator)
{
    // sätt en lämplig längd på heltalen och resultatet
    int    len1 = tal1.length ();
    int    len2 = tal2.length ();
    int    len  = resultat.length ();
    int    maxLen = Math.max (Math.max (len1, len2), len);
    tal1 = sattLen (tal1, maxLen - len1);
    tal2 = sattLen (tal2, maxLen - len2);
    resultat = sattLen (resultat, maxLen - len);

    // visa heltalen och resultatet
    out.println ("  " + tal1);
    out.println ("" + operator + " " + tal2);
    for (int i = 0; i < maxLen + 2; i++)
        out.print ("-");
    out.println ();
```

```
        out.println ("  " + resultat + "\n");
    }

    // sattLen lägger till ett angivet antal mellanslag i början av
    // en given sträng
    public static String sattLen (String s, int antal)
    {
        StringBuilder    sb = new StringBuilder (s);
        for (int i = 0; i < antal; i++)
            sb.insert (0, " ");

        return sb.toString ();
    }
}
```

Uppgifter i samband med problemet och lösningen

1. Skapa en algoritm som adderar två naturliga heltal, givna som teckensträngar. Åskådliggör den algoritmen: skapa en serie bilder som visar hur operationen fortgår. Beskriv algoritmen med motsvarande pseudokod. Implementera algoritmen i form av en Javametod.

2. Skapa en algoritm som subtraherar två naturliga heltal, givna som teckensträngar. Åskådliggör den algoritmen: skapa en serie bilder som visar hur operationen fortgår. Beskriv algoritmen med motsvarande pseudokod. Implementera algoritmen i form av en Javametod.

3. Komplettera programmet `OperationerMedNaturligaHeltalGivnaSomTeckenstrangar` så att det blir en meningsfull enhet.

4. Om så önskas, arbeta även med multiplikation och division av naturliga heltal, givna som teckensträngar.

Problem 2

Ett aritmetiskt uttryck som en teckensträng

Ett problem: beräkna ett aritmetiskt uttryck, givet som en teckensträng

Ett aritmetiskt uttryck är givet i form av en teckensträng. Uttrycket består av ett antal heltalsoperander, och operatorer `+` och `*`. Operanderna och operatorerna utgör ord i teckensträngen: de är separerade med mellanslag. Ett exempel på ett sådant uttryck är: `5 * 2 + 3 * 4 * 5 + 10`.

Ett program ska skapas som beräknar aritmetiska uttryck givna som teckensträngar.

En lösning till problemet – innehåller ett fel

```
import java.util.*;  // Scanner

public class BeraknaUttryck
{
    public static void main (String[] args)
    {
        // inmatningsverktyg
        Scanner    in = new Scanner (System.in);

        // mata in och beräkna uttryck
        String    uttryck = "";
        int    resultat = 0;
        System.out.println ("Ett uttryck, avsluta med q " +
                            "eller Q:");
        uttryck = in.nextLine ();
        while (!uttryck.equals ("q")  &&  !uttryck.equals ("Q"))
        {
            resultat = berakna (uttryck);
            System.out.println (resultat);

            System.out.println ("Ett uttryck, avsluta med q " +
                                "eller Q:");
            uttryck = in.nextLine ();
        }
    }

    // berakna beräknar ett aritmetiskt uttryck givet som en
    // teckensträng, och returnerar resultatet.
    // Uttrycket består av ett antal heltalsoperander och
    // operatorer + och *, separerade med mellanslag.
    public static int berakna (String uttryck)
    {
        String[]    ord = uttryck.split ("\\s");
        int    operand = 0;
        String    operator = null;
        int    res = 0;
        int    p = 1;
        int    pos = 0;
        while (pos < ord.length)
        {
            p = Integer.parseInt (ord[pos]);
            pos++;
            if (pos < ord.length)
            {
                operator = ord[pos];
```

```
                while (operator != null  &&  operator.equals ("*"))
                {
                    pos++;
                    operand = Integer.parseInt(ord[pos]);
                    p = p * operand;

                    pos++;
                    if (pos < ord.length)
                        operator = ord[pos];
                    else
                        operator = null;
                }
            }

            res = res + p;
            pos++;
        }

        return res;
    }
}
```

Uppgifter i samband med problemet och lösningen

1. Testa programmet och rätta fel.

2. Programmet beräknar exempelvis `5 + 4` och `5 - 4` på samma sätt. Inför undantag i fallet att en otillåten operator anges i uttrycket.

3. Lös problemet på ett annat sätt – använd en minneseffektivare strategi. I stället för att bestämma och lagra alla ord redan i början, plocka ut och lagra bara de ord som behövs vid ett visst tillfälle.

Problem 3

Sortering av objekt

A) Ett problem att lösa: skapa en sorterad sekvens utifrån två sorterade sekvenser (sammanfoga två sorterade sekvenser)

Det finns två sorterade sekvenser med heltal.

Skapa en ny sorterad sekvens, som ska omfatta alla heltal – både dem från den första sekvensen och dem från den andra sekvensen.

En lösning till problemet – sammanfogningssortering

```
public static int[] sammanfoga (int[] v1, int[] v2)
{
    int[]    v = new int[v1.length + v2.length];
    int    i = 0;
    int    j = 0;
    int    k = 0;
    while (i < v1.length  &&  j < v2.length)
        if (v1[i] < v2[j])
            v[k++] = v1[i++];
        else
            v[k++] = v2[j++];

    while (i < v1.length)
        v[k++] = v1[i++];

    while (j < v2.length)
        v[k++] = v2[j++];

    return v;
}
```

Uppgifter i samband med problemet och lösningen

1. Åskådliggör sammanfogningsalgoritmen. Beskriv algoritmen med pseudokod. Skapa kort med heltal och ett slags pekare, och simulera algoritmen.

2. Lös problemet på ett annat sätt: lägg den andra sekvensen efter den första, och använd därefter urvalssortering. Simulera även denna algoritm.

3. Vilken av de två algoritmerna är effektivare när det gäller tiden?

B) Två problem att lösa: sortera en sekvens med objekt + sammanfoga två sorterade sekvenser med objekt

En sekvens med objekt av typen `java.math.BigInteger` ska sorteras.

Två sorterade sekvenser med objekt av typen `java.math.BigInteger` ska sammanfogas i en ny sorterad sekvens.

En lösning till problemet – ej fullständig

```
import java.math.*;    // BigInteger

class ObjektSortering
{
    public static final int    ANTAL1 = 10;
    public static final int    ANTAL2 = 10;

    public static void main (String[] args)
    {
        // en sekvens med heltal
        // - ett heltal i sekvensen innehåller mellan 15
        // och 25 siffror.
        BigInteger[]    tal1 = new BigInteger[ANTAL1];
        int    langd = 0;
        for (int i = 0; i < tal1.length; i++)
        {
            langd = (int) (11 * Math.random ()) + 15;
            tal1[i] = slumpTal (langd);
        }

        // en sekvens med heltal
        // - ett heltal i sekvensen innehåller mellan 20
        // och 30 siffror.
        BigInteger[]    tal2 = new BigInteger[ANTAL2];
        for (int i = 0; i < tal2.length; i++)
        {
            langd = (int) (11 * Math.random ()) + 20;
            tal2[i] = slumpTal (langd);
        }

        // sortera och visa talen
        sortera (tal1);
        sortera (tal2);
        visa (tal1);
        System.out.println ();
        visa (tal2);
        System.out.println ();

        // sammanfoga de sorterade sekvenserna i en ny sorterad
        // sekvens, och visa den nya sekvensen
        BigInteger[]    tal = sammanfoga (tal1, tal2);
        visa (tal);
    }


    // slumpTal skapar och returnerar ett slumpmässigt heltal
    // av en given längd.
    private static BigInteger slumpTal (int langd)
    {
```

```
        // koden ska läggas här
    }


    // visa visar en sekvens med heltal, ett heltal per rad
    public static void visa (BigInteger[] tal)
    {
        for (int i = 0; i < tal.length; i++)
            System.out.println (tal[i]);
    }


    // sortera sorterar en given sekvens med heltal i
    // icke-avtagande ordning
    public static void sortera (BigInteger[] tal)
    {
        // koden ska läggas här
    }


    // sammanfoga tar emot två sorterade sekvenser med heltal, och
    // skapar och returnerar en ny sorterad sekvens. Den nya
    // sekvensen omfattar alla heltal - både dem från den första
    // sekvensen och dem från den andra sekvensen.
    public static BigInteger[] sammanfoga (BigInteger[] tal1,
                                           BigInteger[] tal2)
    {
        // koden ska läggas här
    }
}
```

Uppgifter i samband med problemet och lösningen

1. Komplettera programmet `ObjektSortering` så att det blir en meningsfull enhet.

2. Kan man på samma sätt sortera och sammanfoga objekt av klassen `java.math.BigDecimal`? Gäller det även klasserna `java.lang.String` och `java.lang.StringBuilder`?

Kapitel 11
Undantag

Övning 1

```
import java.util.*;    // Scanner, NoSuchElementException
import static java.lang.System.out;

class KastaUndantag
{
    public static void main (String[] args)
    {
        Scanner    in = new Scanner (System.in);

        out.print ("gränser för intervallet: ");
        int    fran = in.nextInt ();
        int    till = in.nextInt ();

        int    minPrimtal = minstaPrimtal (fran, till);
        out.println ("minsta primtalet i intervallet: " + minPrimtal);
    }

    // minstaPrimtal returnerar det minsta primtalet i ett
    // givet heltalsintervall. Intervallet anges med dess
    // nedre och dess övre gräns: de båda gränserna är positiva
    // heltal och ingår i intervallet.
    // Om ett felaktigt intervall anges, kastas ett undantag av
    // typen java.lang.IllegalArgumentException.
    // Om givna intervallet saknar primtal, kastas ett undantag
    // av typen java.util.NoSuchElementException.
    public static int minstaPrimtal (int franTal, int tillTal)
        throws IllegalArgumentException, NoSuchElementException
    {
        if (!(franTal > 0  &&  franTal <= tillTal))
            throw new IllegalArgumentException (
                                "felaktigt heltalsintervall");

        int    minPrim = 0;
        int    tal = franTal;
        while (tal <= tillTal)
        {
```

```
            if (arPrimtal (tal))
            {
                minPrim = tal;
                break;
            }

            tal++;
        }

        if (minPrim == 0)
            throw new NoSuchElementException (
                              "intervallet sakanar primtal");

        return minPrim;
    }

    // arPrimtal tar emot ett heltal, och returnerar true
    // om heltalet är ett primtal, annars returnerar den false.
    public static boolean arPrimtal (int n)
    {
        if (n < 2)
            return false;

        boolean    arPrim = true;
        int    r = (int) Math.floor (Math.sqrt (n));
        for (int k = 2; k <= r; k++)
        {
            if (n % k == 0)
            {
                arPrim = false;
                break;
            }
        }

        return arPrim;
    }
}
```

a) Spåra metoden `arPrimtal` för olika värden på parametern.

b) Spåra metoden `minstaPrimtal` för olika heltalsintervall.

c) Vad händer när programmet `KastaUndantag` exekveras? I vilka situationer kastas undantag under exekveringen? Vad händer i så fall?

Övning 2

```
// Vektorer.java
```

```
class Vektorer
{
    // sortera tar emot en vektor med teckensträngar och två
    // positioner. Metoden sorterar den del av vektorn som finns
    // mellan de givna positionerna, inklusive dessa positioner.
    // Om den första positionen är större än den andra positionen,
    // kastas ett undantag av typen IllegalArgumentException.
    // I fall att den första positionen är negativ, och i fall att
    // den andra positionen är större än den sista positionen,
    // kastas ett undantag av typen ArrayIndexOutOfBoundsException.
    public static void sortera (String[] s, int pos1, int pos2)
        throws IllegalArgumentException,
               ArrayIndexOutOfBoundsException
    {
        if (pos1 > pos2)
            throw new IllegalArgumentException (pos1 + " > " + pos2);
        if (pos1 < 0  ||  pos2 >= s.length)
            throw new ArrayIndexOutOfBoundsException (
                "felaktigt intervall: [" + pos1 + ", " + pos2 + "]");

        int    pos = pos1;
        int    p = 0;
        int    minPos = 0;
        String    str = null;
        while (pos < pos2)
        {
            minPos = pos;
            p = pos + 1;
            while (p <= pos2)
            {
                if (s[p].compareToIgnoreCase (s[minPos]) < 0)
                    minPos = p;

                p++;
            }

            str = s[pos];
            s[pos] = s[minPos];
            s[minPos] = str;

            pos++;
        }
    }
}


// StrategierMedUndantag.java

import java.util.*;    // Arrays, Scanner
import static java.lang.System.out;
```

```
class StrategierMedUndantag
{
    public static Scanner    in = new Scanner (System.in);

    public static void main (String[] args)
    {
        int    m = (int) (3 * Math.random ());
        switch (m)
        {
        case 0:
            metod0 ();
            break;
        case 1:
            metod1 ();
            break;
        case 2:
            metod2 ();
            break;
        }
    }

    public static void metod0 ()
    {
        out.println ("flera ord i en rad: ");
        String    rad = in.nextLine ();
        String[]    ord = rad.split ("\\s");
        out.println (Arrays.toString (ord));
        out.println ();

        out.print ("positioner för två ord: ");
        int    p1 = in.nextInt ();
        int    p2 = in.nextInt ();

        Vektorer.sortera (ord, p1, p2);
        out.println (Arrays.toString (ord));
    }

    public static void metod1 ()
    {
        out.println ("flera ord i en rad: ");
        String    rad = in.nextLine ();
        String[]    ord = rad.split ("\\s");
        out.println (Arrays.toString (ord));
        out.println ();

        out.print ("positioner för två ord: ");
        int    p1 = in.nextInt ();
        int    p2 = in.nextInt ();

        try
        {
```

```
            Vektorer.sortera (ord, p1, p2);
            out.println (Arrays.toString (ord));
        }
        catch (IllegalArgumentException e)
        {
            out.println ("felaktig ordning på positionerna");
        }
        catch (IndexOutOfBoundsException e)
        {
            out.println ("minst en position utanför vektorn");
        }
    }

    public static void metod2 ()
    {
        out.println ("flera ord i en rad: ");
        String    rad = in.nextLine ();
        String[]    ord = rad.split ("\\s");
        out.println (Arrays.toString (ord));
        out.println ();

        out.print ("positioner för två ord: ");
        int    p1 = in.nextInt ();
        int    p2 = in.nextInt ();
        while (!(p1 <= p2  &&  p1 >= 0  &&  p2 < ord.length))
        {
            out.print ("felaktiga positioner, upprepa: ");
            p1 = in.nextInt ();
            p2 = in.nextInt ();
        }

        Vektorer.sortera (ord, p1, p2);
        out.println (Arrays.toString (ord));
    }
}
```

a) Vad gör metoden `sortera` i klassen `Vektorer`? Åskådliggör algoritmen i metoden.

b) Hur reagerar metoden `sortera` på de undantagssituationer som kan uppstå?

c) Vad händer när programmet `StrategierMedUndantag` exekveras? Går det att avgöra vilken av metoderna – `metod0`, `metod1` eller `metod2` – som har anropats? I vilka situationer kastas undantag under exekveringen?

d) Hur hanterar metoderna `metod0`, `metod1` och `metod2` de undantag som kan uppstå när metoden `sortera` exekveras? Vilken av de utnyttjade strategierna är bäst?

e) I metoden `sortera` kan undantag av typen `ArrayIndexOutOfBoundsException` uppstå. I vilket `catch`-block i metoden `metod1` fångas sådana undantag? Varför är det möjligt?

f) Hur kan man fånga alla undantag som kan uppstå i metoden `metod1` i ett enda `catch`-block? Ange flera möjliga lösningar.

Övning 3

```
import java.io.*;    // InputStreamReader, IOException
import static java.lang.System.out;

class FramtvingaUndantagshantering
{
    public static void main (String[] args)
    {
        out.println ("mata in en text, avsluta med |:");
        String    text = mataInText ('|');
        out.println ();

        out.println (text);
    }

    public static String mataInText (char slutTecken)
    {
        InputStreamReader    in = new InputStreamReader (System.in);

        StringBuilder    sb = new StringBuilder ();
        char    c = (char) in.read ();
        while (c != slutTecken)
        {
            sb.append (c);
            c = (char) in.read ();
        }

        return sb.toString ();
    }
}
```

a) Följ stegen i metoden `mataInText`. Kan en text bestående av flera rader matas in med denna metod?

b) Varför går det inte att kompilera programmet? Hur kan de undantag som kan uppstå deklareras, och hur kan de fångas och hanteras?

Kapitel 12
Inmatning och utmatning

Övning 1

```
import java.io.*;
// PrintWriter, InputStreamReader, BufferedReader, IOException

class MataInOrd
{
    public static void main (String[] args) throws IOException
    {
        PrintWriter    out = new PrintWriter (System.out, true);
        InputStreamReader    isr = new InputStreamReader (System.in);
        BufferedReader       br = new BufferedReader (isr);

        out.println ("flera ord:");
        String    ord1 = mataInOrd (isr);
        String    ord2 = mataInOrd (isr);
        String    resten = br.readLine ();
        out.println (ord1);
        out.println (ord2);
        out.println (resten);
        out.println ();

        out.println ("flera ord:");
        ord1 = mataInOrd (br);
        ord2 = mataInOrd (br);
        resten = br.readLine ();
        out.println (ord1);
        out.println (ord2);
        out.println (resten);
    }

    public static String mataInOrd (InputStreamReader in)
                                                throws IOException
    {
        char    c = (char) in.read ();
        while (Character.isWhitespace (c))
            c = (char) in.read ();
```

```
        StringBuilder    sb = new StringBuilder ();
        while (!Character.isWhitespace (c))
        {
            sb.append (c);
            // in.mark (1);          // (1)
            c = (char) in.read ();
        }
        // in.reset ();              // (2)

        // in.close ();              // (3)

        return sb.toString ();
    }

    public static String mataInOrd (BufferedReader in)
                                       throws IOException
    {
        char    c = (char) in.read ();
        while (Character.isWhitespace (c))
            c = (char) in.read ();

        StringBuilder    sb = new StringBuilder ();
        while (!Character.isWhitespace (c))
        {
            sb.append (c);
            in.mark (1);
            c = (char) in.read ();
        }
        in.reset ();

        // in.close ();             // (4)

        return sb.toString ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Följ noggrant stegen i de två metoderna som heter `mataInOrd`. Vad är skillnaden?

c) Vad händer om satsen `(1)`, eller satsen `(2)`, inkluderas i programmet?

d) Vad händer om satsen `(3)`, eller satsen `(4)`, inkluderas i programmet?

e) Följ inmatningsbufferten och objektet av typen `StringBuilder` under en exekvering av metoden `mataInOrd` – den metoden som har en parameter av typen `BufferedReader`.

f) Hur kan ett objekt av typen `java.util.Scanner` användas för ordinmatning?

Övning 2

```
import java.io.*;
// File, PrintWriter, FileInputStream, InputStreamReader, IOException

class RaderFranOlikaKallor
{
    public static void main (String[] args)
    {
        PrintWriter    out = new PrintWriter (System.out, true);
        InputStreamReader  in = new InputStreamReader (System.in);
        File    fil = new File ("fil.txt");

        try (
            PrintWriter    fout = new PrintWriter (fil);
            InputStreamReader    fin = new InputStreamReader (
                                        new FileInputStream (fil)) )
        {
            out.println ("två rader:");
            String    rad1 = mataInRad (in);
            String    rad2 = mataInRad (in);
            out.println (rad1);
            out.println (rad2);
            out.println ();

            fout.println (rad1);
            fout.println (rad2);
            // fout.flush ();                     // (1)
            rad1 = mataInRad (fin);
            rad2 = mataInRad (fin);
            out.println (rad1);
            out.println (rad2);
        }
        catch (IOException e)
        {
            e.printStackTrace ();
        }
    }

    public static String mataInRad (InputStreamReader in)
                                            throws IOException
    {
        String    radslut = System.getProperty ("line.separator");
        int    antal = radslut.length ();
        StringBuilder    sb = new StringBuilder ();
        char    c = (char) in.read ();
        while (!radslut.contains ("" + c))
        {
            sb.append (c);
            c = (char) in.read ();
```

```
        }
        for (int i = 0; i < antal - 1; i++)
            c = (char) in.read ();
        String    rad = sb.toString ();

        return rad;
    }
}
```

a) Vad händer när det här programmet exekveras? Vad händer när satsen `(1)` inkluderas?

b) Metoden `mataInRad` kan mata in en rad både från standardinmatningsenheten och från en fil. Hur är det möjligt?

c) Följ noggrant stegen i metoden `mataInRad`. Vilken funktion uppfyller `for`-loopen?

d) Hur kan man mata in rader med ett objekt av typen `java.io.BufferedReader`? Går det att mata in både från standardinmatningsenheten och från en fil? Kan ett objekt av typen `java.util.Scanner` användas i stället?

Övning 3

```
import java.io.*;
// File, FileOutputStream, OutputStreamWriter,
// FileReader, BufferedReader, IOException

class MataUtTal
{
    public static void main (String[] args)
    {
        OutputStreamWriter  out = new OutputStreamWriter (System.out);
        File    fil = new File ("fil.txt");

        try (
            OutputStreamWriter    fout = new OutputStreamWriter (
                                        new FileOutputStream (fil));
            BufferedReader    fin = new BufferedReader (
                                            new FileReader (fil)) )
        {
            int    n1 = 12345;
            int    n2 = 10011;
            println (fout, n1);
            println (fout, n2);

            int   i1 = Integer.parseInt (mataInOrd (fin));
            int   i2 = Integer.parseInt (mataInOrd (fin));
            println (out, i1);
            println (out, i2);
        }
```

```
        catch (IOException e)
        {
            e.printStackTrace ();
        }
    }

    public static void println (OutputStreamWriter out, int n)
                                            throws IOException
    {
        String    inS = String.valueOf (n);
        String    radslut = System.getProperty ("line.separator");
        String    outS = inS + radslut;
        out.write (outS);
        out.flush ();
    }

    public static String mataInOrd (BufferedReader in)
                                        throws IOException
    {
        char    c = (char) in.read ();
        while (Character.isWhitespace (c))
            c = (char) in.read ();

        StringBuilder    sb = new StringBuilder ();
        while (!Character.isWhitespace (c))
        {
            sb.append (c);
            in.mark (1);
            c = (char) in.read ();
        }
        in.reset ();

        return sb.toString ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Metoden `println` kan mata ut heltal både till standardutmatningsenheten och till en fil. Hur är det möjligt?

c) Följ noggrant stegen i metoden `println`. Hur bearbetas heltalet före utskriften? Vad händer om anropet till metoden `flush` bortkommenteras?

d) Hur kan man mata ut tal med ett objekt av typen `java.io.PrintWriter`? Går det att mata ut både till standardutmatningsenheten och till en fil?

Övning 4

```
import java.io.*;    // File, PrintWriter
import java.util.*;  // Scanner

class ValjCharset
{
    public static void main (String[] args)
    {
        PrintWriter    out = new PrintWriter (System.out, true);
        File    fil = new File ("fil.txt");
        PrintWriter    fout = null;
        Scanner    fin = null;

        try
        {
            String     charset = "US-ASCII";
            // charset = "UTF-16";                  // (1)
            fout = new PrintWriter (fil, charset);
            // charset = "ISO-8859-1";              // (2)
            // charset = "UTF-8";                   // (3)
            // charset = "UTF-16";                  // (4)
            // charset = "ISO-100";                 // (5)
            fin = new Scanner (fil, charset);

            fout.println ("[gryningen, middag, skymningen]");
            fout.flush ();

            String    s = fin.nextLine ();
            out.println (s);
        }
        catch (Exception e)
        {
            e.printStackTrace ();
        }
        finally
        {
            fout.close ();
            fin.close ();
        }
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer om någon av satserna `(2)`, `(3)`, `(4)` eller `(5)` inkluderas? Vad händer i fall att satsen `(1)` inkluderas? Vad händer om man inkluderar både satsen `(1)` och någon av satserna `(2)`, `(3)`, `(4)` eller `(5)`?

