Exam ID1018 19 March 2014 — required part: solutions

ID1018

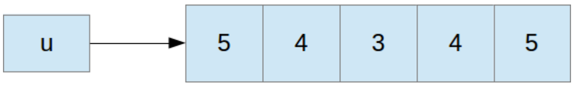
July 9, 2014

Exam — required part: solutions

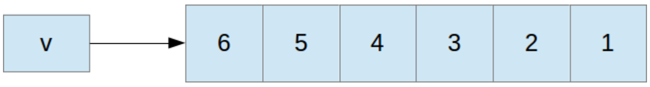
Tasks

Task 1 (1 point + 1 point)

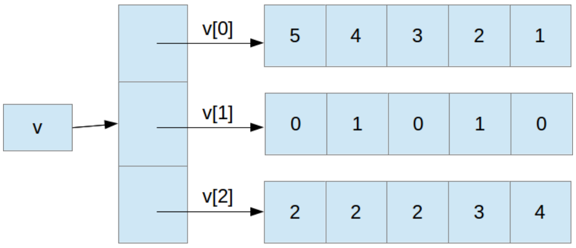
a) (1 point)



b) (1 point)



Task 2 (3 points)



Task 3 (2 points + 1 point)

a) (2 points)

```
// angle accepts the sides of a triangle and returns the
// angle (in degrees) opposite the first side.
public static double angle (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 point)

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

Task 4 (2 points)

```
// min accepts a non-empty, twodimensional vector of integers
// and returns the least integer in the vector.
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;
}
```

Task 5 (3 points)

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

Task 6 (2 points + 1 points)

a) (2 points)

```
// removeLeadingZeros accepts a non-empty character string
// of arbitrary length that only contains digits. In the
// beginning of the string there may be a run of zeros, but
// the string does not consist entirely of zeros. The
// method returns a new character string that contains the
// same digit sequence, but without the leading zeros.
public static String removeLeadingZeros (String digitSequence)
{
  int    pos = 0;
  while (digitSequence.charAt (pos) == '0')
    pos++;

  String rtnSequence = digitSequence.substring (pos);

  return rtnSequence;
}
```

b) (1 point)

```
String digitSequence = "00000458009755";
String newDigitSequence =
  removeLeadingZeros (digitSequence); // "458009755"
```

Task 7 (3 points)

```
Xaaaa
Bbbb
bbbB
```

Task 8 (2 points + 2 points + 2 points)

a) (2 points)

```
// toString returns the point's string representation.
// The returned string is on the form: [3.0, 4.0].
public String toString ()
{
        return "[" + this.x + ", " + this.y + "]";
}
```

b) (2 points)

```
// equals returns true if the point has equal coordinates
// to a given point, otherwise false.
public boolean equals (Point p)
{
        return this.x == p.x  &&  this.y == p.y;
}
```

c) (2 points)

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

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

Task 9 (2 points + 2 points + 2 points)

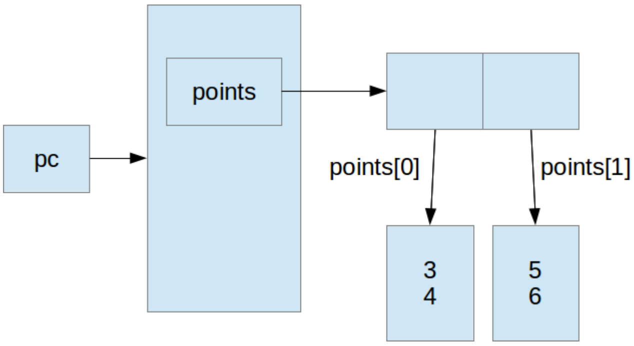
a) (2 points)

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

b) (2 points)

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

c) (2 points)



Task 10 (1 point + 1 point + 1 point)

a) (1 point)

The class `java.lang.Object` is a common superclass to all Java classes. Because of that, references of type `Object` can refer to objects of all types. The references in the vector `v` is of type `Object` and can therefore refer both to objects of type `String` and to objects of type `Color`.

b) (1 point)

```
RED
GREEN
BLUE
```

c) (1 point)

The reference `v[pos]` is of type `java.lang.Object` and can only call the methods present in class `Object`. The method `toString` is present, but the method `toLowerCase` is not. The methods `toUpperCase` and `toLowerCase` are defined in the class `java.lang.String` and can be called with a reference of type `String` (for example the reference `s` which is obtained through a type cast).