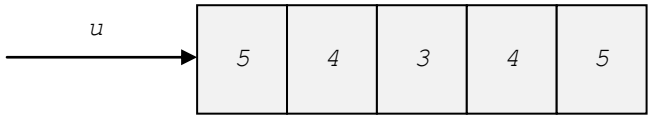
Tentamen – obligatorisk del: lösning

Uppgifter: lösningar

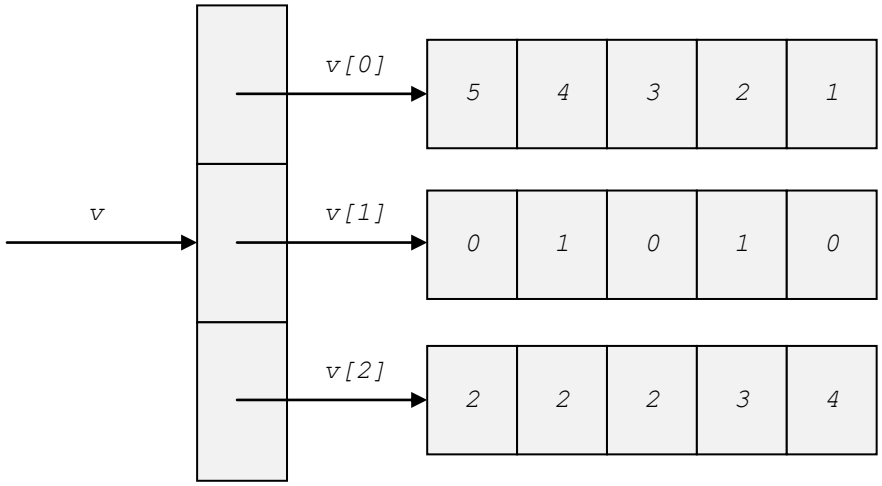
Uppgift 1 (1 poäng + 1 poäng)

a) (1 poäng)



b) (1 poäng)

Uppgift 2 (3 poäng)



Uppgift 3 (2 poäng + 1 poäng)

a) (2 poäng)

```
// vinkel tar emot en triangels sidor, och returnerar vinkeln
// (i grader) som motsvarar den första sidan.
public static double vinkel (double a, double b, double c)
{
    double    cosAlfa = (b * b + c * c - a * a) / (2 * b * c);
    double    alfa = Math.toDegrees(Math.acos(cosAlfa));
```

```
    return alfa;
}
```

b) (1 poäng)

```
double    vin = vinkel (2.5, 4.0, 3.5);
```

Uppgift 4 (2 poäng)

```
// min tar emot en icke-tom, tvådimensionell vektor med heltal,
// och returnerar det minsta heltalet i denna vektor
public static int min (int[][] v)
{
    int    m = v[0][0];
    for (int i = 0; i < v.length; i++)
        for (int j = 0; j < v[i].length; j++)
            if (v[i][j] < m)
                m = v[i][j];

    return m;
}
```

Uppgift 5 (3 poäng)

i	x_i	m	x_m
1	*7*	*1*	*7*
2	*3*	*1*	*7*
3	*2*	*1*	*7*
4	*8*	*4*	*8*
5	*9*	*5*	*9*
6	*10*	*6*	*10*
7	*4*	*6*	*10*
8	*1*	*6*	*10*

Uppgift 6 (2 poäng + 1 poäng)

a) (2 poäng)

```
// taBortInledandeNollor tar emot en icke-tom teckensträng
// av godtycklig längd, som bara innehåller siffror. I början
// av strängen kan ett antal nollor finnas, men strängen består
// inte enbart av nollor. Metoden returnerar en ny teckensträng,
// som innehåller samma siffersekvens, men utan de eventuella
// inledande nollorna.
public static String taBortInledandeNollor (String sifferSekvens)
{
    int    pos = 0;
    while (sifferSekvens.charAt (pos) == '0')
        pos++;
```

```
    String    nySifferSekvens = sifferSekvens.substring (pos);

    return nySifferSekvens;
}
```

b) (1 poäng)

```
String    sifferSekvens = "00000458009755";
String    nySifferSekvens = taBortInledandeNollor (sifferSekvens); // "458009755"
```