Övning 5

```
import java.io.*;
// PrintWriter, File, FileOutputStream, DataOutputStream,
// InputStream, FileInputStream, BufferedInputStream, IOException

class MataInHeltal
{
    public static PrintWriter    out = null;

    public static void main (String[] args)
    {
        out = new PrintWriter (System.out, true);
        File    fil = new File ("fil.dat");

        DataOutputStream    fout = null;
        InputStream    is = null;
        try
        {
            fout = new DataOutputStream (new FileOutputStream (fil));
            int   p = -254500007;
            out.println (p);
            // out.println (Integer.toBinaryString (p));    // (1)
            out.println ();
            fout.writeInt (p);
            fout.flush ();

            is = new FileInputStream (fil);
            // is = new BufferedInputStream (is);     // (*)
            int    n = mataInInt (is);
            out.println (n);
        }
        catch (IOException e)
        {
            e.printStackTrace ();
        }
        finally
        {
            try
            {
                fout.close ();
                is.close ();
            }
            catch (IOException e)
            {
                e.printStackTrace ();
            }
        }
    }
```

```
    public static int mataInInt (InputStream in) throws IOException
    {
        int    n = 0x00000000; // alla bitar noll
        // out.println (Integer.toBinaryString (n));    // (2)
        int    k = 0;
        for (int i = 0; i < 4; i++)
        {
            k = in.read ();
            n = n  |  (k  <<  (3 - i) * 8);
            // << flyttar bitar till vänster ett angivet antal
            // positioner
            // | bitvis ELLER
            // out.println (Integer.toBinaryString (n));    // (3);
        }
        // out.println ();    // (4)

        return n;
    }
}
```

a) Vad händer när det här programmet exekveras?

b) På vilket sätt matas in ett heltal av typen `int` i metoden `mataInInt`? Kan man på ett liknande sätt mata in tal av andra typer?

c) Vad händer om satserna `(1)`, `(2)`, `(3)` och `(4)` inkluderas i programmet?

d) Vad händer om satsen `(*)` inkluderas i programmet?

e) Man använder en referens av typen `java.io.InputStream` för att referera både till ett objekt av tpen `java.io.FileInputStream` och till ett objekt av typen `java.io.BufferedInputStream`. Varför är det möjligt?

f) Hur kan ett objekt av typen `java.io.DataInputStream` användas för att mata in ett heltal av typen `int`?

g) Tack vare `finnally`-blocket ska de strömmar som använder filen stängas oavsett om ett undantag uppstår eller inte. Kan man åstadkomma samma sak på ett annat sätt?

Övning 6

```
import java.io.*;
// PrintWriter, File, OutputStream, FileOutputStream,
// BufferedOutputStream, FileInputStream, DataInputStream, IOException

class MataUtHeltal
{
    public static PrintWriter    sout = null;
```

```
public static void main (String[] args)
{
    sout = new PrintWriter (System.out, true);
    File    fil = new File ("fil.dat");

    OutputStream    os = null;
    DataInputStream    fin = null;
    try
    {
        os = new FileOutputStream (fil);
        // os = new BufferedOutputStream (os);         // (*)
        int   p = -254500007;
        sout.println (p);
        // sout.println (Integer.toBinaryString (p));  // (1)
        sout.println ();
        mataUtInt (os, p);

        fin = new DataInputStream (new FileInputStream (fil));
        int    n = fin.readInt ();
        sout.println (n);
    }
    catch (IOException e)
    {
        e.printStackTrace ();
    }
    finally
    {
        try
        {
            os.close ();
            fin.close ();
        }
        catch (IOException e)
        {
            e.printStackTrace ();
        }
    }
}

public static void mataUtInt (OutputStream out, int n)
                                        throws IOException
{
    int    bitmask = 0xFF000000;
    // första åtta bitar ettor, andra bitar nollor
    int    k = 0;
    for (int i = 0; i < 4; i++)
    {
        k = bitmask >> i * 8;
        // sout.println (Integer.toBinaryString (k));  // (2)
        k = n & k;    // bitvis OCH
        // sout.println (Integer.toBinaryString (k));  // (3)
```

```
            k = k >> (3 - i) * 8;
            // flytta bitar till höger
            // sout.println (Integer.toBinaryString (k));  // (4)
            // sout.println ();                            // (5)

            out.write (k);
        }
        out.flush ();
        // sout.println ();                                // (6)
    }
}
```

a) Vad händer när det här programmet exekveras?

b) På vilket sätt matas ut ett heltal av typen `int` i metoden `mataUtInt`? Kan man på ett liknande sätt mata ut tal av andra typer?

c) Vad händer om satserna `(1)`, `(2)`, `(3)`, `(4)`, `(5)` och `(6)` inkluderas i programmet?

d) Vad händer om satsen `(*)` inkluderas i programmet?

e) Man använder en referens av typen `java.io.OutputStream` för att referera både till ett objekt av tpen `java.io.FileOutputStream` och till ett objekt av typen `java.io.BufferedOutputStream`. Varför är det möjligt?

f) Hur kan ett objekt av typen `java.io.DataOutputStream` användas för att mata ut ett heltal av typen `int`?

Övning 7

```
// FInteger.java

public class FInteger // implements java.io.Serializable
{
    public int    n;
}

// FNumber.java

public abstract class FNumber implements java.io.Serializable
{
}

// FDouble.java

public class FDouble // extends FNumber
{
    public double    d;
}
```

```
// SerializableObjektEllerEj.java

import java.io.*;
// FileOutputStream, ObjectOutputStream,
// FileInputStream, ObjectInputStream, IOException

class SerializableObjektEllerEj
{
    public static void main (String[] args)
    {
        String    f = "objektfil.dat";

        try (
            ObjectOutputStream    fout = new ObjectOutputStream (
                                         new FileOutputStream (f));
            ObjectInputStream    fin = new ObjectInputStream (
                                         new FileInputStream (f)) )
        {
            FInteger    obj1 = new FInteger ();
            obj1.n = 5;
            fout.writeObject (obj1);

            FDouble    obj2 = new FDouble ();
            obj2.d = 1.6;
            fout.writeObject (obj2);
            fout.flush ();

            FInteger    fi = (FInteger) fin.readObject ();
            System.out.println (fi.n);
            FDouble     fd = (FDouble) fin.readObject ();
            System.out.println (fd.d);
        }
        catch (IOException | ClassNotFoundException e)
        {
            e.printStackTrace ();
        }
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer om kommentarsmarkeringen i klassen `FInteger` tas bort?

c) Vad händer om även kommentarmarkeringen i klassen `FDouble` tas bort?

d) Varför utförs typomvandlingen vid inläsningen?

e) När kastas ett undantag av typen `java.lang.ClassNotFoundException`? Är det nödvändigt att hantera undantag av den typen?

Övning 8

```
// FInteger.java

public class FInteger // implements java.io.Serializable
{
    private int    n;

    public FInteger (int m)
    {
        n = m;
    }

    public String toString ()
    {
        return "" + n;
    }
}

// FIntegerPair.java

import java.io.*;    // Serializable

public class FIntegerPair implements Serializable
{
    public FInteger    i1 = new FInteger (5);
    public FInteger    i2 = new FInteger (10);
}

// SerializableRefereradeObjektEllerEj.java

import java.io.*;
// FileOutputStream, ObjectOutputStream,
// FileInputStream, ObjectInputStream, IOException

class SerializableRefereradeObjektEllerEj
{
    public static void main (String[] args)
    {
        String    f = "objektfil.dat";

        try (
            ObjectOutputStream    fout = new ObjectOutputStream (
                                            new FileOutputStream (f));
            ObjectInputStream    fin = new ObjectInputStream (
                                               new FileInputStream (f)) )
        {
            FIntegerPair    obj = new FIntegerPair ();
            fout.writeObject (obj);
            fout.flush ();

            FIntegerPair   ip = (FIntegerPair) fin.readObject ();
```

```
            System.out.println (ip.i1);
            System.out.println (ip.i2);
        }
        catch (IOException | ClassNotFoundException e)
        {
            e.printStackTrace ();
        }
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Vad händer om kommentarsmarkeringen i klassen `FInteger` tas bort?

c) Ett objekt av typen `FIntegerPair` innehåller två referenser, som refererar till två objekt av typen `FInteger`. Vad sparas när ett objekt av typen `FIntegerPair` sparas – sparas dessa referenser?

d) Vilken metod anropas när objekten `ip.i1` och `ip.i2` skrivs ut?

Övning 9

Det finns en textfil `lander.txt`, som innehåller uppgifter om olika länder:

```
Länder: namn, huvudstad, yta (kvadratkilometer), befolkning

Albanien, Tirana, 28748, 3563112
Bosnien och Hercegovina, Sarajevo, 51129, 4700000
Bulgarien, Sofia, 110910, 7450349
Grekland, Aten, 131940, 10668058
Italien, Rom, 301230, 57715625
Kroatien, Zagreb, 56542, 4495904
Makedonien, Skopje, 25713, 2100000
Montenegro, Podgorica, 13812, 684736
Serbien, Belgrad, 78474, 7678991
Slovenien, Ljubljana, 20273, 2000000
Sverige, Stockholm, 449964, 9223766
Tyskland, Berlin, 357021, 83251851
```

Filen används i följande program:

```
// Lander.java

import java.io.*;  // File, PrintWriter, RandomAccessFile, IOException
import java.util.*;  // Scanner

class Lander
{
    public static final int    NAMN_LANGD = 30;
```

```
public static final int    HUVUDSTAD_LANGD = 20;
public static final int    POST_LANGD = 2 * NAMN_LANGD +
                              + 2 * HUVUDSTAD_LANGD + 2 * 4;

public static void main (String[] args)
{
    PrintWriter    out = new PrintWriter (System.out, true);
    File    textFil = new File ("lander.txt");
    File    dataFil = new File ("lander.dat");

    try (Scanner fin = new Scanner (textFil);
      RandomAccessFile fio = new RandomAccessFile (dataFil, "rw"))
    {
        String     rad = "";
        fin.nextLine ();
        fin.nextLine ();
        String[]    ord = null;
        while (fin.hasNextLine ())
        {
            rad = fin.nextLine ();
            ord = rad.split (", ");
            fio.writeChars (setLangd (ord[0], NAMN_LANGD));
            fio.writeChars (setLangd (ord[1], HUVUDSTAD_LANGD));
            fio.writeInt (Integer.parseInt (ord[2]));
            fio.writeInt (Integer.parseInt (ord[3]));
        }

        int    n = 10;
        // n = 1;                                    // (1)
        fio.seek (n * POST_LANGD);
        String    namn = readChars (fio, NAMN_LANGD).trim ();
        String    huvudstad = readChars (fio,
                                         HUVUDSTAD_LANGD).trim ();
        int    yta = fio.readInt ();
        int    antaletInvanare = fio.readInt ();
        out.println ("[" + namn + ", " + huvudstad + ", "
                   + yta + ", " + antaletInvanare + "]");
    }
    catch (IOException e)
    {
        e.printStackTrace ();
    }
}

public static String setLangd (String s, int langd)
{
    StringBuilder    sb = new StringBuilder (s);
    int    antal = langd - s.length ();
    for (int i = 0; i < antal; i++)
        sb.append (" ");
```

```
        return sb.toString ();
    }

    public static String readChars (RandomAccessFile in, int antal)
                                                    throws IOException
    {
        StringBuilder    sb = new StringBuilder ();
        for (int i = 0; i < antal; i++)
            sb.append (in.readChar ());

        return sb.toString ();
    }
}
```

a) Vad händer när det här programmet exekveras?

b) Hur ser ut en post i filen `lander.dat`? Vilka delar ingår i posten och hur långa är de?

c) Vad händer om satsen `(1)` inkluderas?

d) Är det bättre att lagra data i en textfil, eller i en fil med direkt åtkomst?

Problem 1

Skapa nödvändiga filer

Problem: skapa nya filer utifrån dem som redan finns

Tre filer innehåller uppgifter om elever i en grundskola. Det finns information om elevernas namn, längder, vikter och annat. Den första filen innehåller uppgifter om elever på lågstadiet, den andra filen gäller mellanstadiet och den tredje filen handlar om högstadiet. Eleverna i en fil är sorterade enligt deras namn.

En skolsköterska behöver ha ytterligare filer. För varje nivå behövs en fil där eleverna är sorterade enligt sina längder och en fil där eleverna ligger ordnade enligt sina vikter. Tre ytterligare filer ska innehålla uppgifter om alla elever i skolan. I den ena filen ska eleverna ligga ordnade enligt sina namn, i den andra filen enligt sina längder och i den tredje filen enligt sina vikter.

Alla de nödvändiga filerna ska skapas.

Uppgift i samband med filerna

Skapa ett program som utgår ifrån de filer som redan finns, och genererar de andra nödvändiga filerna. När en fil som omfattar alla elever i skolan ska skapas, ska man dra nytta av det faktum att motsvarande filer som gäller enskilda nivåer redan är sorterade. Man ska sammanfoga tre sorterade sekvenser i en ny sorterad sekvens.

Problem 2

Länder

A) Skapa en fil med direkt åtkomst utifrån en textfil

Det finns en textfil `lander.txt`, som innehåller uppgifter om olika länder i världen. Uppgigter om ett land finns i en rad i filen. Man anger landets namn, huvudstad, yta, befolkning, valuta och språk (det kan finnas flera språk i ett land). Länderna i filen är sorterade enligt deras namn.

Utifrån denna textfil ska en fil med direkt åtkomst, som heter `lander.dat`, skapas. Filen ska bestå av ett antal lika långa poster – en post per land.

Uppgifter i samband med filerna

1. Skapa ett program `Lander1`, som läser filen `lander.txt` och skapar filen `lander.dat`. Programmet läser sedan in den nyskapade filen, och visar dess innehåll på standardutmatningsenheten.

2. Hur ser en post i filen ut? Rita en sådan post: det ska klart framgå hur många byte varje del i posten tar upp.

B) Söka i filen med direkt åtkomst

Det går att söka i filen `lander.dat`. Utifrån ett lands namn kan andra uppgifter om landet erhållas. Ett lands namn är en nyckel, som på ett entydigt sätt identifierar motsvarande post i filen.

I en sökningsstrategi börjar man med den första posten. Man avläser namnet i posten, och jämför det med det givna namnet. Om de överenstämmer, läses även andra uppgifter i posten. I motsatt fall går man till nästa post och avläser namnet där. Sökningen är sekventiell: man undersöker posterna i tur och ordning, tills den rätta posten hittas.

Eftersom posterna i filen är sorterade enligt ländernas namn, kan den binära sökningen användas. Den mittersta posten prövas först. Det kan hända att namnet i posten överensstämmer med det givna namnet. I så fall avläses även andra uppgifter i posten. Om namnen inte överensstämmer, kan det avgöras om den sökta posten finns framför (om landets namn kommer före än namnet i posten) eller efter den mittersta posten. Sökningen utförs sedan på samma sätt i den del av filen där posten finns. I varje steg halveras det område i filen som återstår att undersöka.

Uppgift i samband med sökningen i filen

Skapa ett program `SokLand1`, som söker uppgifter om olika länder. Användaren anger ett lands namn, varvid programmet hittar motsvarande uppgifter i filen `lander.dat` och visar dem på standardutmatningsenheten. Därefter anges namnet på ett annat land, och allt fortsätter på samma sätt.

Förutom metoden `main`, har programmet även två sökningsmetoder. De båda metoderna söker uppgifter om ett givet land, men de använder olika sökningsstrategier. Den ena metoden använder sekventiell sökning, och den andra metoden använder den binära sökningen. Metoden `main` anropar dessa metoder, och mäter och visar tiden som de kräver.

C) Använda ett register med ländernas namn

Oavsett sökningsstrategi, tar det tid att undersöka posterna i filen med länderna. I stället kan en sorterad sekvens med ländernas namn, ett landsregister, skapas och användas. Ordningsnumret för ett land kan på så sätt hittas snabbt (med den binära sökningen i registret), och därmed platsen för den motsvarande posten i filen. Man går direkt till den rätta posten och avläser den.

När textfilen `lander.txt` läses, kan ländernas namn sparas i en vektor. Den här vektorn kan sparas som ett objekt i en fill med namnet `landsregister.dat`. Ett sökprogram kan läsa in denna vektor och använda den vid sökningen i filen `lander.dat`.

Uppgift i samband med landsregistret

1. Skapa ett program `Lander2`, som läser filen `lander.txt` och skapar filen `lander.dat`. Programmet samtidigt skapar en vektor med ländernas namn, och sparar den som ett objekt i en fil som heter `landsregister.dat`. Programmet läser sedan in de nyskapade filerna, och visar deras innehåll på standardutmatningsenheten.

2. Skapa ett program `SokLand2`, som söker uppgifter om olika länder. Användaren anger ett lands namn, varvid programmet hittar motsvarande uppgifter i filen `lander.dat` och

visar dem på standardutmatningsenheten. Därefter anges namnet på ett annat land, och allt fortsätter på samma sätt.

Programmet läser först in vektorn med ländernas namn från filen `landsregister.dat`. Denna vektor används sedan för att bestämma positioner för de poster i filen som motsvarar de inmatade namnen.

Kapitel 13
Skapa nya objekttyper

Övning 1

Det finns en textfil `Countries.txt`, som har följande utseende:

```
Countries: name, capital city, population
12 countries
Albania, Tirana, 3563112
Bosnia and Herzegovina, Sarajevo, 4700000
Bulgaria, Sofia, 7450349
Greece, Athens, 10668058
Italy, Rome, 57715625
Croatia, Zagreb, 4495904
Macedonia, Skopje, 2100000
Montenegro, Podgorica, 684736
Serbia, Belgrade, 7678991
Slovenia, Ljubljana, 2000000
Sweden, Stockholm, 9223766
Germany, Berlin, 83251851
```

Det finns två Javafiler: `Country.java` och `Countries.java`:

```
// Country.java

import java.io.*;  // Serializable

class Country implements Serializable
{
    private String    name;
    private String    capitalCity;
    private int       population;

    public Country ()
    {
        this.name = "";
        this.capitalCity = "";
        this.population = 0;
    }
```

```
    public Country (String name, String capitalCity, int population)
    {
        this.name = name;
        this.capitalCity = capitalCity;
        this.population = population;
    }

    public String toString ()
    {
       return "[" + name + ", " + capitalCity + ", "
            + population + "]";
    }

    public String getName ()
    {
        return name;
    }

    public String getCapitalCity ()
    {
        return capitalCity;
    }

    public int getPopulation ()
    {
        return population;
    }

    public void setName (String name)
    {
        this.name = name;
    }

    public void setCapitalCity (String capitalCity)
    {
        this.capitalCity = capitalCity;
    }

    public void setPopulation (int population)
    {
        this.population = population;
    }

    public boolean equals (Country country)
    {
        return this.name.equals (country.name)  &&
               this.capitalCity.equals (country.capitalCity)  &&
               this.population == country.population;
    }
}
```

```
// Countries.java

import java.io.*;
// File, PrintWriter, FileOutputStream, ObjectOutputStream,
// FileInputStream, ObjectInputStream, IOException
import java.util.*;  // Scanner

class Countries
{
    public static void main (String[] args)
    {
        PrintWriter    out = new PrintWriter (System.out, true);
        File    textFile = new File ("countries.txt");
        File    dataFile = new File ("countries.dat");

        try (Scanner fin1 = new Scanner (textFile);
             ObjectOutputStream fout = new ObjectOutputStream (
                                       new FileOutputStream (dataFile));
             ObjectInputStream fin2 = new ObjectInputStream (
                                       new FileInputStream (dataFile)) )
        {
            String     line = "";
            fin1.nextLine ();
            int    countCountries = fin1.nextInt ();
            fin1.nextLine ();
            Country[]    countries1 = new Country[countCountries];
            String[]    words = null;
            for (int pos = 0; pos < countCountries; pos++)
            {
                line = fin1.nextLine ();
                words = line.split (", ");
                countries1[pos] = new Country (
                    words[0], words[1], Integer.parseInt (words[2]));
            }
            fout.writeObject (countries1);
            fout.flush ();

            Country[]    countries2 = (Country[]) fin2.readObject ();
            for (int pos = 0; pos < countries2.length; pos++)
                out.println (countries2[pos]);
            out.println ();

            int    pos = 0;
            while (!countries2[pos].getCapitalCity ().equals (
                                                        "Stockholm"))
                pos++;
            out.println (countries2[pos]);
        }
        catch (IOException | ClassNotFoundException e)
```

```
        {
            e.printStackTrace ();
        }
    }
}
```

a) Vad händer när programmet `Countries` exekveras?

b) Vilken utskrift skapas på standardutmatningsenheten?

c) Åskådliggör vektorn `countries2`.

d) Vilken typ av sökning används i `while`-loopen? Kan man använda binär sökning?

Övning 2

```
// DataOutputStream.java

class DataOutputStream implements AutoCloseable
{
    private java.io.OutputStream    os;

    public DataOutputStream (java.io.OutputStream os)
    {
        this.os = os;
    }

    public void writeInt (int n) throws java.io.IOException
    {
        int    bitmask = 0xFF000000;
        int    k = 0;
        for (int i = 0; i < 4; i++)
        {
            k = bitmask >> i * 8;
            k = n & k;
            k = k >> (3 - i) * 8;

            os.write (k);
        }
        os.flush ();
    }

    public void flush () throws java.io.IOException
    {
        os.flush ();
    }

    public void close () throws java.io.IOException
    {
```

```
        os.close ();
    }
}



// DataInputStream.java

class DataInputStream implements AutoCloseable
{
    private java.io.InputStream    is;

    public DataInputStream (java.io.InputStream is)
    {
        this.is = is;
    }

    public int readInt () throws java.io.IOException
    {
        int    n = 0x00000000;
        int    k = 0;
        for (int i = 0; i < 4; i++)
        {
            k = is.read ();
            n = n  |  (k  <<  (3 - i) * 8);
        }

        return n;
    }

    public void close () throws java.io.IOException
    {
        is.close ();
    }
}


// IntegersToFromFile.java

class IntegersToFromFile
{
    public static void main (String[] args)
    {
        java.io.PrintWriter    out =
                        new java.io.PrintWriter (System.out, true);
        java.io.File    file = new java.io.File ("file.dat");


        try (DataOutputStream    fout = new DataOutputStream (
                                new java.io.FileOutputStream (file));
```

```
             DataInputStream    fin = new DataInputStream (
                                new java.io.FileInputStream (file)) )
        {
            int   p = -254500007;
            out.println (p);
            out.println ();

            fout.writeInt (p);
            fout.flush ();

            int    n = fin.readInt ();
            out.println (n);
        }
        catch (java.io.IOException e)
        {
            e.printStackTrace ();
        }
    }
}
```

a) Vad händer när programmet `IntegersToFromFile` exekveras?

b) Åskådliggör objekten som refereras av referenserna `fout` och `fin`.

c) Åskådliggör de algoritmer som används i metoderna `writeInt` och `readInt`.

d) Modifiera programmet `IntegersToFromFile` så att standardklasser från paketet `java.io` används: `DataOutputStream` för utmatning av heltalet till filen, och `DataInputStream` för inmatning av heltalet.

Övning 3

```
// Point.java

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

    public int    x;
    public int    y;

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

    public String toString ()
    {
```

```
        return "(" + x + ", " + y + ")";
    }

    public void setLocation (int x, int y)
    {
        this.x = x;
        this.y = y;
    }


    public void translate (int dx, int dy)
    {
        x += dx;
        y += dy;
    }
}


// PointMover.java

// import java.awt.Point;                   //  (1)

class PointMover
{
    public static final int[][]   steps = {{-1, -2}, {-2, -1},
           {-2, 1}, {-1, 2}, {1, 2}, {2, 1},{2, -1}, {1, -2}};

    private Point    p;
    private final Point    homePoint;
    private int    step = 0;

    public PointMover (Point p)
    {
        this.p = p;
        homePoint = new Point (p.x, p.y);    //  (2)
        // homePoint = p;                    //  (3)
    }

    public void movePoint ()
    {
        p.translate (steps[step][0], steps[step][1]);
        step = (step + 1) % steps.length;
    }

    public void movePointHome ()
    {
        p.x = homePoint.x;
        p.y = homePoint.y;
    }
}
```

```
// PointMovements.java

class PointMovements
{
    public static void main (String[] args)
    {
        java.io.PrintWriter    out = new java.io.PrintWriter (
                                              System.out, true);

        Point    point = new Point (0, 0);
        // point = Point.ORIGO;                  // (4)
        out.println (point);
        out.println ();

        PointMover     mover = new PointMover (point);
        for (int i = 0; i < PointMover.steps.length; i++)
        {
            mover.movePoint ();
            out.println (point);
            // out.println (Point.ORIGO);        // (5)
            mover.movePointHome ();
            // out.println (point);              // (6);
        }
    }
}
```

a) Vad händer när programmet `PointMovements` exekveras?

b) Vilken utskrift skapas? Vad händer om satsen `(6)` inkluderas?

c) Vad händer om satsen `(1)` inkluderas i programmet?

d) Åskodliggör objektet (som refereras av referensen) `mover`.

e) Vad händer i metoderna `movePoint` och `movePointHome`?

f) Vad händer om satsen `(3)` inkluderas i stället för satsen `(2)`? Prova att inkludera även satsen `(6)`.

g) Vad händer om satserna `(4)` och `(5)` inkluderas? Prova att inkludera även satsen `(6)`.

Övning 4

```
// StringCreator.java

class StringCreator
{
    public static final int    DEFAULT_INITIAL_CAPACITY = 10;
```

```
    public static final int    INCREASE_VALUE = 50;  // 50%

    private char[]    chars;
    private int    charCount;

    public StringCreator ()
    {
        this.chars = new char[DEFAULT_INITIAL_CAPACITY];
        charCount = 0;
    }

    public StringCreator (int capacity)
    {
        this.chars = new char[capacity];
        charCount = 0;
    }

    public void append (char c)
    {   /*
        if (charCount >= chars.length)
        {
            int    newCapacity = 1 + chars.length
                                 + INCREASE_VALUE * chars.length / 100;
            char[]    newChars = new char[newCapacity];
            int    pos = 0;
            for (pos = 0; pos < chars.length; pos++)
                newChars[pos] = chars[pos];
            chars = newChars;
        }
        */

        this.chars[charCount++] = c;
    }

    public String toString ()
    {
        return new String (chars, 0, charCount);
    }
}

// UseStringCreator.java

class UseStringCreator
{
    public static void main (String[] args)
    {
        StringCreator    sc = new StringCreator (4);
        // sc = new StringCreator ();   // (1)
        char[]    c = {'a', 'b', 'c', 'd'};
        for (int i = 0; i < c.length; i++)
```

```
            sc.append (c[i]);
        // sc.append ('e');             // (2)

        System.out.println (sc);
    }
}
```

a) Vad händer när programmet `UseStringCreator` exekveras?

b) Rita objektet `sc`, och följ ändringar i det efter varje anrop till metoden `append`.

c) Vad händer om satsen `(2)` inkluderas i programmet?

d) Vad händer om både satsen `(1)` och satsen `(2)` inkluderas? Rita objektet `sc` och följ ändringar i det.

e) Vad händer om man inkluderar satsen `(2)` och tar bort kommentarmarkeringarna i metoden `append`? Rita objektet `sc` och följ ändringar i det.

Övning 5

```
// StringCreator.java

class StringCreator
{
    private char[]    chars;
    private int       charCount;

    public StringCreator (String s)
    {
        this.chars = new char[s.length () + 10];
        for (int pos = 0; pos < s.length (); pos++)
            this.chars[pos] = s.charAt (pos);
        charCount = s.length ();     // (*)
    }

    public StringCreator (char[] chars)
    {
        this.chars = new char[chars.length + 10];
        for (int pos = 0; pos < chars.length; pos++)
            this.chars[pos] = chars[pos];
        // this.chars = chars;       // (1)
        charCount = chars.length;    // (*)
    }

    public void insert (char c, int pos)
                    throws ArrayIndexOutOfBoundsException
    {
        if (pos > charCount)
```

```
            throw new ArrayIndexOutOfBoundsException (
                                "bad index: " + pos);

        for (int p = charCount - 1; p >= pos; p--)
            chars[p + 1] = chars[p];
        this.chars[pos] = c;
        charCount++;                // (*)
    }

    public char[] getChars ()
    {
        char[]    c = new char[charCount];
        for (int pos = 0; pos < charCount; pos++)
            c[pos] = chars[pos];
        // c = chars;                // (2)

        return c;
    }

    public void setChars (char[] chars)
    {
        this.chars = chars;
        charCount = chars.length;   // (*)
    }

    public String toString ()
    {
        return new String (chars, 0, charCount);
    }
}

// UseStringCreator.java

class UseStringCreator
{
    public static void main (String[] args)
    {
        StringCreator    sc = new StringCreator ("abcde");
        char[]    digits = {'1', '2', '3', '4'};
        int    index = 0;
        // index = 2;                // (3)
        // index = 5;                // (4)
        // index = 6;                // (5)
        for (int i = 0; i < digits.length; i++)
            sc.insert (digits[i], index);

        char[]    ch = sc.getChars ();
        ch[0] = 'A';
        System.out.println (sc);

        StringCreator    dsc = new StringCreator (digits);
```

```
        digits[0] = '0';
        System.out.println (dsc);

        // dsc.setChars (digits);       // (6)
        // digits[1] = '9';             // (7)
        // System.out.println (dsc);    // (8)
    }
}
```

a) Vad händer när programmet `UseStringCreator` exekveras?

b) Rita objektet `sc`, och följ ändringar i det efter varje anrop till metoden `insert`. Vad händer om någon av satserna `(3)`, `(4)` eller `(5)` inkluderas?

c) Rita vektorn `ch`. Vad händer om satsen `(2)` inkluderas?

d) Rita vektorn `digits` och objektet `dsc`. Vad händer om satsen `(1)` inkluderas?

e) Vad händer om satserna `(6)` och `(8)` inkluderas? Vad händer om även satsen `(7)` inkluderas? Implementera metoden `setChars` så att det aktuella objektet inte beror på argumentvektorn efter metodens exekvering.

f) Vad händer om någon av de satser som är markerade med `(*)` bortkommenteras?

Övning 6

```
// CharSequence.java

class CharSequence
{
    private char[]    chars;
    private int    charCount;

    public CharSequence (char[] chars)
    {
        this.chars = new char[chars.length + 100];
        for (int pos = 0; pos < chars.length; pos++)
            this.chars[pos] = chars[pos];
        charCount = chars.length;
    }

    public void append (char c)
    {
        this.chars[charCount++] = c;
    }

    public String toString ()
    {
        return new String (chars, 0, charCount);
```

```
    }


    public class CharSequenceIterator
    {
        private int    pos;

        public CharSequenceIterator ()
        {
            pos = (charCount > 0)? 0 : -1;
        }

        public boolean hasChar ()
        {
             return pos >= 0  &&  pos < charCount;
        }

        public char currentChar ()
        {
            return chars[pos];
        }

        public void moveForward ()
        {
            if (pos < charCount)
                pos++;
        }

        public void moveBackward ()
        {
            if (pos >= 0)
                pos--;
        }
    }
}

// UseCharSequence.java

class UseCharSequence
{
    public static void main (String[] args)
    {
        char[]    characters = {'a', 'b', 'c', 'd', 'e'};
        CharSequence    cs = new CharSequence (characters);
        System.out.println (cs);
        System.out.println ();

        CharSequence.CharSequenceIterator    iterator =
                            cs.new CharSequenceIterator ();
        char    c = 0;
        while (iterator.hasChar ())
```

```
        {
            c = iterator.currentChar ();
            System.out.print (c + " ");

            iterator.moveForward ();
        }
        System.out.println ();

        iterator.moveBackward ();             // (1)
        while (iterator.hasChar ())
        {
            c = iterator.currentChar ();
            System.out.print (c + " ");

            iterator.moveBackward ();
        }
        System.out.println ();

        iterator.moveForward ();             // (2)
        while (iterator.hasChar ())
        {
            c = iterator.currentChar ();
            System.out.print (c + " ");

            iterator.moveForward ();
        }
        System.out.println ();
    }
}
```

a) Vad händer när programmet `UseCharSequence` exekveras?

b) Vad händer om satsen `(1)`, eller satsen `(2)`, bortkommenteras?

c) Åskådliggör den algoritm som används i metoden `append`. Bestäm tidskomplexiteten för den algoritmen.

d) Rita objekten (som refereras av referenserna) `cs` och `iterator`. Vilken relation finns mellan dessa objekt?

e) Följ ändringar i objektet `iterator` under en iteration genom teckensekvensen. Vad händer när teckensekvensen traverseras baklänges?

Övning 7

```
// CharSequence.java

class CharSequence
{
```

```
private class Node
{
    public char    c;
    public Node    nextNode;

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


private Node    firstNode = null;

public CharSequence (char[] chars)
{
    if (chars.length != 0)
    {
        Node    node = new Node (chars[0]);
        firstNode = node;                              // (1)
        for (int pos = 1; pos < chars.length; pos++)
        {
            node.nextNode = new Node (chars[pos]);
            node = node.nextNode;                      // (2)
        }
    }
}

public void append (char c)
{
    if (firstNode == null)
        firstNode = new Node (c);
    else
    {
        Node    node = firstNode;
        while (node.nextNode != null)
            node = node.nextNode;
        node.nextNode = new Node (c);
    }
}

public String toString ()
{
    String    s = "";
    Node    node = firstNode;
    while (node != null)
    {
        s = s + node.c;
        node = node.nextNode;
    }
```

```
        return s;
    }

    public int charCount ()
    {
        int     count = 0;
        Node    node = firstNode;
        while (node != null)
        {
            count++;
            node = node.nextNode;
        }

        return count;
    }


    public class CharSequenceIterator
    {
        private Node    pos;

        public CharSequenceIterator ()
        {
            pos = (firstNode != null)? firstNode : null;
        }

        public boolean hasChar ()
        {
            return pos != null;
        }

        public char currentChar ()
        {
            return pos.c;
        }

        public void moveForward ()
        {
            if (pos != null)
                pos = pos.nextNode;
        }
    }
}

// UseCharSequence.java

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

```
        char[]    characters = {'a', 'b', 'c', 'd', 'e'};
        CharSequence    cs = new CharSequence (characters);
        System.out.println (cs);
        System.out.println (cs.charCount ());
        System.out.println ();

        CharSequence.CharSequenceIterator    iterator =
                            cs.new CharSequenceIterator ();
        char    c = 0;
        while (iterator.hasChar ())
        {
            c = iterator.currentChar ();
            System.out.print (c + " ");

            iterator.moveForward ();                    // (3)
        }
        System.out.println ();
    }
}
```

a) Vad händer när programmet `UseCharSequence` exekveras?

b) Åskådliggör den algoritm som används i konstruktorn `CharSequence`. Vad händer om satsen `(1)`, eller satsen `(2)`, bortkommenteras?

c) Åskådliggör den algoritm som används i metoden `append`. Bestäm tidskomplexiteten för den algoritmen och bevisa den. Måste man behandla fallet `firstNode == null` som ett speciellt fall?

d) Åskådliggör den algoritm som används i metoden `toString`. Bestäm den algoritmens tidskomplexitet. På vilket sätt liknar metoderna `toString` och `charCount`?

e) Rita objekten (som refereras av referenserna) `cs` och `iterator`. Vilken relation finns mellan dessa objekt?

f) Följ ändringar i objektet `iterator` under en iteration genom teckensekvensen. Vad händer om satsen `(3)` bortkommenteras? Går det att traversera teckensekvensen baklänges?

g) Tecken i en teckensekvens kan lagras i en teckenvektor eller i en nodsekvens. Vilken strategi är bättre?

Övning 8

```
// StringSequence.java

class StringSequence
{
```

```
private class Node
{
    public String    string;
    public Node      nextNode;

    public Node (String string)
    {
        this.string = string;
        this.nextNode = null;
    }
}


private Node    firstNode = null;
private Node    lastNode = null;

public void append (String string)
{
    Node    node = new Node (string);
    if (firstNode == null)
        firstNode = node;
    else
        lastNode.nextNode = node;
    lastNode = node;
}

public String toString ()
{
    String    s = "[";
    Node    node = firstNode;
    if (lastNode != null)
    {
        while (node != lastNode)
        {
            s = s + node.string + ", ";
            node = node.nextNode;
        }
        s = s + lastNode.string;
    }
    s = s + "]";

    return s;
}

public int stringCount ()
{
    int     count = 0;
    Node    node = firstNode;
    while (node != null)
    {
        count++;
```

```
            node = node.nextNode;
        }

        return count;
    }

    public void insert (String string, int pos)
                      throws ArrayIndexOutOfBoundsException
    {
        int     count = this.stringCount ();

        if (pos < 0  ||  pos > count)
             throw new ArrayIndexOutOfBoundsException (
                                        "wrong position: " + pos);

        Node    node = new Node (string);
        if (firstNode == null)
        {
            firstNode = node;
            lastNode = node;
        }
        else if (pos == 0)
        {
            node.nextNode = firstNode;
            firstNode = node;
        }
        else if (pos == count)
        {
            lastNode.nextNode = node;
            lastNode = node;
        }
        else
        {
            Node    p = firstNode;
            for (int i = 0; i < pos - 1; i++)
                p = p.nextNode;
            node.nextNode = p.nextNode;
            p.nextNode = node;
        }
    }
}

// UseStringSequence.java

class UseStringSequence
{
    public static void main (String[] args)
    {
        StringSequence    seq = new StringSequence ();
        System.out.println (seq);
        System.out.println ();
```

```
        // seq.insert ("z", 0);                     // (1)
        String[]    s = {"a", "b", "c", "d", "e"};
        for (int i = 0; i < s.length; i++)
            seq.append (s[i]);
        System.out.println (seq);

        int    pos = 0;
        // pos = 3;                                 // (*)
        // pos = 5;                                 // (*)
        // pos = 6;                                 // (*)
        seq.insert ("1", pos);
        System.out.println (seq);
    }
}
```

a) Vad händer när programmet `UseStringSequence` exekveras?

b) Vad händer om satsen `(1)`, eller någon av satserna `(*)`, inkluderas?

c) Rita objektet (som refereras av referensen) `seq` vid två olika tillfällen: före och efter anropen till metoden `append`.

d) Åskådliggör den algoritm som används i metoden `append`. Varför skiljer man två olika fall i metoden?

e) Åskådliggör den algoritm som används i metoden `toString`. Varför kontrollerar man huruvida `lastNode` är skild från `null`?

f) Åskådliggör den algoritm som används i metoden `insert`. Varför skiljer man fyra olika fall i metoden?

Övning 9

```
// StringSequence.java

class StringSequence
{
    private String[]    strings;
    private int    stringCount;

    public StringSequence (int capacity)
    {
        strings = new String[capacity];
        stringCount = 0;
    }

    public void append (String string)
                        throws IllegalStateException
```

```
    {
        if (stringCount == strings.length)
            throw new IllegalStateException ("sequence already full");

        strings[stringCount++] = string;
    }

    public String toString ()
    {
        String    s = "[";
        if (stringCount > 0)
        {
            for (int pos = 0; pos < stringCount - 1; pos++)
                s = s + strings[pos] + ", ";

            s = s + strings[stringCount - 1];
        }
        s = s + "]";

        return s;
    }

    public void insert (String string, int pos)
                      throws ArrayIndexOutOfBoundsException
    {
        if (pos < 0  ||  pos > stringCount  ||  pos >= strings.length)
            throw new ArrayIndexOutOfBoundsException (
                                            "wrong position: " + pos);

        for (int p = stringCount - 1; p >= pos; p--)
            strings[p + 1] = strings[p];
        strings[pos] = string;
        stringCount++;
    }
}

// UseStringSequence.java

class UseStringSequence
{
    public static void main (String[] args)
    {
        int    cap = 10;
        // cap = 4;                        // (1)
        StringSequence    seq = new StringSequence (cap);
        System.out.println (seq);
        System.out.println ();

        // seq.insert ("z", 0);           // (2)
        String[]    s = {"a", "b", "c", "d", "e"};
        for (int i = 0; i < s.length; i++)
```

```
            seq.append (s[i]);
        System.out.println (seq);

        int    pos = 0;
        // pos = 3;                          // (*)
        // pos = 5;                          // (*)
        // pos = 6;                          // (*)
        seq.insert ("1", pos);
        System.out.println (seq);
    }
}
```

a) Vad händer när programmet `UseStringSequence` exekveras?

b) Vad händer om satsen `(1)`, satsen `(2)`, eller någon av satserna `(*)`, inkluderas?

c) Rita objektet (som refereras av referensen) `seq` vid två olika tillfällen: före och efter anropen till metoden `append`.

d) Åskådliggör den algoritm som används i metoden `append`. När kastas ett undantag i metoden?

e) Åskådliggör den algoritm som används i metoden `toString`. Varför kontrollerar man huruvida `stringCount` är positiv?

f) Åskådliggör den algoritm som används i metoden `insert`. Bestäm algoritmens tidskomplexitet. När kastas ett undantag i metoden?

g) Man kan lagra ett antal strängar i en vektor, men även i en sekvens av länkade noder. På vilket sätt skiljer sig dessa strategier?

Övning 10

```
// CharSequence.java

class CharSequence
{
    private class Node
    {
        public char    c;
        public Node    nextNode;
        public Node    previousNode;

        public Node (char c)
        {
            this.c = c;
            this.nextNode = null;
            this.previousNode = null;
        }
```

```
}


private Node    firstNode = null;
private Node    lastNode = null;
private int     charCount = 0;

public void append (char c)
{
    Node    node = new Node (c);
    if (firstNode == null)
        firstNode = node;
    else
    {
        lastNode.nextNode = node;
        node.previousNode = lastNode;
    }
    lastNode = node;

    charCount++;
}

public String toString ()
{
    String    s = "";
    Node    node = firstNode;
    while (node != null)
    {
        s = s + node.c;
        node = node.nextNode;
    }

    return s;
}

public void insert (char c, int pos)
               throws ArrayIndexOutOfBoundsException
{
    if (pos < 0  ||  pos > charCount)
        throw new ArrayIndexOutOfBoundsException (
                                  "wrong position: " + pos);

    Node    node = new Node (c);
    if (firstNode == null)
    {
        firstNode = node;
        lastNode = node;
    }
    else if (pos == 0)
    {
        node.nextNode = firstNode;
```

```
            firstNode.previousNode = node;
            firstNode = node;
        }
        else if (pos == charCount)
        {
            lastNode.nextNode = node;
            node.previousNode = lastNode;
            lastNode = node;
        }
        else
        {
            Node    p = firstNode;
            for (int i = 0; i < pos - 1; i++)
                p = p.nextNode;
            node.nextNode = p.nextNode;

            node.previousNode = p;
            p.nextNode.previousNode = node;
            p.nextNode = node;
        }

        charCount++;
    }


    public class CharSequenceIterator
    {
        private Node    pos;

        public CharSequenceIterator ()
        {
            pos = (firstNode != null)? firstNode : null;
        }

        public boolean hasChar ()
        {
            return pos != null;
        }

        public char currentChar ()
        {
            return pos.c;
        }

        public void moveForward ()
        {
            if (pos != null)
                pos = pos.nextNode;
            else
                pos = firstNode;
        }
```

```
        public void moveBackward ()
        {
            if (pos != null)
                pos = pos.previousNode;
            else
                pos = lastNode;
        }
    }
}

// UseCharSequence.java

class UseCharSequence
{
    public static void main (String[] args)
    {
        char[]    characters = {'a', 'b', 'c', 'd', 'e'};
        CharSequence    seq = new CharSequence ();
        for (int i = 0; i < characters.length; i++)
            seq.append (characters[i]);
        System.out.println (seq);
        System.out.println ();

        CharSequence.CharSequenceIterator    iterator =
                            seq.new CharSequenceIterator ();
        char    c = 0;
        while (iterator.hasChar ())
        {
            c = iterator.currentChar ();
            System.out.print (c + " ");

            iterator.moveForward ();
        }
        System.out.println ();

        iterator.moveBackward ();
        while (iterator.hasChar ())
        {
            c = iterator.currentChar ();
            System.out.print (c + " ");

            iterator.moveBackward ();
        }
        System.out.println ();

        iterator.moveForward ();
        while (iterator.hasChar ())
        {
            c = iterator.currentChar ();
            System.out.print (c + " ");
```

```
            iterator.moveForward ();
        }
        System.out.println ();
        System.out.println ();

        int    pos = 2;
        seq.insert ('1', pos);
        System.out.println (seq);
    }
}
```

a) Vad händer när programmet `UseCharSequence` exekveras?

b) Åskådliggör den algoritm som används i metoden `append`.

c) Hur ser objektet `seq` ut efter anropen till metoden `append`?

d) Hur modifieras objektet `iterator` under rn resa genom teckensekvensen fram och tillbaka?

e) Åskådliggör den algoritm som används i metoden `insert`.

Övning 11

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.

Övning 12

Klassen `List` representerar en heltalslista:

```
class List
{
    private static class Node
    {
        public int     value;
        public Node    next;

        public Node (int value)
        {
            this.value = value;
            this.next = null;
        }
    }

    private Node    first = null;

    // add lägger till ett givet heltal till listan
    public void add (int value)
    {
        Node    node = new Node (value);
```

```
        if (first == null)
            first = node;
        else
        {
            Node    n = first;
            while (n.next != null)
                n = n.next;
            n.next = node;
        }
    }

    // ytterligare metoder
}
```

a) Skapa en tom lista av typen `List`, och rita den.

b) Skapa en lista av typen `List` som innehåller följande heltal: 1, 4, 5 och 7. Rita den listan.

Övning 13

En klass, `Color`, representerar en färg:

```
class Color
{
    // vit färg
    public static final Color    WHITE = new Color (255, 255, 255);
    // svart färg
    public static final Color    BLACK = new Color (0, 0, 0);

    // färgens komponenter
    private int    red;
    private int    green;
    private int    blue;

    // Color initierar färgens komponenter utifrån givna värden.
    // En komponent har ett värde mellan 0 (inklusive) och 255
    // (inklusive).
    public Color (int red, int green, int blue)
    {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }

    public String toString ()
    {
```

```
        return "[" + this.red + ", " + this.green + ", " + this.blue
               + "]";
    }

    public Color combineWith (Color c)
    {
        return new Color (this.red, c.green,
                          (this.blue + c.blue) % 256);
    }
}
```

a) Vilken utskrift skapas när följande kodavsnitt exekveras?

```
Color    color1 = new Color (100, 50, 225);
Color    color2 = new Color (0, 40, 35);
Color    color = color1.combineWith (color2);
System.out.println (color);
```

b) Anropa metoden `combineWith` i samband med de två fördefinierade färgerna: `BLACK` och `WHITE`.

c) Skapa en ny konstruktor i stället för den som redan finns. Den nya konstruktorn ska kasta ett undantag av typen `java.lang.IllegalArgumentException` i fall att något eller några argument hamnar utanför intervallet `[0, 255]`.

Övning 14

En klass, `Int`, representerar ett heltal:

```
class Int
{
    private int    n;

    public Int (int n)
    {
        this.n = n;
    }

    public String toString ()
    {
        return "[" + this.n + "]";
    }

    public Int add (Int i)
    {
        return new Int (this.n + i.n);
    }
}
```

En klass, `IntCollection`, representerar en samling heltal:

```
class IntCollection
{
    private int      size;
    private Int[]    ints;
    private int      addPos;

    public IntCollection (int size)
    {
        this.size = size;
        this.ints = new Int[size];
        addPos = 0;
    }

    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("{");
        for (int j = 0; j < addPos; j++)
            sb.append (ints[j].toString ());
        sb.append ("}");

        return sb.toString ();
    }

    public void add (Int i)
    {
        ints[addPos++] = i;
    }

    public Int sum ()
    {
        Int    s = new Int (0);
        for (int j = 0; j < addPos; j++)
            s = s.add (ints[j]);

        return s;
    }
}
```

a) Vilken utskrift skapas när följande kodavsnitt exekveras:

```
IntCollection    ic = new IntCollection (4);
for (int j = 1; j < 5; j++)
    ic.add (new Int (j));
System.out.println (ic);
```

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

c) Vilken utskrift skapas när följande kodavsnitt exekveras:

```
IntCollection    ic = new IntCollection (4);
for (int j = 1; j < 5; j++)
    ic.add (new Int (j));
Int    s = ic.sum ();
System.out.println (s);
```

d) Vad händer när följande kodavsnitt exekveras:

```
IntCollection    ic = new IntCollection (4);
for (int j = 1; j < 5; j++)
    ic.add (new Int (j));
Int    i = new Int (10);
ic.add (i);
```

Övning 15

Man använder klasserna `Worker` och `Supporter`:

```
Worker    worker = new Worker (10);
Worker.Supporter    supporter = worker.new Supporter (0.5);
worker.work (8);
supporter.support ("actively");
```

När det här kodavsnittet utförs, erhålls följande utskrift:

```
working 10 x 8 hours
supporting 0.5 x 10 days actively
```

En annan utskrift erhålls när följande kodavsnitt utförs.