Uppgift 7 (3 poäng)

```
Xaaaa
Bbbb
bbbB
```

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

a) (2 poäng)

```
// toString returnerar punktens strängrepresentation
// Den returnerade strängen är på formen: [3.0, 4.0].
public String toString ()
{
     return "[" + this.x + ", " + this.y + "]";
}
```

b) (2 poäng)

```
// equals returnerar true om punkten har lika koordinater
// som en given punkt, annars false.
public boolean equals (Punkt p)
{
     return this.x == p.x  &&  this.y == p.y;
}
```

c) (2 poäng)

```
Punkt    p1 = new Punkt (1, 5);
Punkt    p2 = new Punkt (4, 2);

boolean    lika = p1.equals (p2);
String    s1 = p1.toString ();
String    s2 = p2.toString ();
```

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

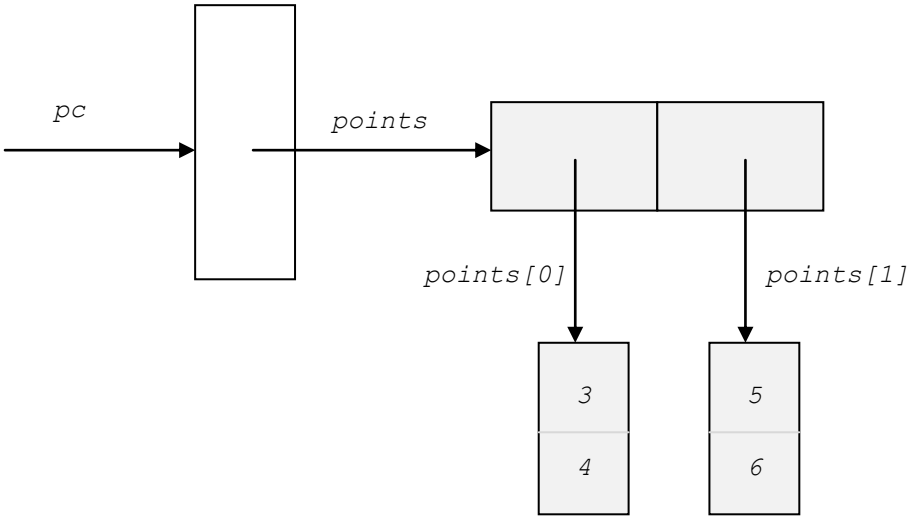
a) (2 poäng)

```
PointCollection    pc = new PointCollection ();
Point    p1 = new Point (3, 4);
Point    p2 = new Point (5, 6);
pc.add (p1);
pc.add (p2);
```

b) (2 poäng)

```
{ [x=3,y=4] [x=5,y=6] }
```

c) (2 poäng)



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

a) (1 poäng)

Klassen `java.lang.Object` är en gemensam superklass till alla Javaklasser. Därför kan referenser av typen `Object` referera till objekt av alla typer. Referenserna i vektorn `v` är av typen `Object`, och kan referera både till objekt av typen `String` och till objekt av typen `Color`.

b) (1 poäng)

```
RED
GREEN
BLUE
```

c) (1 poäng)

Referensen `v[pos]` är av typen `java.lang.Object`, och kan aktivera bara de metoder som finns med i klassen `Object`. Metoden `toString` finns där, men metoden `toLowerCase` finns inte där. Metoderna `toUpperCase` och `toLowerCase` definieras i klassen `java.lang.String`, och kan aktiveras med en referens av typen `String` (till exempel med referensen `s`, som erhålls genom en typomvandling).