```
Worker    worker = new Worker (60);
Worker.Supporter    supporter = worker.new Supporter (0.75);
worker.work (7);
supporter.support ("passively");
```

I det här fallet blir utskriften:

```
working 60 x 7 hours
supporting 0.75 x 60 days passively
```

Skapa klasserna `Worker` och `Supporter`.

Övning 16

Klassen `IntegerQueue` representerar en kö för heltal:

```
class IntegerQueue
{
    private Integer[]    elements;
```

```
    private int    firstIndex = 0;
    private int    lastIndex = -1;

    public IntegerQueue (int capacity)
    {
        elements = new Integer[capacity];
    }

    public boolean isEmpty ()
    {
        return elements[firstIndex] == null;
    }

    public void put (Integer element)
    {
        lastIndex = lastIndex + 1;
        elements[lastIndex] = element;
    }

    public Integer take ()
    {
        Integer    firstElement = elements[firstIndex];
        int    index = firstIndex + 1;
        while (index < elements.length  &&  elements[index] != null)
        {
            elements[index - 1] = elements[index];
            index++;
        }
        elements[lastIndex] = null;
        lastIndex--;

        return firstElement;
    }
}
```

En heltalskö skapas och används så här:

```
IntegerQueue    queue = new IntegerQueue (4);
for (int i = 1; i <= 4; i++)
    queue.put (i);

Integer    n1 = queue.take ();
Integer    n2 = queue.take ();
queue.put (5);                      // (1)

Integer    n = null;
while (!queue.isEmpty ())
{
    n = queue.take ();
```

```
    System.out.print (n + " ");
}
```

a) Vilken utskrift skapas när det givna kodavsnittet utförs?

b) Hur ser ut kön när satsen `(1)` har utförts – rita den.

c) Vilken tidskomplexitet har metoden `take`?

Övning 17

Klassen `IntegerQueue` representerar en kö för heltal:

```
class IntegerQueue
{
    private Integer[]    elements;

    private int    firstIndex = 0;
    private int    lastIndex = -1;

    public IntegerQueue (int capacity)
    {
        elements = new Integer[capacity];
    }

    public boolean isEmpty ()
    {
        return elements[firstIndex] == null;
    }

    public void put (Integer element)
    {
        lastIndex = (lastIndex + 1) % elements.length;
        elements[lastIndex] = element;
    }

    public Integer take ()
    {
        Integer    firstElement = elements[firstIndex];
        elements[firstIndex] = null;
        firstIndex = (firstIndex + 1) % elements.length;

        return firstElement;
    }
}
```

En heltalskö skapas och används så här:

```
IntegerQueue    queue = new IntegerQueue (4);
```

```
for (int i = 1; i <= 4; i++)
    queue.put (i);

Integer    n1 = queue.take ();
Integer    n2 = queue.take ();
queue.put (5);                        // (1)

Integer    n = null;
while (!queue.isEmpty ())
{
    n = queue.take ();
    System.out.print (n + " ");
}
```

a) Vilken utskrift skapas när det givna kodavsnittet utförs?

b) Hur ser ut kön när satsen `(1)` har utförts – rita den.

c) Vilken tidskomplexitet har metoden `take`?

Övning 18

Klassen `IntegerQueue` representerar en kö för heltal:

```
class IntegerQueue
{
    private Integer[]    elements;
    private int    firstIndex = 0;
    private int    lastIndex = -1;

    // Priconditions: initialCapacity > 0
    public IntegerQueue (int initialCapacity)
    {
        elements = new Integer[initialCapacity];
    }

    public boolean isEmpty ()
    {
        return elements[firstIndex] == null;
    }

    public int size ()
    {
        int    countElements = 0;
        if (lastIndex != -1)
            countElements = (firstIndex <= lastIndex)?
                        lastIndex - firstIndex + 1 :
                        lastIndex + 1 + elements.length - firstIndex;
```

```
        return countElements;
    }

    private void increaseCapacity ()
    {
        Integer[]    e = new Integer[3 * elements.length / 2];
        int    index = firstIndex;
        for (int eIndex = 0; eIndex < elements.length; eIndex++)
        {
            e[eIndex] = elements[index];
            index = (index + 1) % elements.length;
        }
        firstIndex = 0;
        lastIndex = elements.length - 1;

           elements = e;
    }

    public void put (Integer element)
    {
        if (this.size () == elements.length)
            this.increaseCapacity ();

        lastIndex = (lastIndex + 1) % elements.length;
        elements[lastIndex] = element;
    }

    public Integer take ()
    {
        Integer    firstElement = elements[firstIndex];
        elements[firstIndex] = null;
        firstIndex = (firstIndex + 1) % elements.length;

        return firstElement;
    }
}
```

En heltalskö skapas och används så här:

```
IntegerQueue    queue = new IntegerQueue (4);
for (int i = 1; i <= 4; i++)
    queue.put (i);

Integer    n1 = queue.take ();
Integer    n2 = queue.take ();
queue.put (5);
queue.put (6);  // (1)
queue.put (7);  // (2)
```

a) Hur ser ut kön när satsen `(1)` har utförts – rita den.

b) Hur ser ut kön när satsen `(2)` har utförts – rita den.

Övning 19

Klassen `Numbers` representerar en heltalslista:

```
class Numbers
{
    // en nod som lagrar ett heltal
    private static class Node
    {
        public int     number;
        public Node    next;

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

    private Node    first = null;

    // add lägger till ett givet heltal till listan
    public void add (int number)
    {
        Node    node = new Node (number);

        if (first == null)
            first = node;
        else
        {
            Node    n = first;
            while (n.next != null)
                n = n.next;
            n.next = node;
            }
        }


    // en iterator till listan
    public class Iterator
    {
        // koden här
    }
}
```

En heltalslista skapas och itereras så här:

```
Numbers    numbers = new Numbers ();
int[]    num = {1, 16, 25, 49};
for (int i = 0; i < num.length; i++)
    numbers.add (num[i]);

Numbers.Iterator    iterator = numbers.new Iterator ();  // (1)
int    number = 0;
while (iterator.hasNumber ())
{
    number = iterator.number ();
    System.out.print (Math.sqrt (number) + "  ");
    iterator.move ();
}

// number = iterator.number ();  // (2)
```

När detta kodavsnitt utförs, skapas följande utskrift:

```
1.0  4.0  5.0  7.0
```

Om satsen `(2)` inkluderas i koden, kastas ett undantag av typen `java.util.NoSuchElementException`.

a) Skapa den inre klassen `Iterator`.

b) Hur ser ut listan och iteratorn när satsen `(1)` har utförts? Rita både listan och iteratorn.

Hur ser ut listan och iteratorn när två varv i `while`-loopen har utförts? Rita dem även i så fall.

Problem 1

En modell av en polylinje – polylinjens hörn ska lagras i en vektor

A) En modell av en punkt i planet

En punkt i planet har sitt namn och sina koordinater. Punktens koordinater är hela tal.

En punkt kan bestämmas med ett namn och två heltalskoordinater. Det går även att bestämma en punkt utifrån en annan punkt – en kopia kan skapas.

Man kan skapa en teckensträng som representerar en punkt. Denna teckensträng kan vara på formen `(A 3 4)`. En punkts namn och koordinater kan erhållas. Koordinaterna kan ändras, en koordinat i taget. Avståndet mellan två givna punkter kan bestämmas. Man kan kontrollera huruvida två givna punkter är likadana eller inte.

En modell av en punkt i planet ska skapas: man ska skapa en definitionsklass som heter `Punkt`.

Ett enkelt testprogram för klassen Punkt

```
import java.io.*;    // PrintWriter

class PunktTest
{
    public static void main (String[] args)
    {
        PrintWriter    out = new PrintWriter (System.out, true);

        // testa en konstruktor och en transformator
        Punkt    p1 = new Punkt ("A", 3, 4);
        Punkt    p2 = new Punkt ("B", 5, 6);
        out.println (p1 + "    " + p2);

        // testa inspektorer
        String    n = p1.getNamn ();
        int    x = p1.getX ();
        int    y = p1.getY ();
        out.println (n + " " + x + " " + y);

        // testa en kombinator och en komparator
        double    d = p1.avstand (p2);
        out.println (d);
        boolean    b = p1.equals (p2);
        out.println (b);

        // testa mutatorer
        p2.setX (1);
        p2.setY (2);
        out.println (p2);

        // testa en konstruktor till
        Punkt    p = new Punkt (p1);
        out.println (p);
    }
}
```

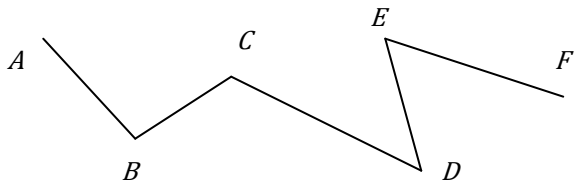
Uppgifter i samband med punkter i planet

1. Skapa en definitionsklass som heter `Punkt`, som representerar en punkt i planet. Kör testprogrammet `PunktTest` i samband med den nyskapade klassen.

2. Skapa en bild som visar hur avståndet mellan två punkter i planet bestäms. Relatera denna bild till koden i den motsvarande metoden: för in variablernas namn i bilden.

B) En polylinje i planet

En polylinje är en geometrisk figur som består av en serie bundna linjesegment. Ändpunkter för dessa segment utgör polylinjens hörn. En polylinje kan föreställas så här:



En polylinje kan bestämmas med sina hörn, sin färg och sin bredd. En tom polylinje saknar hörn.

Man kan skapa en teckensträng som representerar en polylinje. Denna teckensträng kan vara på formen `{[(A 3 4)(B 1 2)(C 2 3)(D 5 1)], svart, 1}`. En polylinjes hörn, färg, bredd och längd kan erhållas. Färgen och bredden kan ändras. En polylinjes utseende ändras även när dess hörnsekvens ändras. Det går att lägga till ett nytt hörn till polylinjen – antingen på slutet, eller framför ett hörn vars namn är givet. Det går även att ta bort ett hörn med ett givet namn.

En modell av en polylinje i planet ska skapas: man ska skapa en definitionsklass som heter `Polylinje`.

En modell av en polylinje – ej fullständig

```
public class Polylinje
{
    private Punkt[]    horn;
    private String     farg = "svart";
    private int        bredd = 1;

    public Polylinje ()
    {
        this.horn = new Punkt[0];
    }

    public Polylinje (Punkt[] horn)
```

```
    {
        this.horn = new Punkt[horn.length];
        for (int i = 0; i < horn.length; i++)
            this.horn[i] = new Punkt (horn[i]);
    }

    public String toString () {}

    public Punkt[] getHorn () {}

    public String getFarg () {}

    public int getBredd () {}

    public void setFarg (String farg) {}

    public void setBredd (int bredd) {}

    public double langd () {}

    public void laggTill (Punkt horn)
    {
        Punkt[]    h = new Punkt[this.horn.length + 1];
        int    i = 0;
        for (i = 0; i < this.horn.length; i++)
            h[i] = this.horn[i];
        h[i] = new Punkt (horn);

        this.horn = h;
    }

    public void laggTillFramfor (Punkt horn, String hornNamn) {}

    public void taBort (String hornNamn) {}
}
```

Uppgifter i samband med polylinjer

1. Komplettera definitionsklassen `Polylinje`: implementera metoderna och kommentera klassen samt dess medlemmar. I metoden `laggTillFramfor` ska det hörn som parameterreferensen refererar till kopieras. I metoden `getHorn` ska en vektor som innehåller kopior av polylinjens hörn skapas, och en referens till denna vektor ska returneras.

2. Skapa en definitionsklass till: `Polylinje1`. I denna klass ska parameterreferenser kopieras, i stället för de refererade resurserna. I metoden `getHorn` ska referensen till egna hörn returneras.

Vilken strategi är bättre: att kopiera resurser (som i klassen `Polylinje`) eller bara referenser (som i klassen `Polylinje1`)?

3. Skapa ett enkelt testprogram, `PolylinjeTest`, som använder konstruktorerna och metoderna i klassen `Polylinje`.

4. Skapa en bild som representerar ett objekt av typen `Polylinje`. Bland annat ska bilden innehålla objektet och dess referenser, den refererade vektorn och dess referenser, samt de hörn som refereras ifrån vektorn. Bilden ska förses med rätta beteckningar.

5. Åskådliggör den algoritm som används i metoden `laggTillFramfor`. Man ska rita en serie bilder som visar hur det givna hörnet så småningom förs in på rätt ställe i den aktuella polylinjen. Bilderna ska förses med rätta beteckningar.

6 Ett objekt av typen `Polylinje` har sin egen vektor, där polylinjens hörn lagras. Är denna strategi minneseffektiv? Är den tidseffektiv? Vad händer om man ofta lägger till eller tar bort hörn?

C) Skapa och använda polylinjer

I ett program, som heter `ValjPolylinje`, skapar man och använder polylinjer av typen `Polylinje`.

Ett antal slumpmässiga polylinjer skapas och visas. En polylinje har ett slumpmässigt antal hörn, och färgen är antingen blå, röd eller gul. Namn på polylinjens hörn är stora bokstäver i engelska alfabetet. Namnen bestäms slumpmässigt, men två hörn kan inte ha likadana namn. Programmet bestämmer och visar den kortaste av de gula polylinjerna.

Programmet `ValjPolylinje` har följande struktur:

```
class ValjPolylinje
{
    public static final Random    rand = new Random ();
    public static final int    ANTAL_POLYLINJER = 10;

    public static void main (String[] args)
    {
        // skapa ett antal slumpmässiga polylinjer
        Polylinje[]    polylinjer = new Polylinje[ANTAL_POLYLINJER];
        for (int i = 0; i < ANTAL_POLYLINJER; i++)
            polylinjer[i] = slumpPolylinje ();

        // visa polylinjerna

        // bestäm den kortaste av de polylinjer som är gula

        // visa den valda polylinjen
```

```
    }

    // slumpPunkt returnerar en punkt med ett slumpmässigt namn,
    // som är en stor bokstav i det engelska alfabetet, och
    // slumpmässiga koordinater.
    public static Punkt slumpPunkt ()
    {
        String    n = "" + (char) (65 + rand.nextInt (26));
        int    x = rand.nextInt (11);
        int    y = rand.nextInt (11);

        return new Punkt (n, x, y);
    }

    // slumpPolylinje returnerar en slumpmässig polylinje, vars färg
    //  är antingen blå, eller röd eller gul. Namn på polylinjens
    // hörn är stora bokstäver i det engelska alfabetet. Två hörn
    // kan inte ha samma namn.
    public static Polylinje slumpPolylinje ()
    {
        // skapa en tom polylinje, och lägg till hörn till den
        Polylinje    polylinje = new Polylinje ();
        int    antalHorn = 2 + rand.nextInt (7);
        int    antalValdaHorn = 0;
        boolean[]    valdaNamn = new boolean[26];
        // ett och samma namn kan inte förekomma flera gånger
        Punkt    valdPunkt = null;
        char    valtChar = 0;
        while (antalValdaHorn < antalHorn)
        {

        }


        // sätt färg
    }
}
```

Uppgift i samband med användning av polylinjer

Komplettera och prova programmet `ValjPolylinje`. Vad händer om ingen av polylinjerna är gul?

D) En iterator till en polylinje

En iterator till en polylinje kan definieras och implementeras. Med en sådan iterator kan hörnen i polylinjen besökas och användas i tur och ordning.

En iterator till en polylinje kan definieras och implementeras i en inre klass i klassen `Polylinje`. Denna klass kan heta `PolylinjeIterator`, och kan se ut så här:

```
public class PolylinjeIterator
{
    private int    aktuell = -1;

    public PolylinjeIterator ()
    {
        if (Polylinje.this.horn.length > 0)
            aktuell = 0;
    }

    public boolean finnsHorn ()
    {
        return aktuell != -1;
    }

    public Punkt horn () throws java.util.NoSuchElementException
    {
        if (!this.finnsHorn ())
            throw new java.util.NoSuchElementException (
                                    "slut av iterationen");

        Punkt    horn = Polylinje.this.horn[aktuell];

        return horn;
    }

    public void gaFram ()
    {
        if (aktuell >= 0  &&
            aktuell < Polylinje.this.horn.length - 1)
            aktuell++;
        else
            aktuell = -1;
    }
}
```

Uppgifter i samband med iteratorer

1. Skapa en iterator (ett objekt av typen `PolylinjeIterator`) i testprogrammet för klassen `Polylinje`. Besök med denna iterator polylinjens hörn i tur och ordning, och skriv ut dem.

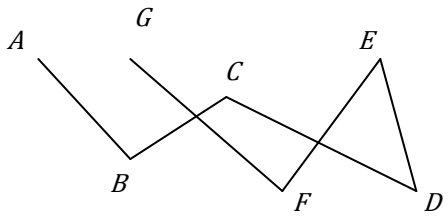
2- Visualisera en iteration genom en polylinje. Följ ändringarna i iterator-objektet.

Problem 2

En modell av en polylinje – polylinjens hörn ska lagras i länkade noder

A) En polylinje i planet

En polylinje är en geometrisk figur som består av en serie bundna linjesegment. Ändpunkter för dessa segment utgör polylinjens hörn. En polylinje kan föreställas så här:



En polylinje kan bestämmas med sina hörn, sin färg och sin bredd. En tom polylinje saknar hörn.

Man kan skapa en teckensträng som representerar en polylinje. Denna teckensträng kan vara på formen `{[(A 3 4)(B 1 2)(C 2 3)(D 5 1)], svart, 1}`. En polylinjes hörn, färg, bredd och längd kan erhållas. Färgen och bredden kan ändras. En polylinjes utseende ändras även när dess hörnsekvens ändras. Det går att lägga till ett nytt hörn till polylinjen – antingen på slutet, eller framför ett hörn vars namn är givet. Det går även att ta bort ett hörn med ett givet namn.

En modell av en polylinje i planet ska skapas: man ska skapa en definitionsklass som heter `Polylinje`.

En modell av en polylinje – ej fullständig

Man utgår ifrån att det finns en klass `Punkt`, som på ett passande sätt representerar en punkt i planet.

```
public class Polylinje
{
    private static class Nod
    {
        public Punkt    horn;
        public Nod      nastaNod;

        public Nod (Punkt horn)
        {
            this.horn = horn;
            nastaNod = null;
        }
    }

    private Nod       forstaNod;
    private String    farg = "svart";
    private int       bredd = 1;  // pixlar

    public Polylinje ()
    {
        this.forstaNod = null;
    }

    public Polylinje (Punkt[] horn)
    {
        if (horn.length > 0)
        {
            Nod   nod = new Nod (new Punkt (horn[0]));
            this.forstaNod = nod;
            int    pos = 1;
            while (pos < horn.length)
            {
                nod.nastaNod = new Nod (new Punkt (horn[pos++]));
                nod = nod.nastaNod;
            }
        }
    }

    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("{[");
        Nod    nod = this.forstaNod;
        while (nod != null)
        {
            sb.append (nod.horn);
            nod = nod.nastaNod;
```

```
    }
    sb.append ("], " + farg + ", " + bredd + "}");

    return sb.toString ();
}

public Punkt[] getHorn () {}

public String getFarg () {}

public int getBredd () {}

public void setFarg (String farg) {}

public void setBredd (int bredd) {}

public double langd () {}

public void laggTill (Punkt horn) {}

// laggTillFramfor lägger till ett givet hörn till
// polylinjen. Hörnet hamnar framför det hörn vars namn
// anges.
public void laggTillFramfor (Punkt horn, String hornNamn)
{
    // bestäm position för det hörn som har det givna namnet
    Nod    posNod = this.forstaNod;
    Nod    foregaendeNod = null;
    while (posNod != null  &&
           !posNod.horn.getNamn ().equals (hornNamn))
    {
        foregaendeNod = posNod;
        posNod = posNod.nastaNod;
    }

    // om hörnet med det givna namnet inte finns
    if (posNod == null)
        return;

    // lägg till det givna hörnet på rätt plats
    Nod    nod = new Nod (new Punkt (horn));
    if (foregaendeNod == null)
    {
        nod.nastaNod = this.forstaNod;
        this.forstaNod = nod;
    }
    else
    {
        nod.nastaNod = posNod;
        foregaendeNod.nastaNod = nod;
    }
```

```
    }

    public void taBort (String hornNamn) {}
}
```

Uppgifter i samband med polylinjer

1. Komplettera definitionsklassen `Polylinje`: implementera metoderna, och kommentera klassen och dess medlemmar. Implementera klassen på två olika sätt: antingen ska referenserna eller de utpekade resurserna kopieras.

2. Skapa ett enkelt testprogram, som använder konstruktorerna och metoderna i klassen `Polylinje`. Kör testprogrammet i samband med båda mplementationerna.

3. Skapa en bild som representerar ett objekt av typen `Polylinje`.

4. Åskådliggör den algoritm som används i metoden `laggTillFramfor`.

5. Ett objekt av typen `Polylinje` har sina egna noder, där polylinjens hörn lagras. Är denna strategi minneseffektiv? Är den tidseffektiv?

B) En iterator till en polylinje

En iterator till en polylinje kan definieras och implementeras. Med en sådan iterator kan hörnen i polylinjen besökas och användas i tur och ordning.

En iterator till polylinjen kan definieras och implementeras i en inre klass i klassen `Polylinje`. Denna klass kan heta `PolylinjeIterator`, och kan se ut så här (behöver kompletteras):

```
public class PolylinjeIterator
{
    private Nod    aktuell = null;

    public PolylinjeIterator ()
    {
        aktuell = Polylinje.this.forstaNod;
    }

    public boolean finnsHorn () {}

    public Punkt horn ()
           throws java.util.NoSuchElementException
    {}

    public void gaFram ()
    {
```

```
        if (aktuell != null)
            aktuell = aktuell.nastaNod;
    }
}
```

Uppgifter i samband med iteratorer

1. Komplettera klassen `PolylinjeIterator`.

2. Skapa en itertaor (ett objekt av typen `PolylinjeIterator`) i testprogrammet för klassen `Polylinje`. Besök med denna iterator polylinjens hörn i tur och ordning, och skriv ut dem. Gör så här:

```
Polylinje.PolylinjeIterator    iterator =
                          polylinje.new PolylinjeIterator ();
while (iterator.finnsHorn ())
{
    Punkt    aktuellHorn = iterator.horn ();
    System.out.print (aktuellHorn + " ");

    iterator.gaFram ();
}
```

3- Visualisera en iteration genom en polylinje. Följ ändringarna i iterator-objektet.

Kapitel 14

Utveckla nya objekttyper

Övning 1

Klassen `Person` hanterar olika uppgifter om en person. Klassen hanterar en persons efternamn, förnamn, personnummer, och andra relevanta uppgifter.

Applikationen `Persons` hanterar ett antal personer. I denna applikation matar man in uppgifter om ett antal personer, och bearbetar dessa uppgifter på olika sätt. Vid inmatningen kontrolleras att inte uppgifter om en och samma person matas in flera gånger. Efter inmatningen visas dessa personer. Personerna visas på så sätt att deras personnummer hamnar under varandra. Man får en utskrift av den här formen:

```
Anna Björn        19890813-4297
Sanna Storskog    19880430-3597
```

Till sist beräknas medelåldern för dessa personer, och resultatet visas.

a) Skapa klassen `Person`.

b) Skapa ett testprogram `PersonTest`, som testar klassen `Person`. Skapa ett dokument med olika testdata och motsvarande resultat.

c) Skapa ett dokument `PersonDescription`, som beskriver klassen `Person` och dess tjänster.

d) Skapa applikationen `Persons`.

Övning 2

a) Studera noggrant dokumentationen för klassen `java.lang.Integer` i Javas standardbibliotek. Fundera över denna klass och dess tjänster. Är det någonting som kan göras bättre? Är det någonting som är överflödig? Är det någonting som saknas?

b) Skapa en egen klass som fungerar ungefär som klassen `java.lang.Integer`. Testa klassen, och använd den i någon applikation. Hur kan man förbättra den klassen?

c) Skapa ett dokument som beskriver klassen som skapats, och de tjänster som den tillhandahåller.

Övning 3

a) Studera noggrant dokumentationen för klassen `java.lang.StringBuilder` i Javas standardbibliotek. Fundera över denna klass och dess tjänster. Är det någonting som kan göras bättre? Är det någonting som är överflödig? Är det någonting som saknas?

b) Skapa en egen klass som fungerar ungefär som klassen `java.lang.String Builder`. Testa klassen, och använd den i någon applikation. Kan klassen förbättras på något sätt?

c) Skapa ett dokument som beskriver klassen som skapats, och de tjänster som den tillhandahåller.

Övning 4

Klassen `IntList` hanterar en lista med objekt av typen `java.lang.Integer`.

a) Skapa klassen `IntList` och motsvarande testprogram.

b) Skapa en applikation som använder klassen `IntList`.

c) Skapa ett dokument som beskriver klassen `IntList` och dess tjänster.

Problem 1

Naturliga bråk

Utveckla en modell av ett naturligt bråk

Ett bråk är bestämt med sin täljare och sin nämnare. Om både täljaren och nämnaren är naturliga heltal, kan även bråket betraktas som naturligt. Exempel på sådana naturliga bråk är: `1/5`, `2/3`, `3/2`, `9/10`, `24/8`, och så vidare. Om täljaren är mindre än nämnaren sägs det att bråket är äkta – `2/3` är ett äkta bråk, men `3/2` är det inte.

Man kan utföra olika operationer med naturliga bråk. Ett bråk kan förkortas och förlängas. Vid förkortningen kan det största gemensamma delare för täljaren och nämnaren behövas. Det går att omvandla ett bråk till ett reellt tal. Ett bråk kan inverteras: genom invertering av `2/3` erhåller man `3/2`. Två bråk kan adderas, subtraheras, multipliceras och divideras. Det går att jämföra två bråk: ett bråk kan vara mindre, likadant som eller större än ett annat bråk. Vid additionen, subtraktionen och jämförelser av två bråk behöver en gemensam

nämnare, möjligen minsta gemensamma nämnare, bestämmas. Man kan kontrollera om ett bråk är äkta eller inte.

En modell av ett naturligt bråk ska utvecklas: man ska skapa en definitionsklass som heter `NBrak`, testa den och beskriva den.

Uppgifter i samband med naturliga bråk

1. Skapa en definitionsklass `NBrak` som representerar ett naturligt bråk.

2. Skapa ett utförligt, välorganiserat testprogram för klassen `NBrak`.

3. Skapa ett välorganiserat webbdokument om klassen och dess medlemmar.

Kapitel 15
Arv

Övning 1

Klassen `X` skapas, och sedan härleds klassen `Y` från klassen `X`:

```
class X
{
    protected int    m;
}

class Y extends X
{
    private int    n;

    public Y (int m, int n)
    {
        this.m = m;
        this.n = n;
    }

    public String toString ()
    {
        return m + ", " + n;
    }
}
```

Klassen `Y` används på följande vis:

```
class UseY
{
    public static void main (String[] args)
    {
        Y    y = new Y (3, 4);
        System.out.println (y);
    }
}
```

a) Vilken utskrift skapas när programmet `UseY` exekveras? Vilka variabler finns i objektet `y`?

b) Vad händer om man deklarerar variabeln `m` i klassen `X` som `private`?

Övning 2

Klassen `X` skapas, och sedan härleds klasserna `Y`, `Z` och `U` från klassen `X`:

```
class X
{
    private int    m;

    public X (int m)
    {
        this.m = m;
    }

    public int getM ()
    {
        return m;
    }

    public String toString ()
    {
        return "[" + m + "]";
    }
}

class Y extends X
{
    private int    n;

    public Y (int m, int n)
    {
        super (m);
        this.n = n;
    }

    public String toString ()
    {
        return super.toString () + "[" + n + "]";
    }
}

class Z extends X
{
    private int    n;

    public Z (int m, int n)
    {
```

```
        super (m);
        this.n = n;
    }

    public String toString ()
    {
        return "[" + this.getM () + ", " + n + "]";
    }
}

class U extends X
{
    private int    n;

    public U (int m, int n)
    {
        super (m);
        this.n = n;
    }
}
```

Klasserna `X`, `Y`, `Z` och `U` används på följande vis:

```
class UseXYZU
{
    public static void main (String[] args)
    {
        X    x = new X (5);
        Y    y = new Y (3, 4);
        Z    z = new Z (1, 2);
        U    u = new U (7, 8);

        System.out.println (x);
        System.out.println (y);
        System.out.println (z);
        System.out.println (u);
    }
}
```

Vilken utskrift skapas när programmet `UseXYZU` exekveras?

Övning 3

Klasserna `X`, `Y` och `UseXY` skapas så här:

```
class X
{
    public void printA()
```

```
    {
        System.out.println ("A");
    }
}

class Y extends X
{
    public void printA (int n)
    {
        for (int i = 0; i < n; i++)
            System.out.print ("A");
        System.out.println ();
    }
}

class UseXY
{
    public static void main (String[] args)
    {
        X    x = new X ();
        Y    y = new Y ();
        x.printA ();
        y.printA ();
        System.out.println ();
        // x.printA (4);                    // (1)
        y.printA (5);
    }
}
```

a) Vilken utskrift skapas när programmet `UseXY` exekveras?

b) Vad händer när satsen `(1)` inkluderas i programmet?

Övning 4

En klass, `NaturalNumber`, representerar ett naturligt heltal:

```
class NaturalNumber
{
    protected int    value;

    public NaturalNumber (int value) throws IllegalArgumentException
    {
        if (value < 0)
            throw new IllegalArgumentException (
                                        "not natural number");
        this.value = value;
    }
```

```
    public String toString ()
    {
        return "[" + this.value + "]";
    }

    public NaturalNumber add (NaturalNumber nn)
    {
        return new NaturalNumber (this.value + nn.value);
    }
}
```

En klass, `NamedNaturalNumber`, representerar ett naturligt heltal med namn:

```
class NamedNaturalNumber extends NaturalNumber
{
    private String    name = "-";

    public NamedNaturalNumber (int value, String name)
    {
        super (value);
        this.name = name;
    }

    public String toString ()
    {
        return "[" + this.value + ", " + this.name + "]";
    }

    public String getName ()
    {
        return name;
    }
}
```

Klasserna `NatualNumber` och `NamedNaturalNumber` används i följande program:

```
class NaturalNumbers
{
    public static void main (String[] args)
    {
        NaturalNumber   n1 = new NaturalNumber (5);
        NaturalNumber   n2 = new NaturalNumber (7);
        NamedNaturalNumber   nn1 = new NamedNaturalNumber (4, "four");
        NamedNaturalNumber   nn2 = new NamedNaturalNumber (10, "ten");
        // NamedNaturalNumber   nn3 =
        //          new NamedNaturalNumber (-4, "minus four"); // (1)
        System.out.println (n1);
        System.out.println (nn1);
        System.out.println ("---------");

        NaturalNumber   s1 = n1.add (n2);
        NaturalNumber   s2 = nn1.add (nn2);
```

```
        System.out.println (s1);
        System.out.println (s2);
        System.out.println ("---------");
        // NamedNaturalNumber   s3 = nn1.add (nn2);            // (2)

        // String    name1 = n1.getName ();                    // (3)
        String    name2 = nn1.getName ();
        System.out.println (name2);
    }
}
```

a) Vilken utskrift skapas när programmet `NaturalNumbers` exekveras?

b) Vad händer när satsen `(1)` inkluderas i programmet?

c) Vad händer när satsen `(2)` inkluderas i programmet?

d) Vad händer när satsen `(3)` inkluderas i programmet?

Övning 5

Klassen `X` skapas, och sedan härleds klasserna `Y` och `Z` från klassen `X`:

```
class X
{
    public Number getNumber()
    {
        return new Double (1.0);
    }
}

class Y extends X
{
    public Integer getNumber()
    {
        return new Integer (2);
    }
}

class Z extends X
{
    public String getNumber()
    {
        return new String ("3");
    }
}
```

Vilken av dessa klasser är inte korrekt? Varför?

Övning 6

Klasserna `X` och `Y` skapas så här:

```
class X
{
    public void method1() throws java.io.IOException
    {
        throw new java.io.IOException ();
    }

    public void method2()
    {
        System.out.println ("2X");
    }

    public void method3()
    {
        System.out.println ("3X");
    }
}

class Y extends X
{
    public void method1()
    {
        System.out.println ("Y");
    }

    public void method2() throws java.lang.ArithmeticException
    {
        throw new java.lang.ArithmeticException ();
    }

    public void method3() throws java.io.IOException
    {
        throw new java.io.IOException ();
    }
}
```

Vilken av metoderna är inte korrekt? Varför?

Övning 7

Klasserna `X` och `Y` skapas så här:

```
class X
{
```

```
    public void method1()
    {
        System.out.println ("1X");
    }

    protected void method2()
    {
        System.out.println ("2X");
    }
}

class Y extends X
{
    private void method1()
    {
        System.out.println ("1Y");
    }

    public void method2()
    {
        System.out.println ("2Y");
    }
}
```

Vilken av metoderna är inte korrekt? Varför?

Övning 8

```
class AStringBuilder extends StringBuilder
{
    public AStringBuilder ()
    {
        super ();
    }

    public AStringBuilder (int length)
    {
        super (length);
    }

    public AStringBuilder (String s)
    {
        super (s);
    }

    public void append (char c, int count)
    {
        for (int i = 0; i < count; i++)
            this.append (c);
```

```
    }
}

class BStringBuilder
{
    private StringBuilder    sb = null;

    public BStringBuilder ()
    {
        sb = new StringBuilder ("");
    }

    public BStringBuilder (int length)
    {
        sb = new StringBuilder (length);
    }

    public BStringBuilder (String s)
    {
        sb = new StringBuilder (s);
    }

    public void append (char c, int count)
    {
        for (int i = 0; i < count; i++)
            sb.append (c);
    }

    public String toString ()
    {
        return sb.toString ();
    }
}
```

Vilken av dessa klasser är inte korrekt? Varför?

Övning 9

Klasserna `X` och `Y` definieras så här:

```
class X
{
    public int getU ()
    {
        return 0;
    }
}

class Y extends X
```

```
{
    public int getV ()
    {
        return 1;
    }
}
```

Klasserna `X` och `Y` används i ett program så här:

```
class UseXY
{
    public static void main (String[] args)
    {
        X    x1 = new X ();
        X    x2 = new Y ();
        // Y    y1 = new X ();         // (1)
        Y    y2 = new Y ();

        int    i1 = x1.getU ();
        // int    i2 = x1.getV ();     // (2)
        int    i3 = x2.getU ();
        // int    i4 = x2.getV ();     // (3)
        int    i5 = y2.getU ();
        int    i6 = y2.getV ();
        System.out.println (i1 + " " + i3 + " " + i5 + " " + i6);
    }
}
```

a) Vilken utskrift skapas när programmet `UseXY` exekveras?

b) Vad händer om någon av de bortkommenterade satserna inkluderas i programmet? Varför?

Övning 10

Klasserna `X` och `Y` implementeras och används så här:

```
class X
{
    protected int    n;

    public X (int n)
    {
        this.n = n;
    }

    public String toString ()
    {
        return "" + n;
```

```
    }
}

class Y extends X
{
    public Y (int n)
    {
        super (n);
    }

    public String toString ()
    {
        return "" + (1 + n);
    }
}

class MixedTypes
{
    public static void main (String[] args)
    {
        X[]    x = new X[4];
        x[0] = new X (4);
        x[1] = new Y (4);
        x[2] = new X (10);
        x[3] = new Y (10);

        for (int i = 0; i < x.length; i++)
            System.out.println (x[i]);
    }
}
```

a) Man lagrar i vektorn (som refereras av referensen) `x` både objekt av typen `X` och objekt av typen `Y`. Varför är det möjligt?

b) Vilken utskrift skapas när programmet `MixedTypes` exekveras? Förklara!

Övning 11

Klasserna `X` och `Y` definieras så här:

```
class X
{
    public boolean equalsWith (X x)
    {
        return true;
    }
}

class Y extends X
```

```
{
    public boolean equalsWith (Y y)
    {
        return false;
    }
}
```

Man använder klasserna `X` och `Y` i ett program så här:

```
class UseXY
{
    public static void main (String[] args)
    {
        X    x1 = new X ();
        X    x2 = new X ();
        X    x3 = new Y ();
        Y    y1 = new Y ();
        Y    y2 = new Y ();

        boolean    b1 = x1.equalsWith (x2);
        boolean    b2 = x1.equalsWith (x3);
        boolean    b3 = x1.equalsWith (y1);
        boolean    b4 = y1.equalsWith (x1);
        boolean    b5 = y1.equalsWith (x3);
        boolean    b6 = y1.equalsWith (y2);
        boolean    b7 = x3.equalsWith (y2);

        System.out.println (b1 + " " + b2 + " " + b3
            + " " + b4 + " " + b5 + " " + b6 + " " + b7);
    }
}
```

a) Vilken utskrift skapas när programmet `UseXY` exekveras?

b) Vad händer i fall att metoden `equalsWith` i klassen `Y` tas bort?

Övning 12

Klasserna `X`, `Y` och `Z` definieras så här:

```
class X
{
    public int getValue (int i)
    {
        return 0 + i;
    }
}

class Y extends X
{
```

```
    public int getValue (int i)
    {
        return 1 + i;
    }
}

class Z extends X
{
    public int getValue (int i)
    {
        return 2 + i;
    }
}
```

Man använder klasserna `X`, `Y` och `Z` i ett program så här:

```
class UseXYZ
{
    public static void main (String[] args)
    {
        X    x = null;

        int    t = (int) (3 * Math.random ());
        if (t == 0)
            x = new X ();
        else if (t == 1)
            x = new Y ();
        else
            x = new Z ();

        int    i = x.getValue (10);
        System.out.println (i);
    }
}
```

Exekvera programmet `UseXYZ` flera gånger och följ utskriften. Vad händer?

Kapitel 16
Klasshierarkier

Övning 1

Klasserna `X`, `Y`, `Z` och `U` definieras så här:

```
class X
{}

class Y extends X
{}

class Z extends X
{}

class U extends Z
{}
```

Dessa klasser används så här::

```
class UseXYZU
{
    public static void main (String[] args)
    {
        X    x1 = new X ();
        X    x2 = new Y ();
        X    x3 = new Z ();
        X    x4 = new U ();

        Y    y1 = new X ();
        Y    y2 = new Y ();
        Y    y3 = new Z ();
        Y    y4 = new U ();

        Z    z1 = new X ();
        Z    z2 = new Y ();
        Z    z3 = new Z ();
        Z    z4 = new U ();

        U    u1 = new X ();
        U    u2 = new Y ();
```

```
        U    u3 = new Z ();
        U    u4 = new U ();
    }
}
```

a) Rita den klasshierarki som består av klasserna `X`, `Y`, `Z` och `U`.

b) Det går inte att kompilera programmet `UseXYZU`. Varför? Vilka satser ska tas bort?

Övning 2

Klasserna `X`, `Y`, `Z` och `U` definieras så här:

```
class X
{
    public void printX ()
    {
        System.out.println ("XXXXX");
    }
}

class Y extends X
{
    public void printY ()
    {
        System.out.println ("YYYYY");
    }

}

class Z extends X
{
    public void printZ ()
    {
        System.out.println ("ZZZZZ");
    }

}

class U extends Z
{
    public void printU ()
    {
        System.out.println ("UUUUU");
    }

}
```

Dessa klasser används så här:

```
class UseXYZU
{
    public static void main (String[] args)
    {
        X    x = new U ();
        x.printX ();
        x.printY ();
        x.printZ ();
        x.printU ();

        U    u = new U ();
        u.printX ();
        u.printY ();
        u.printZ ();
        u.printU ();
    }
}
```

a) Det går inte att kompilera programmet `UseXYZU`. Varför? Ta bort de felaktiga satserna så att programmet kan kompileras.

b) Vilken utskrift skapas när det rättade programmet körs?

Övning 3

Klassen `ExitFrame` definieras och används så här:

```
class ExitFrame extends javax.swing.JFrame
{
    public ExitFrame()
    {
        this.setTitle (" Exit Frame");
        this.setDefaultCloseOperation (
                  javax.swing.JFrame.EXIT_ON_CLOSE);
    }
}

class UseFrame
{
    public static void main (String[] args)
    {
        javax.swing.JFrame    f1 = new javax.swing.JFrame ();
        f1.setSize (400, 300);
        f1.setLocation (10, 50);
        f1.setVisible (true);

        ExitFrame    f2 = new ExitFrame ();
        f2.setSize (400, 300);
        f2.setLocation (500, 50);
```

```
        f2.setVisible (true);
    }
}
```

a) Exekvera programmet `UseFrame`. Hur avslutar man programmet?

b) Varifrån kommer metoderna `setTitle`, `setDefaultCloseOperation`, `setSize`, `setLocation` och `setVisible`?

Övning 4

Klassen `EmptyQueueException` representerar en undantagssituation:

```
// en klass som representerar den undantagssituation som uppstår
// när man vill ta ett element från en tom kö
class EmptyQueueException extends IllegalStateException
{
    public EmptyQueueException ()
    {
        super ();
    }

    public EmptyQueueException (String message)
    {
        super (message);
    }
}
```

Klassen `IntegerQueue` representerar en kö för heltal:

```
class IntegerQueue
{
    private Integer[]    elements;

    private int    firstIndex = 0;
    private int    lastIndex = -1;

    public IntegerQueue (int capacity)
    {
        elements = new Integer[capacity];
    }

    public boolean isEmpty ()
    {
        return elements[firstIndex] == null;
    }

    public void put (Integer element)
    {
```

```
        lastIndex = (lastIndex + 1) % elements.length;
        elements[lastIndex] = element;
    }

    public Integer take () // throws EmptyQueueException
    {
        // if (this.isEmpty ())
        //     throw new EmptyQueueException ("empty queue");

        Integer    firstElement = elements[firstIndex];
        elements[firstIndex] = null;
        firstIndex = (firstIndex + 1) % elements.length;

        return firstElement;
    }
}
```

En heltalskö skapas och används så här:

```
IntegerQueue    queue = new IntegerQueue (4);
Integer    k = queue.take ();     // (1)
queue.put (5);                    // (2)
System.out.println (queue.isEmpty ());
```

a) Hur ser ut kön när satsen `(1)` har utförts – rita den.

b) Hur ser ut kön när satsen `(2)` har utförts – rita den.

c) Vilken utskrift skapas när det givna kodavsnittet utförs? Varför?

d) Vad händer i det givna kodavsnittet när de bortkommenterade satserna i metoden `take` inkluderas?

e) Vilka metoder ärver klassen `EmptyQueueException`?

Övning 5

Klasserna `X` och `Y` skapas så här:

```
abstract class X
{
    private int    n;

    public X (int n)
    {
        this.n = n;
    }

    public String toString ()
```

```
    {
        return "[" + n + "]";
    }

    public abstract void combine (X x);
}

class Y extends X
{
    private int    p;

    public Y (int n, int p)
    {
        super (n);
        this.p = p;
    }

    public String toString ()
    {
        return super.toString () + "[" + p + "]";
    }

    public void combine (X x) throws IllegalArgumentException
    {
        if (!(x instanceof Y))
            throw new IllegalArgumentException ("" + x);

        Y    y = (Y) x;
        this.p += y.p;
    }
}
```

Klasserna `X` och `Y` används så här:

```
class UseXY
{
    public static void main (String[] args)
    {
        X    x = new Y (0, 1);
        Y    y = new Y (3, 4);
        System.out.println (x);
        System.out.println (y);
        System.out.println ();

        x.combine (y);
        System.out.println (x);
        y.combine (x);
        System.out.println (y);
    }
}
```

Vilken utskrift skapas när programmet `UseXY` exekveras?

Övning 6

Klasserna `X` och `Y` skapas så här:

```
abstract class X
{
    public abstract String getString ();

    public void outString ()
    {
        System.out.println (this.getString ());
    }
}

class Y extends X
{
    public String getString ()
    {
        return "YYYYY";
    }
}
```

Klasserna `X` och `Y` används så här:

```
class UseXY
{
    public static void main (String[] args)
    {
        X    x = new Y ();
        Y    y = new Y ();

        x.outString ();
        y.outString ();
    }
}
```

Vilken utskrift skapas när programmet `UseXY` exekveras?

Övning 7

Klasserna `Man`, `Swedishman` och `Englishman` skapas så här:

```
abstract class Man
{
    public abstract void speak ();
}

class Swedishman extends Man
{
```

```
    public void speak ()
    {
        System.out.println ("Jag tycker om dig!");
    }
}

class Englishman extends Man
{
    public void speak ()
    {
        System.out.println ("I love you!");
    }
}
```

Klasserna `Man`, `Swedishman` och `Englishman` används så här i ett program:

```
class Speakers
{
    public static void main (String[] args)
    {
        Man[]    m = new Man[5];
        m[0] = new Swedishman ();
        m[1] = new Englishman ();
        m[2] = new Englishman ();
        m[3] = new Swedishman ();
        m[4] = new Swedishman ();

        for (int i = 0; i < m.length; i++)
            m[i].speak ();
    }
}
```

Vilken utskrift skapas när programmet `Speakers` exekveras?

Övning 8

Klasserna `X`, `Y`, `Z` och `U` skapas så här:

```
abstract class X
{
    public abstract void outInteger ();
    public abstract void outString ();
}

class Y extends X
{
    public void outInteger ()
    {
        System.out.println (1);
```

```
    }

    public void outString ()
    {
        System.out.println ("YYYYY");
    }
}

class Z extends X
{
    public void outInteger ()
    {
        System.out.println (2);
    }

    public void outString ()
    {
        System.out.println ("ZZZZZ");
    }
}

class U extends Z
{
    public void outInteger ()
    {
        System.out.println (3);
    }

    public void outDouble ()
    {
        System.out.println (3.0);
    }
}
```

Klasserna `X`, `Y`, `Z` och `U` används så här:

```
class UseXYZU
{
    public static void main (String[] args)
    {
        X[]    x = new X[4];
        for (int i = 0; i < x.length; i++)
        {
            int    typ = (int) (3 * Math.random ()) + 1;

            if (typ == 1)
                x[i] = new Y ();
            else if (typ == 2)
                x[i] = new Z ();
            else if (typ == 3)
                x[i] = new U ();
```

```
        }

        int    countU = 0;
        for (int i = 0; i < x.length; i++)
        {
            x[i]. outInteger ();

            if (x[i] instanceof Z)
                x[i].outString ();

            if (x[i] instanceof U)
            {
                countU++;

                U    u = (U) x[i];
                u.outDouble ();
            }

            System.out.println ();
        }

        System.out.println (countU);
    }
}
```

Exekvera programmet `UseXYZU` flera gånger och följ noggrant utskriften. Vad händer under exekveringen?

Övning 9

Klassen `Int` skapas så här:

```
class Int
{
    private int    n;

    public Int (int n)
    {
        this.n = n;
    }

    public boolean equals (Object obj)
    {
        Int    i = (Int) obj;

        return this.n == i.n;
    }
}
```

Klassen `Int` används så här i ett program:

```
class EqualObjects
{
    public static void main (String[] args)
    {
        Int    i1 = new Int (4);
        Int    i2 = new Int (4);
        Integer    k = new Integer (4);

        boolean    b1 = i1.equals (i2);
        System.out.println (b1);
        boolean    b2 = i1.equals (k);
        System.out.println (b2);
    }
}
```

a) Vad händer när programmet `EqualObjects` exekveras? Varför?

b) Utforma metoden `equals` på så sätt att den returnerar `false` i fall att man anger ett argument som inte är av typen `Int`.

Övning 10

Klassen `Int` skapas så här:

```
class Int
{
    private int    n;

    public Int (int n)
    {
        this.n = n;
    }

    public String toString ()
    {
        return "[" + n + "]";
    }

    public boolean equals (Object obj)
    {
        boolean    b = false;

        if (obj instanceof Int)
        {
            Int    i = (Int) obj;
            b = this.n == i.n;
        }
```

```
        return b;
    }

    public int hashCode ()
    {
        return n;
    }
}
```

Klassen `Int` används så här i ett program:

```
class IntAsKey
{
    public static void main (String[] args)
    {
        Int    k1 = new Int (1);
        Int    k2 = new Int (2);
        String    s1 = "1 one";
        String    s2 = "2 two";
        java.util.HashMap<Int, String>    t =
                        new java.util.HashMap<> ();
        t.put (k1, s1);
        t.put (k2, s2);

        int    p = (int) (2 * Math.random () + 1);
        Int      k = new Int (p);
        String    s = t.get (k);
        System.out.println (k + ": " + s);
    }
}
```

När programmet `IntAsKey` exekveras flera gånger, kan olika utskrifter uppstå. Vilka utskrifter är möjliga?

Övning 11

Klassen `X` och klassen `Y` skapas så här:

```
class X
{
    private int    n;

    public X (int n)
    {
        this.n = n;
    }

    public int getN ()
```

```
    {
        return n;
    }

    public String toString ()
    {
        return "[" + n + "]";
    }
}


class Y extends X
{
    private int    p;

    public Y (int n, int p)
    {
        super (n);
        this.p = p;
    }

    public String toString ()
    {
        return "[" + this.getN () + ", " + p + "]";
    }
}
```

Klasserna `X` och `Y` används så här i ett program:

```
class UseXY
{
    public static void main (String[] args)
    {
        Object[]    v = new Object[5];
        v[0] = new String ("balance");
        v[1] = new X (2);
        v[2] = new Y (3, 4);
        v[3] = new Integer (10);
        v[4] = new Y (5, 6);
        showObjects (v);
        System.out.println ();

        for (int i = 0; i < v.length; i++)
        {
            System.out.print (v[i].toString ());


            if (v[i] instanceof X)
            {
                X    x = (X) v[i];
                System.out.print (" " + x.getN ());
            }
```

```
            if (v[i].getClass () == X.class)
            {
                X    x = (X) v[i];
                System.out.print (" " + x.getN ());
            }
            System.out.println ();
        }
    }

    public static void showObjects (Object[] obj)
    {
        for (int i = 0; i < obj.length; i++)
            System.out.println (obj[i]);
    }
}
```

Vilken utskrift skapas när programmet `UseXY` exekveras?

Övning 12

Klasserna `X`, `Y`, `X1`, och `X2` skapas så här:

```
class X
{
    private int    a;

    public X (int a)
    {
        this.a = a;
    }

    public String toString ()
    {
        return "" + a;
    }

    public int getX ()
    {
        return 0;
    }
}

class Y
{
    private int    b;


    public Y (int b)
```

```
    {
        this.b = b;
    }

    public String toString ()
    {
        return "" + b;
    }
}


class X1 extends X
{
    private int    a1;

    public X1 (int a, int a1)
    {
        super (a);
        this.a1 = a1;
    }

    public String toString ()
    {
        return super.toString () + ", " + a1;
    }


    public int getX ()
    {
        return 1;
    }
}

class X2 extends X
{
    private int    a2;

    public X2 (int a, int a2)
    {
        super (a);
        this.a2 = a2;
    }

    public String toString ()
    {
        return super.toString () + ", " + a2;
    }

    public int getX ()
    {
        return 2;
```

```
    }
}
```

Klasserna `X`, `Y`, `X1` och `X2` används så här i ett program:

```
class RelatedObjects
{
    public static void main (String[] args)
    {
        X[]    x = new X[4];
        x[0] = new X (10);
        x[1] = new X1 (11, 1);
        x[2] = new X2 (12, 2);
        x[3] = new Y (55);

        for (int i = 0; i < x.length; i++)
        {
            System.out.println (x[i].toString ());
            System.out.println (x[i].getX ());
        }
    }
```

}

a) Varför går det inte att kompilera programmet `RelatedObjects`?

b) Rätta programmet så att den kan kompileras. Vilken utskrift skapas när programmet exekveras i så fall??

c) Klasserna `X`, `Y`, `X1` och `X2` används även så här::

```
class AllObjects
{
    public static void main (String[] args)
    {
        Object[]    obj = new Object[4];
        obj[0] = new X (10);
        obj[1] = new X1 (11, 1);
        obj[2] = new X2 (12, 2);
        obj[3] = new Y (55);

        for (int i = 0; i < obj.length; i++)
        {
            System.out.println (obj[i].toString ());
            System.out.println (obj[i].getX ());
        }
    }
}
```

Vilket problem uppstår i det här fallet, och hur kan det lösas?

Övning 13

Ett program skapas så här:

```
import java.io.*;

class Writers
{
    public static void main (String[] args) throws IOException
    {
        String    s = "Peace and harmony";

        OutputStreamWriter    sout =
                          new OutputStreamWriter (System.out);
        writeTo (s, sout);

        OutputStreamWriter    osw = new OutputStreamWriter (
                                 new FileOutputStream ("fil.txt"));
        BufferedWriter    fout = new BufferedWriter (osw);
        writeTo (s, fout);
    }

    public static void writeTo (String s, Writer w)
                                        throws IOException
    {
        w.write (s);
        w.flush ();
    }
}
```

a) Vad händer när programmet `Writers` exekveras?

b) Vilka klasser finns i den klasshierarki som har klassen `java.io.Writer` som sin rotklass?

c) Vilka klasser finns i den klasshierarki som har klassen `java.io.Reader` som sin rotklass?

d) Vilka klasser finns i de klasshierarkier som har klasserna `java.io.OutputStream` och `java.io.InputStream` som sina rotklasser?

e) Vilka klasser finns i den klasshierarki som har klassen `java.io.IOException` som sin rotklass?

Övning 14

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 givna kodavsnittet exekveras? Vad händer om man inkluderar satsen `(1)`?

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

Övning 15

```
class ObjectPair<U, V>
{
    private Object    obj1;
    private Object    obj2;

    public ObjectPair (U obj1, V obj2)
    {
        this.obj1 = obj1;
        this.obj2 = obj2;
    }

    public String toString ()
    {
        return "[" + obj1 + ", " + obj2 + "]";
    }

    public U getObject1 ()
    {
        return (U) obj1;
```

```
    }

    public V getObject2 ()
    {
        return (V) obj2;
    }

    public void setObject1 (U obj1)
    {
        this.obj1 = obj1;
    }

    public void setObject2 (V obj2)
    {
        this.obj2 = obj2;
    }
}

class ObjectType
{
    public static void main (String[] args)
    {
        Integer    n = new Integer (1);
        String     s = new String ("one");
        ObjectPair<Integer, String>    op =
                     new ObjectPair<Integer, String> (n, s);

        System.out.println (op + " ");
        Integer    o1 = op.getObject1 ();
        String     o2 = op.getObject2 ();
        System.out.println (o1);
        System.out.println (o2);

        // oc.setObject1 (s);      // (1)
        // oc.setObject2 (n);      // (2)
    }
}
```

a) Vilken utskrift skapas när programmet exekveras?

b) Vad händer när satsen `(1)`, eller satsen `(2)`, inkluderas? Varför?

Övning 16

```
import java.util.*;;

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

```
    {
        AbstractList<Integer>    array = new ArrayList<> ();
        for (int i = 0; i < 10; i++)
            array.add (i + 1);
        System.out.println (array);

        AbstractList<String>   list = new LinkedList<> ();
        Collections.addAll (list,
                            "one", "two", "three", "four", "five");
        System.out.println (list);
        System.out.println ();

        reverse (array);
        reverse (list);
        System.out.println (array);
        System.out.println (list);
        System.out.println ();

        String    total = "";
        for (String s : list)
            total = total.concat (s + " ");
        System.out.println (total);
    }

    public static <E> void reverse (AbstractList<E> list)
    {
        int    size = list.size ();
        E    e = null;
        int    pos = 0;
        for (int i = 0; i < size / 2; i++)
        {
            e = list.get (i);
            pos = size - 1 - i;
            list.set (i, list.get (pos));
            list.set (pos, e);
        }
    }
}
```

a) Vilken utskrift skapas när programmet `Lists` exekveras?

b) Åskådliggör den algoritm som används i metoden `reverse`.

Övning 17

Klassen `ObjectStack` representerar en stack:

```
class ObjectStack
{
```

```
    private Object[]    elements;
    private int    lastIndex = -1;

    public ObjectStack (int capacity)
    {
        elements = new Object[capacity];
    }

    public boolean isEmpty ()
    {
        return lastIndex == -1;
    }

    public void put (Object element)
    {
        lastIndex++;
        elements[lastIndex] = element;
    }

    public Object take ()
    {
        Object    lastElement = elements[lastIndex];
        elements[lastIndex] = null;
        lastIndex--;

        return lastElement;
    }
}
```

En stack skapas och används så här:

```
ObjectStack    stack = new ObjectStack (4);
stack.put (new Integer (1));
stack.put (new String ("s2"));
stack.put (new Integer (3));
stack.put (new String ("s4"));

String     s = (String) stack.take ();
Integer    n = (Integer) stack.take ();
stack.put (new Integer (5));                // (1)

Object    obj = null;
while (!stack.isEmpty ())
{
    obj = stack.take ();
    System.out.print (obj + "  ");
}
```

a) Man lagrar i den skapade stacken både objekt av typen `java.lang.Integer` och objekt av typen `java.lang.String`. Varför är det möjligt?

b) Vilken utskrift skapas när det givna kodavsnittet utförs?

c) Hur ser ut stacken när satsen `(1)` har utförts – rita den.

Övning 18

Klassen `Stack` representerar en stack:

```
class Stack<E>
{
    private Object[]    elements;
    private int    lastIndex = -1;

    public Stack (int capacity)
    {
        elements = new Object[capacity];
    }

    public boolean isEmpty ()
    {
        return lastIndex == -1;
    }

    public void put (E element)
    {
        lastIndex++;
        elements[lastIndex] = element;
    }

    public E take ()
    {
        Object    lastElement = elements[lastIndex];
        elements[lastIndex] = null;
        lastIndex--;

        return (E) lastElement;
    }
}
```

Två instanser av klassen `Stack` skapas och används så här:

```
Stack<Integer>    stack1 = new Stack<> (4);
stack1.put (new Integer (1));
stack1.put (new Integer (2));
stack1.put (new Integer (3));
// stack1.put (new String ("s4"));       // (1);

Integer     n = stack1.take ();
System.out.println (n);
```

```
Stack<String>    stack2 = new Stack<> (4);
stack2.put (new String ("s1"));
stack2.put (new String ("s2"));
stack2.put (new String ("s3"));
// stack2.put (new Integer (4));         // (2);

String     s = stack2.take ();
System.out.println (s);
```

a) Vilken utskrift skapas när det givna kodavsnittet utförs?

b) Vad händer om man inkluderar satsen `(1)`, eller satsen `(2)`?

Problem 1

Schackpjäsernas presentationer

A) Ett schackbräde och schackpjäserna

Ett schackbräde består av 8 x 8 fält.

Ett fält på ett schackbräde är bestämt med sin rad och sin kolumn. Raderna kan betecknas med bokstäverna a, b, c, d, e, f, g och h, och kolumnerna med siffrorna 1, 2, 3, 4, 5, 6, 7 och 8. Fältet a4 finns i så fall i raden a och kolumnen 4. Ett fällt är en behållare för en pjäs: en pjäs kan ställas på fältet, eller tas av det. Ett fält är också en informationsbärare: det går att markera det eller att ta bort markeringen.

Till ett schackbräde hör ett antal schackpjäser. En schackpjäs kan stiga på brädet, på ett visst fält, eller stiga av det. När den väl är på brädet, kan den markera alla de fält som den kan nå med ett enda steg. Den kan även ta bort markeringarna. En pjäs kan ta reda på om den är på brädet eller utanför den.

Pjäserna skiljer sig från varandra enligt sin färg – vit eller svart, och enligt sitt namn. En pjäs är antingen en bonde (eng. *Pawn*), ett torn (eng. *Rook*), en springare (eng. *Knight*), en löpare (eng. *Bishop*), en dam (eng. *Queen*) eller en kung (eng. *King*).

En modell av ett schackbräde och schackpjäserna ska skapas.

En modell av ett schackbräde och schackpjäserna – ej fullständig

```
public class Chessboard
{
```

```
public static class Field
{
    private char    row;
    private byte    column;
    private Chesspiece    piece = null;
    private boolean    marked = false;

    public Field (char row, byte column) {}

    public void put (Chesspiece piece) {}

    public Chesspiece take () {}

    public void mark () {}

    public void unmark ()  {}

    public String toString ()
    {
        String    s = (marked)? "xx" : "--";
        return (piece == null)? s : piece.toString ();
    }
}


public static final int    NUMBER_OF_ROWS = 8;
public static final int    NUMBER_OF_COLUMNS = 8;

public static final int    FIRST_ROW = 'a';
public static final int    FIRST_COLUMN = 1;

private Field[][]    fields;

public Chessboard ()
{
    fields = new Field[NUMBER_OF_ROWS][NUMBER_OF_COLUMNS];
    char    row = 0;
    byte    column = 0;
    for (int r = 0; r < NUMBER_OF_ROWS; r++)
    {
        row = (char) (FIRST_ROW + r);
        column = FIRST_COLUMN;
        for (int c = 0; c < NUMBER_OF_COLUMNS; c++)
        {
            fields[r][c] = new Field (row, column);
            column++;
        }
    }
}
```

```
public String toString ()  {}

public boolean isValidField (char row, byte column)  {}


public abstract class Chesspiece
{
    private char    color;
    // w - white, b - black

    private char    name;
    // K - King, Q - Queen, R - Rook, B - Bishop, N - Knight,
    // P – Pawn

    protected char    row = 0;
    protected byte    column = -1;

    protected Chesspiece (char color, char name)  {}

    public String toString ()
    {
        return "" + color + name;
    }

    public boolean isOnBoard ()
    {
        return Chessboard.this.isValidField (row, column);
    }

    public void moveTo (char row, byte column)
                               throws NotValidFieldException
    {
        if (!Chessboard.this.isValidField (row, column))
              throw new NotValidFieldException (
                          "bad field: " + row + column );

        this.row = row;
        this.column = column;

        int    r = row - FIRST_ROW;
        int    c = column - FIRST_COLUMN;
        Chessboard.this.fields[r][c].put (this);
    }

    public void moveOut ()  {}

    public abstract void markReachableFields ();

    public abstract void unmarkReachableFields ();
}
```

```
    public class Pawn extends Chesspiece
    {
        public Pawn (char color, char name)
        {
            super (color, name);
        }

        public void markReachableFields ()
        {
            byte    col = (byte) (column + 1);
            if (Chessboard.this.isValidField (row, col))
            {
                int    r = row - FIRST_ROW;
                int    c = col - FIRST_COLUMN;
                Chessboard.this.fields[r][c].mark ();
            }
        }

        public void unmarkReachableFields ()
        {
            byte    col = (byte) (column + 1);
            if (Chessboard.this.isValidField (row, col))
            {
                int    r = row - FIRST_ROW;
                int    c = col - FIRST_COLUMN;
                Chessboard.this.fields[r][c].unmark ();
            }
        }
    }


    public class Rook extends Chesspiece  {}

    public class Knight extends Chesspiece  {}

    public class Bishop extends Chesspiece  {}

    public class Queen extends Chesspiece  {}

    public class King extends Chesspiece  {}
}
```

Uppgifter i samband med schackbrädet och schackpjäserna

1. Komplettera definitionsklassen `Chessboard` och alla klasser inuti den. Skapa undantagsklassen `NotValidFieldException`. Beskriv klasserna och deras medlemmar.

2. Varför skapas klassen `Field` som en nästlad klass? Kan den skapas utanför klassen `Chessboard`?

Varför skapas de klasser som representerar pjäserna som inre klasser? Kan de skapas utanför klassen `Chessboard`?

3. Skapa ett enkelt testprogram, där ett objekt av klassen `Chessboard` och flera objekt av de klasser som representerar pjäserna skapas och används.

B) Pjäsernas presentationer

Programmet `ReachableFieldsOnChessboard` skapar ett schackbräde och ett antal pjäser. Detta görs så här:

```
Chessboard    chessBoard = new Chessboard ();
System.out.println (chessBoard + "\n");

Chessboard.Chesspiece[]  pieces = new Chessboard.Chesspiece[6];
pieces[0] = chessBoard.new Pawn ('w', 'P');
pieces[1] = chessBoard.new Rook ('b', 'R');
pieces[2] = chessBoard.new Queen ('w', 'Q');
pieces[3] = chessBoard.new Bishop ('w', 'B');
pieces[4] = chessBoard.new King ('b', 'K');
pieces[5] = chessBoard.new Knight ('w', 'N');
```

Sedan presenterar sig var och en av pjäserna. En pjäs stiger på schackbrädet – på ett slumpmässigt fält, och markerar alla de fält som den kan nå med ett enda steg. Pjäsen väntar en stund, och därefter tar bort markeringarna. Till sist stiger pjäsen av, så att nästa pjäs kan komma och presentera sig.

Schackbrädet visas vid varje presentation. När en vit springare exempelvis stiger på och markerar de fält som den kan nå, kan schackbrädet se ut så här:

```
   1  2  3  4  5  6  7  8
a  -- xx -- -- -- xx -- --
b  -- -- -- wN -- -- -- --
c  -- xx -- -- -- xx -- --
d  -- -- xx -- xx -- -- --
e  -- -- -- -- -- -- -- --
f  -- -- -- -- -- -- -- --
g  -- -- -- -- -- -- -- --
```

```
h  -- -- -- -- -- -- -- --.
```

Uppgifter i samband med pjäsernas presentationer

1. Skapa och prova programmet `ReachableFieldsOnChessboard`.

2. Pjäserna kan lagras i en gemensam vektor, trots att de är av olika typer. Varför är det möjligt? Hur skulle programmet se ut utan denna möjlighet?

3. Trots skillnader i deras beteenden, kan pjäserna presentera sig på ett gemensamt sätt – i en loop. Varför är det möjligt? Finns det något alternativ till detta?

Kapitel 17

Gränssnitt

Övning 1

Gränssnittet `Measurable` och klassen `Rectangle` definieras så här:

```
// Measurable.java

public interface Measurable
{
    double getWidth ();
    double getHeight ();
}


// Rectangle.java

public class Rectangle implements Measurable
{
    private double    w;
    private double    h;

    public Rectangle (int w, int h)
    {
        this.w = w;
        this.h = h;
    }

    public double getWidth ()
    {
        return w;
    }
}
```

Det går inte att kompilera klassen `Rectangle`. Varför?

Komplettera klassen, så att den kan kompileras.

Övning 2

Gränssnittet `Preferable`, och klasserna `A`, `B` och `Filter`, skapas så här:

```
interface Preferable
{}

class A implements Preferable
{
    public String toString ()
    {
        return "with love";
    }
}

class B
{
    public String toString ()
    {
        return "without love";
    }
}

class Filter
{
    public static void main (String[] args)
    {
        Object[]    obj = new Object[4];
        obj[0] = new A ();
        obj[1] = new B ();
        obj[2] = new B ();
        obj[3] = new A ();

        for (int i = 0; i < obj.length; i++)
            if (obj[i] instanceof Preferable)
                System.out.println (obj[i]);
    }
}
```

Vilken utskrift skapas när programmet `Filter` exekveras? Varför?

Övning 3

Gränssnittet `MathConstants`, klassen `Cirkel` och klassen `ShowConstants` definieras så här:

```
public interface MathConstants
{
```

```
    double    PI = 3.14;
    double    E  = 2.7;
}

public class Cirkel implements MathConstants
{
    private double    r;

    public Cirkel (double r)
    {
        this.r = r;
    }

    public double omkrets ()
    {
        return 2 * r * PI;
    }
}

class ShowConstants
{
    public static void main (String[] args)
    {
        System.out.println (PI + " " + E);
    }
}
```

a) Det går inte att kompilera programmet `ShowConstants` – rätta det.

b) Det går att skriva `PI` i klassen `Cirkel`, men inte i klassen `ShowConstants`. Varför?

Övning 4

Klasserna `Int` och `UseInt` är givna, som nedan:

```
class Int
{
    private int    n;

    public Int (int n)
    {
        this.n = n;
    }

    public String toString ()
    {
        return "" + n;
    }
}
```

```
class UseInt
{
    public static void main (String[] args)
    {
        Int[]    i = new Int[4];
        i[0] = new Int (2);
        i[1] = new Int (4);
        i[2] = new Int (1);
        i[3] = new Int (3);

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

Vid exekveringen av programmet `UseInt` kastas ett undantag i metoden `sort`: metoden kan inte sortera en vektor med objekt av typen `Int`. Om man vill kunna sortera vektorn, måste man lägga till ytterligare funktionalitet i klassen `Int`.

Komplettera klassen `Int`, så att följande utskrift erhålls vid exekveringen av programmet `UseInt`:

```
1 2 3 4
```

Övning 5

I programmet `CharSequences` använder man teckensekvenser av olika typer:

```
import java.util.*;

class CharSequences
{
    public static void main (String[] args)
    {
        CharSequence[]    cs = { new String ("oneness "),
                                 new StringBuilder ("balance "),
                                 new StringBuffer ("harmony "),
                                 new String ("peace ") };
        System.out.println (Arrays.toString (cs));

        CharSequence    seq = concat (cs);
        System.out.println (seq);
    }


    public static CharSequence concat (CharSequence[] cs)
    {
        StringBuilder    total = new StringBuilder ();
```

```
        for (int i = 0; i < cs.length; i++)
            total.append (cs[i]);

        return total;
    }
}
```

a) Ange två klasser som implementerar gränssnittet `java.lang.CharSequence`. Vilka metoder finns i detta gränssnitt?

b) På vilka ställen i programmet `CharSequences` utnyttjas referenser av typen `CharSequence`? Vart refererar dem?

c) Vilken utskrift skapas när programmet `CharSequences` exekveras?

Övning 6

Klassen `Rectangle2D` och klassen `Ellipse2D` är definierade i paketet `java.awt.geom`. I samma paket finns även flera klasser som heter `Double`. I paketet `java.awt` finns ett gränssnitt som heter `Shape`, och en abstrakt klass som heter `Graphics2D`.

a) Följande sats är korrekt:

```
Rectangle2D    rect = new Rectangle2D.Double (10.0, 20.0, 60.0, 40.0);
```

Vilken relation finns mellan klassen `Rectangle2D` och klassen `Double`? Ange två slutsatser.

b) Följande kodavsnitt är korrekt:

```
Shape[]    shapes = new Shape[4];
shapes[0] = new Rectangle2D.Double (10.0, 20.0, 60.0, 40.0);
shapes[1] = new Ellipse2D.Double (70.0, 80.0, 40.0, 60.0);
```

Vilken relation finns mellan klassen `Rectangle2D.Double` och gränssnittet `Shape`, och mellan klassen `Ellipse2D.Double` och gränssnittet `Shape`?

c) I klassen `Graphics2D` finns en metod som heter `draw`, som kan rita figurer av olika typer. Metoden deklareras så här:

```
public abstract void draw (Shape s)
```

Låt `g` vara en referens till ett objekt av en icke-abstrakt subklass till klassen `Graphics2D`. Använd detta objekt och dess metod `draw` för att rita två figurer av olika klasser.

Övning 7

I programmet `ObjectsOutIn` skrivs flera objekt: ut till en fill, och läses sedan in från den:

```
import java.io.*;
import java.util.*;

class ObjectsOutIn
{
    public static void main (String[] args) throws Exception
    {
        Object[]    obj = { new Integer (25),
                            new String ("ett"),
                            new StringBuilder ("tio") };
        System.out.println (Arrays.toString (obj));

        ObjectOutput    out = new ObjectOutputStream (
                              new FileOutputStream ("fil.obj"));
        int    n = writeObjectsTo (obj, out);

        ObjectInput    in = new ObjectInputStream (
                            new FileInputStream ("fil.obj"));
        List<Object>    list = readObjectsFrom (n, in);
        System.out.println (list);
    }

    public static int writeObjectsTo (Object[] obj,
                            ObjectOutput out) throws Exception
    {
        int    count = 0;
        for (int i = 0; i < obj.length; i++)
            if (obj instanceof Serializable)
            {
                out.writeObject (obj[i]);
                count++;
            }

        return count;
    }

    public static List<Object> readObjectsFrom (int count,
                             ObjectInput in) throws Exception
    {
        List<Object>    list = new ArrayList <> (count);
        for (int i = 0; i < count; i++)
            list.add (in.readObject ());

        return list;
    }
}
```

a) Ange en implementationsklass och två supergränssnitt för gränssnittet `java.io.ObjectOutput`. Vilka metoder finns i detta gränssnitt? Vart refererar referensen `out` i metoden `writeObjectsTo`?

b) Ange en implementationsklass och två supergränssnitt för gränssnittet `java.io.ObjectInput`. Vilka metoder finns i detta gränssnitt? Vart refererar referensen `in` i metoden `readObjectsFrom`?

c) Ange två implementationsklasser och två supergränssnitt för gränssnittet `java.util.List`. Vilka metoder finns i detta gränssnitt? Vart refererar referensen `list` i metoden `main`?

d) Vilka metoder finns i gränssnittet `java.io.Serializable`?

e) Vilken utskrift skapas när programmet `ObjectsOutIn` exekveras?

Övning 8

```
import static java.lang.System.out;

class FirstLast
{
    public static void main (String[] args)
    {
        Integer[]    u = new Integer[4];
        u[0] = 10;
        u[3] = 40;
        Integer    fu = getFirst (u);
        out.println (fu);

        String[]    v = new String[4];
        v[0] = "tio";
        v[3] = "fortio";
        String    fv = getFirst (v);
        out.println (fv);
        out.println ();

        int[]    w = new int[4];
        w[0] = 10;
        w[3] = 40;
        // int    fw = getFirst (w);  // (1)

        Integer    lu = getLast (u);
        String     lv = getLast (v);
        Integer    l = getLast (5, 10, 4);
        out.println (lu);
        out.println (lv);
        out.println (l);
```

```
        out.println ();

        Integer    f = getFirst (lu, lv);
        out.println (f);
    }

    public static <T> T getFirst (T[] v)
    {
        return v[0];
    }

    public static <T1, T2> T1 getFirst (T1 f, T2 s)
    {
        return f;
    }

    public static <T> T getLast (T... v)
    {
        return v[v.length - 1];
    }
}
```

a) Vilken utskrift skapas när programmet `FirstLast` exekveras?

b) Vad händer när man inkluderar satsen `(1)`? Varför?

Övning 9

```
class MeanValue
{
    public static void main (String[] args)
    {
        Integer[]    i = new Integer[4];
        i[0] = 10;
        i[1] = 20;
        i[2] = 30;
        i[3] = 40;
        System.out.println (meanValue (i));
        System.out.println (meanValue1 (i));

        String[]    s = new String[4];
        s[0] = "tio";
        s[1] = "tjugo";
        s[2] = "trettio";
        s[3] = "fortio";
        // System.out.println (meanValue (s));    // (1)
        // System.out.println (meanValue1 (s));   // (2)
    }
```

```
    public static <T extends Number> double meanValue (T... n)
    {
        double    sum = 0;
        for (int i = 0; i < n.length; i++)
            sum += n[i].doubleValue ();
        double    m = sum / n.length;

        return m;
    }

    public static double meanValue1 (Number... n)
    {
        double    sum = 0;
        for (int i = 0; i < n.length; i++)
            sum += n[i].doubleValue ();
        double    m = sum / n.length;

        return m;
    }
}
```

a) Vilken utskrift skapas när programmet `MeanValue` exekveras?

b) Vad händer när man inkluderar satsen `(1)`, eller satsen `(2)`? Varför?

Övning 10

```
import java.util.*;

class ShowCollections
{
    public static void main (String[] args)
    {
        Set<Integer>    set = new HashSet<> ();
        set.add (10);
        set.add (20);
        set.add (30);
        set.add (40);
        show (set);

        ArrayList<String>    list = new ArrayList<> ();
        list.add ("tio");
        list.add ("tjugo");
        list.add ("trettio");
        list.add ("fortio");
        show (list);

        String[]    v = {"tio", "tjugo", "trettio", "fortio"};
        // show (v);  // (1)
```

```
    }

    public static <T> void show (Collection<T> coll)
    {
        Iterator<T>    it = coll.iterator();
        T    e = null;
        String    s = "[ ";
        while(it.hasNext())
        {
            e = it.next();
            s = s + e + " ";
        }
        s += "]";

        System.out.println (s);
    }
}
```

a) Vilken utskrift skapas när programmet `ShowCollections` exekveras?

b) Vad händer när satsen `(1)` inkluderas? Varför?

c) Vad händer i fall att man byter parametertyp i metoden `show`: `List<T>` i stället för `Collection<T>`?

Övning 11

Gränssnittet `Sortable`, klassen `Char` och klassen `Mat` definieras så här:

```
public interface Sortable<T>
{
    int compareTo (T s);
}

public class Char implements Sortable<Char>
{
    private char    c;

    public Char (char c)
    {
        this.c = c;
    }

    public String toString ()
    {
        return "" + c;
    }

    public int compareTo (Char ch)
```

```
    {
        int    j = 0;
        if (this.c < ch.c)
            j = -1;
        else if (this.c > ch.c)
            j = 1;

        return j;
    }
}

public class Mat
{
    public static <T extends Sortable<T>> T max (T[] o)
    {
        T    m = o[0];
        for (int i = 1; i < o.length; i++)
            if (o[i].compareTo (m) > 0)
                m = o[i];

        return m;
    }
}
```

Klasserna `Char` och `Mat` används så här:

```
public class Chars
{
    public static void main (String[] args)
    {
        Char[]    c = new Char[5];
        c[0] = new Char ('c');
        c[1] = new Char ('b');
        c[2] = new Char ('a');
        c[3] = new Char ('e');
        c[4] = new Char ('d');

        Char    maxc = Mat.max (c);
        System.out.println (maxc);
    }
}
```

a) Vilken utskrift skapas när programmet `Chars` exekveras?

b) Kan man deklarera metoden `max` så här:

```
public static <T extends Object & Sortable<T>> T max (T[] o)
```

c) Kan man deklarera metoden `max` så här:

```
public static <T extends Sortable<? super T>> T max (T[] o)
```

Övning 12

```
import java.util.*;
import static java.lang.System.out;

class FillList
{
    public static void main (String[] args)
    {
        List<Integer>    list = new LinkedList<> ();
        Collections.addAll (list, 3, 2, 5, 1, 4);
        out.println (list);
        int    m = Collections.max (list);
        out.println (m);
        Collections.sort (list);
        out.println (list);

        fill (list, 5);
        out.println (list);
        Collections.fill (list, 10);
        out.println (list);
    }

    public static <T> void fill (List<? super T> list, T o)
    {
        int size = list.size();
        for (int pos = 0; pos < size; pos++)
            list.set(pos, o);
    }
}
```

Vilken utskrift skapas när programmet `FillList` exekveras?

Övning 13

Ett gränssnitt, `StringCollection`, definierar en samling teckensträngar.

En klass, `NodeStringCollection`, representerar en samling teckensträngar. En sådan samling är definierad i gränssnittet `StringCollection`. Element i samlingen lagras i en sekvens av noder.

```
class NodeStringCollection implements StringCollection
{
    private static class Node
    {
        public String    element;
        public Node      nextNode;

        public Node (String element)
```

```
        {
            this.element = element;
            this.nextNode = null;
        }
    }

    private Node    firstNode;

    public NodeStringCollection ()
    {
        firstNode = null;
    }

    // size

    // add

    public String toString ()
    {
        String    s = "";
        Node    node = firstNode;
        while (node != null)
        {
            s = s + node.element + " ";
            node = node.nextNode;
        }

        return s;
    }
}
```

Gränssnittet `StringCollection` och klassen `NodeStringCollection` kan användas så här:

```
StringCollection    sc = new NodeStringCollection ();
sc.add (new String ("A"));
sc.add (new String ("B"));
sc.add (new String ("C"));
sc.add (new String ("D"));
System.out.println (sc);

int    countStrings = sc.size ();
System.out.println (countStrings);
```

När den här kodsekvensen exekveras, erhålls följande utskrift:

```
D C B A
4
```

a) Skapa gränssnittet `StringCollection`.

b) Skapa metoden `size`.

c) Skapa metoden `add`.

Övning 14

Med klassen `EmptyStackException` och gränssnittet `Stack` definieras en typoberoende stack.

```
// EmptyStackException.java

/**********************************************************************

Klassen EmptyStackException representerar den undantagssituation som
uppstår när ett element i en stack försöker kommas åt när stacken är
tom.

**********************************************************************/

public class EmptyStackException extends IllegalStateException
{
    public EmptyStackException ()
    {
        super ();
    }

    public EmptyStackException (String message)
    {
        super (message);
    }
}


// Stack.java

/**********************************************************************

Gränssnittet Stack definierar en stack med obegränsad kapacitet.

**********************************************************************/

public interface Stack<E>
{
    // isEmpty returnerar true om stacken är tom, annars false
    boolean isEmpty ();

    // size returnerar antalet element i stacken
    int size ();
```

```
    // put sätter in ett givet element i stacken.
    void put (E element);

    // take tar ut det element ur stacken som sattes in sist, och
    // returnerar detta element.
    // Om stacken är tom kastas ett undantag av typen
    // EmptyStackException.
    E take () throws EmptyStackException;

    // peek returnerar det element i stacken som sattes in sist
    // (utan att ta ut det).
    // Om stacken är tom kastas ett undantag av typen
    // EmptyStackException.
    E peek () throws EmptyStackException;
}
```

a) Skapa en klass `ArrayStack`, som implementerar gränssnittet `Stack`. Stackens element ska lagras i en vektor av den inbyggda typen.

b) Skapa en klass `NodeStack`, som implementerar gränssnittet `Stack`. Stackens element ska lagras i en sekvens av noder.

c) Gränssnittet `Stack`, och implementationsklasserna `ArrayStack` och `NodeStack`, används på följande vis:

```
class UseInterface
{
    public static void main (String[] args)
    {
        Stack<Integer>    stack = new ArrayStack<> (4);
        // stack = new NodeStack<> ();                    // (1);
        for (int i = 1; i <= 5; i++)
            stack.put (i);

        Integer    n = createInteger (stack);
        System.out.println (n);
    }

    public static Integer createInteger (Stack<Integer> stack)
    {
        String    n = "";
        while (!stack.isEmpty ())
            n += stack.take ();

        return Integer.parseInt (n);
    }
}
```

Vilken utskrift skapas när metoden `main` utförs?

Vilken utskrift skapas i fall att satsen `(1)` inkluderas?

Vad händer om metoden `peek` används i stället för metoden `take`?

Övning 15

Gränssnittet `Set` definierar en mängd:

```
// Set.java

/**********************************************************************

Gränssnitet Set definierar en mängd med obegränsad kapacitet.

En mängd är en samling element, där ett element inte kan förekomma
flera gånger.

**********************************************************************/

import java.util.*;  // Iterable, Iterator

public interface Set<E> extends Iterable<E>
{
    // isEmpty returnerar true om mängden är tom, annars false
    boolean isEmpty ();

    // size returnerar antalet element i mängden
    int size ();

    // contains returnerar true om mängden innehåller ett element som
    // är likadant som ett givet element, annars false.
    boolean contains (E element);

    // add lägger till ett givet element till mängden. Om ett sådant
    // element redan finns i mängden, gör metoden ingenting.
    void add (E element);

    // remove tar bort det element i mängden som är likadant som
    // ett givet element. Om ett sådant element inte finns i mängden,
    // gör metoden ingenting.
    void remove (E element);

    // iterator returnerar en iterator till mängden.
    // Den returnerade iteratorn kan användas för iteration genom
    // mängden, och för borttagning av element från mängden.
    // Medan iteratorn används för iteration genom mängden, ska
    // mängden inte ändras på något sätt (förutom möjligen genom
    // själva iteratorn).
    Iterator<E> iterator ();
```

```
}
```

Klassen `ArraySet`, implementerar gränssnittet `Set`. Mängdens element lagras i en vektor av den inbyggda typen.

```
// ArraySet.java

import java.util.*;  // Iterator, NoSuchElementException
import java.io.*;    // Serializable

public class ArraySet<E> implements Set<E>, Serializable
{
    public static final int    DEFAULT_INITIAL_CAPACITY = 100;
    public static final int    INCREASE_VALUE = 25;  // 25%

    private E[]    elements;
    private int    lastIndex = -1;

    public ArraySet ()
    {
        elements = (E[]) (new Object[DEFAULT_INITIAL_CAPACITY]);
    }

    public ArraySet (int initialCapacity)
    {
        elements = (E[]) (new Object[initialCapacity]);
    }

    public boolean isEmpty ()
    {
        return lastIndex == -1;
    }

    public int size ()
    {
        return lastIndex + 1;
    }

    protected void increaseCapacity ()
    {
        int    newLength = 1 + elements.length
                         + INCREASE_VALUE * elements.length / 100;
        E[]    e = (E[]) new Object[newLength];
        for (int index = 0; index <= lastIndex; index++)
            e[index] = elements[index];

        elements = e;
    }
```

```
protected int indexOf (E element)
{
    int    indexOfElement = -1;
    int    index = 0;
    while (indexOfElement == -1  &&  index <= lastIndex)
    {
        if (element.equals (elements[index]))
            indexOfElement = index;

        index++;
    }

    return indexOfElement;
}

public boolean contains (E element)
{
    int    indexOfElement = this.indexOf (element);
    boolean    hasElement = indexOfElement != -1;

    return hasElement;
}

public void add (E element)
{
    if (!this.contains (element))
    {
        if (lastIndex == elements.length - 1)
            this.increaseCapacity ();

        lastIndex = lastIndex + 1;
        elements[lastIndex] = element;
    }
}

public void remove (E element)
{
    int    indexOfElement = this.indexOf (element);
    if (indexOfElement != -1)
    {
        elements[indexOfElement] = elements[lastIndex];
        elements[lastIndex] = null;
        lastIndex--;
    }
}


private class SetIterator implements Iterator<E>
{
    private int    index;
    private E      lastReturnedElement;
```

```
        public SetIterator ()
        {
            index = 0;
            lastReturnedElement = null;
        }

        public boolean hasNext ()
        {
            return index <= ArraySet.this.lastIndex;
        }

        public E next () throws NoSuchElementException
        {
            if (!this.hasNext ())
                throw new NoSuchElementException (
                              "end of the iteration");

            E    element = elements[index];
            lastReturnedElement = element;
            index++;

            return element;
        }

        public void remove () throws IllegalStateException
        {
            if (lastReturnedElement == null)
                throw new IllegalStateException (
                   "improper iterator state for remove operation");

            ArraySet.this.remove (lastReturnedElement);
            index--;
            lastReturnedElement = null;
        }
    }


    public Iterator<E> iterator ()
    {
        return this.new SetIterator ();
    }
}
```

Klassen `NodeSet`, implementerar gränssnittet `Set`. Mängdens element lagras i en sekvens av noder.

```
// NodeSet.java
```

```
import java.util.*;  // Iterator, NoSuchElementException
import java.io.*;    // Serializable

public class NodeSet<E> implements Set<E>, Serializable
{
    private class Node<K> implements Serializable
    {
        public K          element;
        public Node<K>    nextNode;

        public Node (K element)
        {
            this.element = element;
            this.nextNode = null;
        }
    }


    private Node<E>    firstNode;

    public NodeSet ()
    {
        firstNode = null;
    }

    public boolean isEmpty ()
    {
        return firstNode == null;
    }

    public int size ()
    {
        int     countElements = 0;
        Node<E>    node = firstNode;
        while (node != null)
        {
            countElements++;
            node = node.nextNode;
        }

        return countElements;
    }

    public boolean contains (E element)
    {
        Node<E>    node = firstNode;
        boolean    hasElement = false;
        while (!hasElement  &&  node != null)
        {
            if (element.equals (node.element))
                hasElement = true;
```

```
        else
            node = node.nextNode;
    }

    return hasElement;
}

public void add (E element)
{
    if (!this.contains (element))
    {
        Node<E>    newNode = new Node<> (element);
        newNode.nextNode = firstNode;
        firstNode = newNode;
    }
}

public void remove (E element)
{
    Node<E>    node = firstNode;
    Node<E>    previousNode = null;
    boolean    elementFound = false;
    while (!elementFound  &&  node != null)
    {
        if (element.equals (node.element))
            elementFound = true;
        else
        {
            previousNode = node;
            node = node.nextNode;
        }
    }

    if (elementFound)
    {
        if (node == firstNode)
            firstNode = firstNode.nextNode;
        else
            previousNode.nextNode = node.nextNode;
    }
}


private class SetIterator implements Iterator<E>
{
    private Node<E>    node;
    private E          lastReturnedElement;

    public SetIterator ()
    {
        node = firstNode;
```

```
            lastReturnedElement = null;
        }

        public boolean hasNext ()
        {
            return node != null;
        }

        public E next () throws NoSuchElementException
        {
            if (!this.hasNext ())
                throw new NoSuchElementException (
                                    "end of the iteration");

            E    element = node.element;
            lastReturnedElement = element;
            node = node.nextNode;

            return element;
        }

        public void remove () throws IllegalStateException
        {
            if (lastReturnedElement == null)
                throw new IllegalStateException (
                   "improper iterator state for remove operation");

            NodeSet.this.remove (lastReturnedElement);
            lastReturnedElement = null;
        }
    }

    public Iterator<E> iterator ()
    {
        return this.new SetIterator ();
    }
}
```

Gränssnittet `Set` och implementationsklasserna `ArraySet` och `NodeSet` används på följande vis:

```
// UseSet.java

import java.util.*; // Iterator
import java.io.*;   // FileOutputStream, ObjectOutputStream,
                    // FileInputStream, ObjectInputStream
import static java.lang.System.out;

class UseSet
```

```
{
    public static void main (String[] args) throws Exception
    {
        Set<String>    set = new ArraySet<> (4);
        // set = new NodeSet<> ();                              // (1)
        String[]    elements = {"A", "B", "C", "D", "E"};
        for (String e : elements)
            set.add (e);

        // mängden kan traverseras med en iterator
        Iterator<String>    iterator = set.iterator ();
        while (iterator.hasNext ())
            out.print (iterator.next () + " ");
        out.println ();


        // mängden är Serializable - kan sparas i en fil
        ObjectOutputStream    fout = new ObjectOutputStream (
                                    new FileOutputStream ("file.dat"));
        fout.writeObject (set);
        fout.close ();
        ObjectInputStream    fin = new ObjectInputStream (
                                     new FileInputStream ("file.dat"));
        set = (Set<String>) fin.readObject ();
        fin.close ();

        // mängden är Iterable - det går att använda "for each loop"
        for (String s : set)
            out.print (s + " ");
        out.println ();


        // ta bort ett element
        set.remove ("B");
        for (String s : set)
            out.print (s + " ");
        out.println ();

        // ta bort ett element med en iterator
        iterator = set.iterator ();
        String    element = iterator.next ();
        element = iterator.next ();
        iterator.remove ();
        for (String s : set)
            out.print (s + " ");
        out.println ();
    }
}
```

a) Vilken utskrift skapas när metoden `main` exekveras?

b) Vilken utskrift skapas i fall att satsen `(1)` inkluderas?

Problem 1

Långa naturliga heltal

A) En modell av delbara objekt

Objekt av vissa typer kan brytas ner i sina beståndsdelar – kan partitioneras. Till exempel kan det naturliga heltalet 165 skrivas som 5 + 60 + 100 (tre beståndsdelar: 5, 60 och 100), eller som 3 * 5 * 11 (tre beståndsdelar: 3, 5 och 11). Teckensekvensen *guld som glimmar* kan brytas ner i flera ord – *guld*, *som* och *glimmar*.

Det finns ett antal operationer som man kan utföra i samband med ett delbart objekt, oavsett dess riktiga typ. Till exempel kan man bestämma antalet delar i objektet, eller själva delarna. Dessa operationer kan definieras i ett särskilt gränssnitt, och detta gränssnitt kan implementeras i olika klasser.

Gränssnittet som representerar alla delbara objekt kan heta `Partable`, och kan definieras så här:

```
public interface Partable<T>
{
    // countParts returnerar antalet delar i objektet
    int countParts ();

    // getParts returnerar objektets delar i en vektor
    T[] getParts ();
}
```

Typparametern `T` är typ på delarana. När en teckensträng delas i delar, kan delarna vara strängar, men de kan även vara enskilda tecken. Det betyder att typen `T` inte nödvändigtvis är densamma som typen för objektet vars delar bestäms. Objektet och delarna kan vara av olika typer.

B) Hantera delbara objekt

Partitionsgränssnittet `Partable` definierar en uppsättning standardoperationer som kan utföras i samband med ett delbart objekt. Men det kan finnas även andra operationer som kan utföras i samband med ett eller flera delbara objekt. Delbara objekt kan i vissa sam-

manhang behöva hanteras på ett speciellt sätt. Man kan till exempel vilja ha alla delar i en välformaterad sträng, så här:

```
{
    5,
    60,
    100
}
```

En annan möjlighet som kan önskas är att erhålla delarna i omvänd ordning, så här:

```
[100, 60, 5]
```

Man kan skapa kod som utför olika operationer i samband med delbara objekt, och lägga den i en klass. Den kod ska gälla alla objekt vars definitionsklass implementerar gränssnittet `Partable`, oavsett riktiga typer för objekt. På så sätt kan objekt av många olika typer hanteras på ett gemensamt sätt.

Man kan skapa en klass `Partables` som hanterar objekt av typen `Partable`, till exempel så här:

```
// klassen är inte fullständig – ska kompletteras

public class Partables
{
    // toString returnerar delar av ett givet delbart objekt i en
    // välformaterad teckensträng
    public static <T> String toString (Partable<T> partable)  {}

    // partsReversed tar emot ett delbart objekt och en vektor för
    // objektets delar. Metoden fyller i vektorn med delarna i
    // omvänd ordning.
    public static <T> void partsReversed (Partable<T> partable,
                                          T[] partsReversed)  {}
}
```

Uppgift i samband med delbara objekt

Implementera metoder i klassen `Partables`. Skapa en enkel klass som implementerar gränssnittet `Partable`, och använd metoderna i samband med objekt av denna klass.

C) Modellera objektjämförelser

Objekt av vissa typer kan jämföras med varandra. Det går att undersöka om två givna objekt är likadana, och i fall att de skiljer sig kan det mindre av dem bestämmas. Objekt som kan jämföras på så sätt, kan användas i de sammanhang där jämförelser krävs. Objekten kan till exempel ordnas i en följd – sorteras. Naturliga heltal 7, 12, 1, 10, 5 kan ordnas

i en stigande följd: 1, 5, 7, 10, 12. Orden *guld*, *kunskap*, *hälsa*, *glädje* och *frid* kan sorteras enligt lexikografisk ordning: *frid*, *glädje*, *guld*, *hälsa*, *kunskap*.

Objekt som kan jämföras med varandra kan definieras i ett särskilt gränssnitt. I Javas standardbibliotek, i paketet `java.lang`, finns ett sådant gränssnitt. Gränssnittet heter `Comparable`, och definieras så här:

```
public interface Comparable<T>
{
    int compareTo (T object);
}
```

Typparametern `T` konkretiseras vanligtvis i den klass som implementerar gränssnittet. Om gränssnittet exempelvis implementeras i klassen `Punkt`, byts `T` mot `Punkt`. I metoden `compareTo` i klasssen `Punkt` jämförs den aktuella (`this`) punkten med en given punkt (argumentpunkten). Oavsett typ på de objekt som jämförs, returnerar metoden `compareTo` ett negativt heltal om det aktuella objektet är mindre, `0` i fallet att objekten är likadana, och ett positivt heltal om argumentobjektet är mindre.

Metoden `compareTo` i någon klass bestämmer hur två objekt av den klassen ska jämföras. Om objekten ska jämföras på ett annat sätt, kan en extern jämförelsemetod skapas som fyller den funktionen. I Java kan en klass som implementerar gränssnittet `java.util.Comparator` skapas. I den klassen ska metoden `compare` implementeras, och denna metod ska avgöra hur två objekt av en viss typ ska jämföras. Ett objekt av den klassen är en jämförare, som kan användas i olika sammanhang.

Om två teckensträngar exempelvis ska jämföras på något annat sätt än med metoden `compareTo`, kan en klass som implementerar gränssnittet `Comparator<String>` skapas. Denna klass kan implementeras på den plats där den behövs, som en anonym klass. Man kan göra så här:

```
Comparator<String>    comparator = new Comparator<String> ()
{
    public int compare (String s1, String s2)
    {
        return s2.compareTo (s1);
    }
};
```

Objektet (som refereras med referensen) `comparator` är en strängjämförare. Det kan användas som argument i de metoder som behöver jämföra strängar, till exempel i en sorteringsmetod. I det angivna exemplet är den sträng som är mindre enligt metoden `compareTo` större enligt metoden `compare`. Därför kan objektet `comparator` användas om strängarna ska sorteras i avtagande ordning.

D) Hantera jämförbara objekt

Det går att skriva kod som hanterar jämförbara objekt, oavsett deras riktiga typer. Om objekt av en viss typ kan jämföras, kan det minsta eller det största av dem bestämmas. Det går även att sortera objekt. Metoder som implementerar olika algoritmer med jämförbara objekt kan skapas. Parametrar i dessa metoder kan vara av typen `Comparable`, eller `Comparator`, eller av ett annat jämförelsegränssnitt.

Det går att skapa användbara sorteringsmetoder, som kan användas med objekt av olika typer. Man behöver inte skapa en särskild sorteringsmetod för varje typ av objekt. I stället kan en klass med allmänna sorteringsmetoder skapas:

```
// klassen är inte fullständig – behöver kompletteras

import java.util.Comparator;

public class Sorter
{
    // sort soreterar element i en given vektor.
    // Ordning mellan elementen bestäms med metoden compareTo i
    // elementens definitionsklass.
    public static <E extends Comparable<? super E>> void sort (
                                                E[] elements)  {}

    // sort soreterar element i en given vektor.
    // Ordning mellan elementen bestäms med en given komparator
    // (dess metod compare används för elementjämförelser).
    public static <E> void sort (E[] elements,
                                 Comparator<? super E> comp)  {}
}
```

Uppgift i samband med jämförbara objekt

Implementera metoderna i klassen `Sorter`. Prova metoderna i samband med objekt av olika standardklasser.

E) Naturliga heltal

Ett naturligt heltal består av ett antal siffror. En siffersekvens kan representeras med en teckensträng, och därför kan ett heltal skapas utifrån en välformaterad teckensträng. På så sätt kan långa naturliga heltal skapas och hanteras.

Det går att utföra olika operationer i samband med ett naturligt heltal. Ett heltal kan brytas ner i sina bestående delar: det naturliga heltalet 165 kan exempelvis skrivas som 5 + 60 + 100 (tre bestående delar). Ett naturligt heltal kan jämföras med ett annat naturligt heltal.

Talen kan vara likadana, eller ett av dem kan vara mindre. Ett heltal kan med olika aritmetiska operationer kombineras med ett annat naturligt heltal. Man kan exempelvis bestämma talens summa eller differens.

En modell av ett naturligt heltal ska skapas: man ska skapa en definitionsklass som heter `NaturalNumber`.

En modell av ett naturligt heltal – ej fullständig

```
public class NaturalNumber
    implements Comparable<NaturalNumber>, Partable<NaturalNumber>
{
    private String    number;

    public NaturalNumber (String number)
    {
        if (!validNumberFormat (number))
            throw new NumberFormatException (
                              "bad format: " + number);

        this.number = number;
        this.removeInitialZeros ();
    }

    // validNumberFormat avgör om en given sträng representerar
    // ett välformaterat naturligt heltal. I fall att formatet
    // duger returnerar metoden true, annars false.
    private static boolean validNumberFormat (String number)  {}

    // removeInitialZeros tar bort de eventuella inledande
    // nollorna från det naturliga heltalet
    private void removeInitialZeros ()  {}

    // toString returnerar det naturliga heltalet på strängform
    public String toString ()  {}

    // compareTo jämför det naturliga heltalet med ett givet
    // naturligt heltal. Metoden returnerar -1 om det aktuella
    // heltalet är mindre, 0 om heltalen är likadana, och 1 om
    // det givna heltalet är mindre.
    public int compareTo (NaturalNumber nNumber)  {}

    // equals jämför det här naturliga heltalet med ett givet
    // naturligt heltal. Metoden returnerar true om heltalen är
    // likadana, annars false.
    public boolean equals (Object nNumber)  {}

    // add returnerar summan av det här naturliga heltalet och ett
```

```
    // givet naturligt heltal.
    public NaturalNumber add (NaturalNumber nNumber)  {}

    // suntract returnerar differensen av det här naturliga
    // heltalet och ett givet naturligt heltal.
    // Det givna heltalet ska inte vara större än det aktuella
    // heltalet.
    public NaturalNumber subtract (NaturalNumber nNumber)  {}

    // countParts returnerar antalet delar i det naturliga
    // heltalet
    public int countParts ()  {}

    // getParts returnerar det naturliga heltalets delar i en
    // vektor
    public NaturalNumber[] getParts ()  {}
}
```

Uppgifter i samband med modellen

1. Implementera metoderna i klassen `NaturalNumber`.

2. Varför är metoden `validNumberFormat` definierad som `private static`?

F) Ett enkelt testprogram för klassen NaturalNumber

```
import java.io.*;    // PrintWriter
import java.util.*;  // Scanner, Arrays, Comparator

public class NaturalNumberTest
{
    public static void main (String[] args)
    {
        // utmatnings- och inmatningsverktyg
        PrintWriter    out = new PrintWriter (System.out, true);
        Scanner    in = new Scanner (System.in);

        // ett naturligt heltal och dess delar
        out.print ("natural number: ");
        out.flush ();
        String    n = in.nextLine ();
        out.println ();
        NaturalNumber    number = new NaturalNumber (n);
        out.println ("natural number: " + number);

        int    countParts = number.countParts ();
        out.println ("number of parts: " + countParts);
```

```
NaturalNumber[]    parts = number.getParts ();
out.println ("parts:");
out.println (Arrays.toString (parts));
String    partsFormated = Partables.toString (number);
out.println ("parts formated:");
out.println (partsFormated);
NaturalNumber[]    partsReversed =
                          new NaturalNumber[countParts];
Partables.partsReversed (number, partsReversed);
out.println ("parts reversed:");
out.println (Arrays.toString (partsReversed));
out.println ("\n");


// jämför två naturliga heltal
out.println ("two natural numbers: ");
String    n1 = in.next ();
String    n2 = in.next ();
in.nextLine ();
out.println ();
NaturalNumber    number1 = new NaturalNumber (n1);
NaturalNumber    number2 = new NaturalNumber (n2);

out.println ("natural numbers: " + number1 + ", "
            + number2);
boolean    eq = number1.equals (number2);
out.println (number1 + " equals " + number2 + ": " + eq);
int    comp = number1.compareTo (number2);
if (comp < 0)
    out.println (number1 + " < " + number2);
else if (comp == 0)
    out.println (number1 + " = " + number2);
else
    out.println (number1 + " > " + number2);


// heltalens summa och differens
NaturalNumber    sum = number1.add (number2);
out.println ("sum: " + sum);
if (comp >= 0)
{
    NaturalNumber difference = number1.subtract (number2);
    out.println ("difference: " + difference);
}
else
    out.println ("difference: not possible");
out.println ();


// sortera naturliga heltal
NaturalNumber[]    numbers = new NaturalNumber[10];
```

```
        out.println ("10 natural numbers:");
        for (int index = 0; index < numbers.length; index++)
            numbers[index] = new NaturalNumber (in.next ());
        in.nextLine ();
        out.println ();

        out.println ("numbers:");
        out.println (Arrays.toString (numbers));
        out.println ("numbers sorted - increasing order:");
        Sorter.sort (numbers);
        // Arrays.sort (numbers);
        out.println (Arrays.toString (numbers));
        out.println ("numbers sorted - decreasing order:");
        Comparator<NaturalNumber>    comparator =
                          new Comparator<NaturalNumber> ()
        {
            public int compare(NaturalNumber n1, NaturalNumber n2)
            {
                return n2.compareTo (n1);
            }
        };
        Sorter.sort (numbers, comparator);
        // Arrays.sort (numbers, comparator);
        out.println (Arrays.toString (numbers));
        out.println ();
    }
}
```

Uppgifter i samband med testprogrammet

1. Prova testprogrammet `NaturalNumberTest`.

2. Fundera över testprogrammet och de klasser som används i det. Hur alla delarna hänger samman? Vilka av dessa klasser är typoberoende, dvs. kan användas i samband med objekt av olika klasser? Tack vare vilka gränssnitt är dessa klasser typoberoende?

Problem 2

Teckensekvenser

A) En modell av delbara objekt

Samma som i avsnittet ” Långa naturliga heltal”.

B) Hantera delbara objekt

Samma som i avsnittet ” Långa naturliga heltal”.

C) Modellera objektjämförelser

Samma som i avsnittet ” Långa naturliga heltal”.

D) Hantera jämförbara objekt

Samma som i avsnittet ” Långa naturliga heltal”.

E) En teckensekvens

En teckensekvens består av ett antal tecken ordnade i en följd. Den har ett antal positioner – en viss kapacitet, där tecken kan lagras. Antalet positioner bestäms när en teckensekvens skapas, men kan utökas om så behövs.

De tecken som ska finnas i en teckensekvens kan specificeras när denna teckensekvens skapas. Men det går även att lägga till ytterligare tecken i slutet av teckensekvensen. Om kapaciteten inte räcker, utökas den. Mot varje tecken i en teckensekvens svarar en position: genom att specificera positionen kan man erhålla det motsvarande tecknet. Även antalet tecken i teckensekvensen kan bestämmas.

Tecken i en teckensekvens kan grupperas i enskilda ord. För detta mål används ett antal skiljetecken. Man kan använda förvalda skiljetecken, eller kan välja dem själv. Antalet ord i en teckensekvens kan bestämmas, liksom själva orden (delarna).

En teckensekvens kan jämföras med en annan teckensekvens. Teckensekvenserna kan vara likadana, eller en av dem kan vara mindre.

En modell av en teckensekvens ska skapas: man ska skapa en definitionsklass som heter `CharacterSequence`.

En modell av en teckensekvens – ej fullständig

```
public class CharacterSequence implements
     Comparble<CharacterSequence>, Partable<CharacterSequence>
{
    // objektets förvalda start-kapacitet
    public static final int    DEFAULT_INITIAL_CAPACITY = 100;
```

```
// förvalda skiljetecken mellan enskilda ord i
// teckensekvensen
public static final String    DEFAULT_DELIMITERS = " ";

// tecken i sekvensen
private char[]    characters;

// antalet tecken
private int    countCharacters;

// skiljetecken mellan enskilda delar i teckensekvensen
private String    delimiters = DEFAULT_DELIMITERS;

// CharacterSequence skapar en tom teckensekvens, med en
// förvald start-kapacitet.
public CharacterSequence ()  {}

// CharacterSequence skapar en tom teckensekvens, med en given
// start-kapacitet.
// Start-kapaciteten måste vara positiv, annars kastas ett
// undantag av typen java.lang.IllegalArgumentException.
public CharacterSequence (int initialCapacity)
                    throws IllegalArgumentException  {}

// CharacterSequence skapar en teckensekvens utifrån en given
// sträng. Teckensekvensens startkapacitet är lika med den
// förvalda kapaciteten utökad med strängens längd. Strängens
// tecken kopieras till teckensekvensen.
public CharacterSequence (String string)
{
    int    length = string.length ();
    characters = new char[DEFAULT_INITIAL_CAPACITY + length];
    countCharacters = length;
    for (int index = 0; index < length; index++)
        characters[index] = string.charAt (index);
}

// capacity returnerar teckensekvensens kapacitet
public int capacity ()  {}

// length returnerar teckensekvensen längd (antalet tecken)
public int length ()  {}

// charAt returnerar det tecken i teckensekvensen som finns på
// ett givet index. Om ett felaktigt index anges, kastas
// ett undantag av typen java.lang.IndexOutOfBoundsException.
public char charAt (int index)
             throws IndexOutOfBoundsException  {}
```

```
// toString returnerar teckensekvensen som en sträng
public String toString ()  {}

// increaseCapacity utökar teckensekvensens kapacitet.
// Den nya kapaciteten är dubbelt så stor som den tidigare
// kapaciteten.
private void increaseCapacity ()
{
    int    newCapacity = 2 * characters.length;
    char[]    newCharacters = new char[newCapacity];
    for (int index = 0; index < characters.length; index++)
        newCharacters[index] = characters[index];
    this.characters = newCharacters;
}

// append lägger till ett givet tecken till teckensekvensen
public void append (char character)
{
    if (countCharacters + 1 > characters.length)
        this.increaseCapacity ();

    characters[countCharacters++] = character;
}

// append lägger till tecken i en given vektor till
// teckensekvensen
public void append (char[] chars)  {}

// append lägger till tecken i en given sträng till
// teckensekvensen
public void append (String string)  {}

// compareTo jämför den här teckensekvensen med en given
// teckensekvens. Metoden returnerar -1 om den här
// teckensekvensen är mindre, 0 om teckensekvenserna är
// likadana, och 1 om givna teckensekvensen är mindre.
public int compareTo (CharacterSequence charSequence)  {}

// equals jämför det här teckensekvensen med en given
// teckensekvens. Metoden returnerar true om teckensekvenserna
// är likadana, annars false.
public boolean equals (Object charSequence)  {}

// setDelimiters sätter teckensekvensens skiljetecken (som
// separerar enskilda delar) till en given sträng – varje
// tecken i strängen är ett skiljetecken
public void setDelimiters (String delimiters)  {}

// countParts returnerar antalet delar (ord) i teckensekvensen
public int countParts ()  {}
```

```
    // getParts returnerar teckensekvensens delar (ord) i en
    // vektor
    public CharacterSequence[] getParts ()  {}
}
```

Uppgift i samband med modellen

Implementera konstruktorerna och metoderna i klassen `CharacterSequence`.

F) Ett enkelt testprogram CharacterSequenceTest

Programmet `CharacterSequenceTest` är ett enkelt testprogram för klassen `CharacterSequence`.

I testprogrammet skapas en teckensekvens (de nödvändiga uppgifterna matas in), varefter dess kapacitet, längd och tecken bestäms och visas. Därefter bestäms och visas antalet enskilda delar (ord) och själva delarna. Delarna visas även i form av en välformaterad teckensträng och i omvänd ordning (metoder i klassen `Partables` används).

Vidare utökas teckensekvensen på olika sätt, och resultatet följs: teckensekvensen, dess kapacitet, längd och delarna bestäms och visas efter utökningen.

Även jämförelsemetoderna provas i programmet. Först skapas och jämförs två teckensekvenser. Därefter skapas ett antal teckensekvenser (de nödvändiga uppgifterna matas in) och lagras i en vektor. Dessa teckensekvenser sorteras, både i stigande och avtagande ordning (metoder i klassen `Sorter` används), och resultatet följs.

Uppgifter i samband med testprogrammet

1. Skapa och prova testprogrammet `CharacterSequenceTest`.

2. Fundera över testprogrammet och de klasser som används i det. Hur alla delarna hänger samman? Vilka av dessa klasser är typoberoende – kan användas i samband med objekt av olika klasser? Tack vare vilka gränssnitt är dessa klasser typoberoende?

Problem 3

En abstrakt modell av en polylinje

En polylinje i planet

En polylinje är en geometrisk figur som består av en serie bundna linjesegment. Ändpunkter för dessa segment utgör polylinjens hörn. En polylinje är bestämd med sina hörn, sin färg och sin bredd. En tom polylinje saknar hörn.

En polylinjes hörn, färg, bredd och längd kan erhållas. Färgen och bredden kan ändras. En polylinjes utseende ändras även när dess hörnsekvens ändras. Det går att lägga till ett nytt hörn till polylinjen – antingen på slutet, eller framför ett hörn vars namn är givet. Det går även att ta bort ett hörn med ett givet namn.

Man kan iterera genom en polylinje: polylinjens hörn kan besökas och användas i tur och ordning.

En abstrakt modell av en polylinje ska skapas: man ska skapa ett gränssnitt som heter `Polylinje`.

En modell av en polylinje

Man underförstår att det finns en klass `Punkt`, som på ett passande sätt representerar en punkt i planet. Objekt av denna klass ska användas för att representera polylinjens hörn.

```
public interface Polylinje extends java.lang.Iterable<Punkt>
{
    Punkt[] getHorn ();

    String getFarg ();

    int getBredd ();

    double langd ();

    void setFarg (String farg);

    void setBredd (int bredd);

    public void laggTill (Punkt horn);

    void laggTillFramfor (Punkt horn, String hornNamn);

    void taBort (String hornNamn);
```

```
    java.util.Iterator<Punkt> iterator ();
}
```

Uppgifter i samband med polylinjer

1. Skapa en klass `VPolylinje` som representerar en polylinje i planet, och som implementerar gränssnittet `Polylinje`. Förutom de metoder som specificeras i gränssnittet, ska även metoden `toString` (som returnerar polylinjens strängrepresentation) implementeras. Polylinjens hörn ska lagras i en vektor av den inbyggda typen.

2. Skapa en klass `NPolylinje` som representerar en polylinje i planet, och som implementerar gränssnittet `Polylinje`. Förutom de metoder som specificeras i gränssnittet, ska även metoden `toString` (som returnerar polylinjens strängrepresentation) implementeras. Polylinjens hörn ska lagras i en sekvens av länkade noder.

Klassen `NPolylinje` ska påbörjas så här:

```
public class NPolylinje implements Polylinje
{
    private static class Nod
    {
        public Punkt    horn;
        public Nod      nastaNod;

        public Nod (Punkt horn)
        {
            this.horn = horn;
            nastaNod = null;
        }
    }

    private Nod       forstaNod;
    private String    farg = "svart";
    private int       bredd = 1;  // pixlar

    public NPolylinje ()
    {
        this.forstaNod = null;
    }

    public NPolylinje (Punkt[] horn)
    {
        if (horn.length > 0)
        {
            Nod   nod = new Nod (new Punkt (horn[0]));
            this.forstaNod = nod;
            int    pos = 1;
```

```
            while (pos < horn.length)
            {
                nod.nastaNod = new Nod (new Punkt (horn[pos++]));
                nod = nod.nastaNod;
            }
        }
    }

    // ytterligare kod här
}
```

3. Rita ett objekt av typen `NPolylinje`. Objektets nodsekvens (med motsvarande hörn) ska finnas med i ritningen.

4. Skapa ett gemensamt testprogram för klasserna `VPolylinje` och `NPolylinje`. En referens av gränssnittet `Polylinje` ska användas för att referera till objekt av de implementerade klasserna, och för att aktivera de olika metoderna. Man kan göra enligt följande mönster:

```
Polylinje     polylinje = null;
polylinje = new VPolylinje ();       // (1)
// polylinje = new NPolylinje ();    // (2)
```

Beroende på den klass som ska testas, bortkommenteras antingen satsen `(1)` eller satsen `(2)`.

5. Kan man iterera en polylinje så här:

```
for (Punkt horn : polylinje)
    System.out.println (horn);
```

Varför?

6. Skapa en statisk metod – i en särskild klass som heter `Polylinjer`, som tar emot en vektor med ett antal polylinjer av typen `Polylinje`, och returnerar den kortaste av de polylinjer i vektorn som är gula.

Använd denna metod tre gånger: i samband med en vektor med polylinjer av typen `VPolylinje`, i samband med en vektor med polylinjer av typen `NPolylinje` och i samband med en vektor som innehåller polylinjer av båda typerna.

Hur kan en vektor innehålla objekt av olika typer? Hur kan en och samma metod ta emot objekt av olika typer?

Problem 4

Markerbara listor

En markerbar lista

I ett textdokument kan enskilda ord markeras på något sätt, så att det blir lätt att märka dem. De markerade orden bär med sig en viss innebörd. Man kan använda samma idé i samband med listor, och markera enskilda element i någon lista. Dessa element skiljer sig på något sätt från andra element i listan. Man ska exempelvis kunna skilja primtal i en lista med heltal, och europeiska länder i en lista med länder från olika världsdelar.

Man kan utveckla en modell som representerar en markerbar lista. Det ska gå att markera en position i listan, att kontrollera om en viss position är markerad och att ta bort markeringen från en position. Den här funktionaliteten ska definieras i ett separat gränssnitt, och listklasserna ska sedan implementera det här gränssnittet.

Särskilda rutiner kan definieras för just markerbara listor. Till exempel kan de markerade elementen erhållas i en separat lista, eller så kan alla markeringar rensas bort.

Uppgifter i samband med markerbara listor

1. Skapa ett gränssnitt `List` som representerar en lista. Använd en typparameter för att beteckna typen för element i listan.

2. Skapa ett gränssnitt `MarkableList` som representerar en markerbar lista. Detta gränssnitt ska vara ett subgränssnitt till gränssnittet `List`.

3. Implementera gränssnittet `MarkableList` i två olika klasser: `ArrayMarkableList` och `NodeMarkableList`. I den ena klassen lagras element i en vektor av den inbyggda typen, och i den andra klassen i en sekvens av länkade noder.

4. Skapa ett gemensamt testprogram för klasserna `ArrayMarkableList` och `NodeMarkableList`.

5. Skapa en klass `MarkableLists`, vars (statiska) metoder hanterar markerbara listor på olika sätt. En av metoderna i klassen tar emot en markerbar lista, och returnerar en annan markerbar lista, som innehåller bara de element som är markerade i argumentlistan. En annan metod tar emot en markerbar lista, och rensar bort alla markeringar i den. Använd klassen `MarkableLists` på något sätt.

Problem 5

En modell av ett ord med synonymer

A) Ett ord med synonymer

Ett ord kan ha synonymer. Synonymer till ordet *mamma* är exempelvis *mor*, *moder* och *morsa*. En sådan kombination av ett ord och dess synonymer kan simpelt kallas för synord. Ett synord kan föreställas så här: *{mamma: mor, moder, morsa}*. Alla synonymer i synordet är unika: en synonym kan inte förekomma flera gånger.

Ord och synonymer som tillhör ett givet synord kan erhållas. Det gäller även antalet synonymer. Synonymerna i synordet kan besökas i tur och ordning. Man kan avgöra huruvida synordet har ett givet ord som sin synonym eller inte. Synordet kan redigeras på olika sätt: synonymer kan byta plats, nya synonymer kan läggas till, och de synonymer som redan finns kan tas bort. Två synord kan jämföras: man kan avgöra om de är likadana eller inte. Två synord är likadana om deras ord är likadana, och om varje synonym som finns i det ena synordet finns även i det andra synordet, och tvärtom.

En modell av ett synord ska skapas: man ska skapa ett gränssnitt som heter `SynOrd`.

En modell av ett ord med synonymer

```
import java.util.*;  // Iterator

public interface SynOrd extends java.lang.Iterable<String>
{
    // ord returnerar synordets ord
    String ord ();

    // antalSynonymer returnerar antalet synonymer i synordet
    int antalSynonymer ();

    // synonymer returnerar synordets synonymer
    String[] synonymer ();

    // harSynonym returnerar true om synordet har ett givet ord
    // som sin synonym, annars false.
    boolean harSynonym (String ord);

    // läggTillSynonymer lägger till givna synonymer till
    // synordet, om de inte finns
    void laggTillSynonymer (String... synonymer);

    // taBortSynonymer tar bort givna synonymer från synordet,
    // om de finns i det
```

```
    void taBortSynonymer (String... synonymer);

    // taBortAllaSynonymer tar bort alla synonymer från synordet
    void taBortAllaSynonymer ();

    // bytPlats byter plats på två givna synonymer
    void bytPlats (String synonym1, String synonym2);

    // equals returnerar true om synordet är likadant som ett
    // givet synord, annars false.
    // Två synord är likadana om och bara om deras ord och deras
    // synonymer är likadana (varje synonym i det ena synordet
    // finns även i det andra synordet, och tvärtom).
    boolean equals (Object synOrd);

    // iterator returnerar en iterator till synordet.
    // Med denna iterator kan synordets synonymer besökas i tur
    // och ordning. De besökta synonymerna kan tas bort.
    Iterator<String> iterator ();
}
```

B) Implementera gränssnittet – lagra synonymer i en vektor

```
import java.util.*;  // Iterator, NoSuchElementException

public class VSynOrd implements SynOrd
{
    // förvald kapacitet för synonymbehållaren
    public static final int    FORVALD_KAPACITET = 3;

    // ett ord och dess synonymer
    private String      ord;
    private String[]    synonymer;

    // antalet synonymer
    private int    antalSynonymer;


    // VSynOrd skapar ett synord av ett givet ord.
    // Synordet saknar synonymer.
    public VSynOrd (String ord)
    {
        this.ord = ord;
        synonymer = new String[FORVALD_KAPACITET];
        antalSynonymer = 0;
    }

    // okaKapacitetTill utökar synonymbehållarens kapacitet
```

```
// till en given kapacitet
private void okaKapacitetTill (int kapacitet)
{
    String[]    syn = new String[kapacitet];
    for (int pos = 0; pos < antalSynonymer; pos++)
        syn[pos] = synonymer[pos];
    synonymer = syn;
}

// toString returnerar synordets strängrepresentation
public String toString ()
{
    String    s = "{" + ord + ": ";
    for (int pos = 0; pos < antalSynonymer - 1; pos++)
        s = s + synonymer[pos] + ", ";
    if (antalSynonymer > 0)
        s = s + synonymer[antalSynonymer - 1];
    s = s + "}";

    return s;
}

// läggTillSynonymer lägger till givna synonymer till
// synordet, om de inte finns där tidigare
public void laggTillSynonymer (String... synonymer)
{
    for (int i = 0; i < synonymer.length; i++)
    {
        if (!this.ord.equalsIgnoreCase (synonymer[i])  &&
              !this.harSynonym (synonymer[i]))
        {
            if (antalSynonymer == this.synonymer.length)
                this.okaKapacitetTill (1 + antalSynonymer
                                + antalSynonymer / 4);
            this.synonymer[antalSynonymer++] = synonymer[i];
        }
    }
}

// equals returnerar true om synordet är likadant som ett
// givet synord, annars false.
// Två synord är likadana om och bara om deras ord och deras
// synonymer är likadana (varje synonym i det ena synordet
// finns även i det andra synordet, och tvärtom).
public boolean equals (Object synOrd)
{
    if (!(synOrd instanceof VSynOrd))
        return false;

    VSynOrd    synord = (VSynOrd) synOrd;
    boolean    lika =  this.ord.equalsIgnoreCase (synord.ord)
```

```
           &&  this.antalSynonymer () == synord.antalSynonymer ();
        if (lika)
        {
            for (int pos = 0; pos < antalSynonymer; pos++)
                if (!synord.harSynonym (synonymer[pos]))
                {
                    lika = false;
                    break;
                }
        }

        return lika;
    }
}
```

Uppgifter i samband med implementeringen av gränssnittet

1. Komplettera implementationsklassen `VSynOrd`: implementera de andra metoderna i gränssnittet.

2. Skapa ett enkelt testprogram, som använder konstruktorn och metoderna i klassen `VSynOrd`. Utforma testprogrammet så att det kan användas även i samband med andra klasser som implementerar gränssnittet `SynOrd`: använd en referens av typen `SynOrd` för att referera till de objekt som skapas. Kör testprogrammet med olika data, och kontrollera om alla metoder gör sitt jobb korrekt.

3. Skapa en bild som representerar ett objekt av typen `VSynOrd`.

4. Åskådliggör den algoritm som används i metoden `laggTillSynonymer`.

5. Ett objekt av typen `VSynOrd` har sin egen vektor, där ordets synonymer lagras. Är denna strategi minneseffektiv? Är den tidseffektiv?

C) Implementera gränssnittet – lagra synonymer i noder

```
import java.util.*;  // Iterator, NoSuchElementException

public class NSynOrd implements SynOrd
{
    // Nod representerar en nod - en behållare för en synonym
    // och en referens till nästa nod
    private class Nod
    {
```

```
    public String    synonym;
    public Nod       nastaNod;

    public Nod (String synonym)
    {
        this.synonym = synonym;
        this.nastaNod = null;
    }
}

// ordet
private String    ord;

// referensen till den första noden i nodsekvensen
private Nod    forstaNod;


// NSynOrd skapar ett synord ifrån ett givet ord.
// Synordet saknar synonymer.
public NSynOrd (String ord)
{
    this.ord = ord;
    forstaNod = null;
}

// toString returnerar synordets strängrepresentation
public String toString ()
{
    String    s = "{" + ord + ": ";
    Nod   nod = forstaNod;
    while (nod != null)
    {
        s = s + nod.synonym;
        if (nod.nastaNod != null)
            s = s + ", ";
        nod = nod.nastaNod;
    }
    s = s + "}";

    return s;
}


// taBortSynonymer tar bort givna synonymer från synordet,
// om de finns i det.
public void taBortSynonymer (String... synonymer)
{
    // ta bort synonym efter synonym
    int    pos = 0;
    Nod   nod = null;
    Nod    foregaendeNod = null;
```

```
        boolean    synonymFunnen = false;
        while (pos < synonymer.length)
        {
            nod = forstaNod;
            foregaendeNod = null;
            synonymFunnen = false;
            while (nod != null)
            {
                if (nod.synonym.equalsIgnoreCase (synonymer[pos]))
                {
                    synonymFunnen = true;
                    break;
                }

                foregaendeNod = nod;
                nod = nod.nastaNod;
            }

            if (synonymFunnen)
            {
                if (foregaendeNod == null)
                    forstaNod = nod.nastaNod;
                else
                    foregaendeNod.nastaNod = nod.nastaNod;
            }

            pos++;
        }
    }
}
```

Uppgifter i samband med implementeringen av gränssnittet

1. Komplettera implementationsklassen `NSynOrd`: implementera de andra metoderna i gränssnittet.

2. Använd det testprogram som har skapats i avsnittet B för att testa klassen `NSynOrd`.

3. Skapa en bild som representerar ett objekt av typen `NSynOrd`.

4. Åskådliggör den algoritm som används i metoden `taBortSynonymer`. Denna algoritm kan kort formuleras så här: för varje synonym som ska tas bort – ta bort den om den finns! En annan möjlig strategi är följande: för varje synonym i synordet – om den ska tas bort – ta bort den! Åskådliggör även denna strategi och implementera den.

5. Ett objekt av typen `NSynOrd` har sina egna noder, där ordets synonymer lagras. Är denna strategi minneseffektiv? Är den tidseffektiv?

D) Ett synords spektrum

Ordet *mamma* har orden *mor*, *moder* och *morsa* som sina synonymer. Detta kan föreställas i form av ett synord: *{mamma: mor, moder, morsa}*. Men detta betyder att även ordet *mor* har orden *mamma*, *moder* och *morsa* som sina synonymer. Från det första synordet kan ett annat synord härledas: *{mor: mamma, moder, morsa}*. På samma sätt kan man härleda även synorden *{moder: mamma, mor, morsa}* och *{morsa: mamma, mor, moder}*. Alla dessa synord med en och samma innebörd bildar ett spektrum av synord. Detta spektrum kan föreställas så här:

{mamma: mor, moder, morsa}
{mor: mamma, moder, morsa}
{moder: mamma, mor,morsa}
{morsa: mamma, mor, moder}

Uppgift i samband med ett synords spektrum

Skapa ett program `SynOrdSpektrum`, som matar in ett synord, och bestämmer och visar detta synords spektrum. Programmet ska ha en metod `synOrdSpektrum`, som bestämmer ett synords spektrum. Metoden ska stödja sig på gränssnittet `SynOrd` – den ska deklareras så här:

```
public static SynOrd[] synOrdSpektrum (SynOrd synOrd)
```