Exam ID1018 19 March 2014 — required part

ID1018

July 9, 2014

Exam — required part

Explanations

This part of the exam is required and sufficient for the grade E

To pass the exam, the student is required to pass this part of the exam. If only this part is passed, the exam grade is E. To achieve a higher grade, the student must assemble a sufficient number of points on the optional part of the exam. Depending on that number of points, the grade can be D, C, B, or A.

To pass the required part of the exam

To pass this part of the exam, the student must achieve at least two thirds of the total number of points in this part. The points in this part do not contribute to grades above E. This part is either passed or failed.

The number of points

In total: 34 points
To pass: 22 points or more

Carefully formulate answers, code, and images

Form your answers briefly and carefully.

Code is to be written so that it is easy to read and trace. In some situations suitable comments may be used to clarify. Small syntactical errors in the code can be tolerated. If parts of a code cannot be formulated precisely, well shaped pseudo-code may contribute to the solution. Do not write more code than necessary; if all that is required is a method there is no need to create a whole class. All code is to be written in Java.

When a vector or an object is drawn, it must clearly be shown what data is inside the vector or the object. When a vector or an object contains a reference, the resource that is referred to (an object or vector) must also be drawn. All references are to be relevantly labelled.

Tasks

Task 1 (1 point + 1 point)

a)

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

What does the created vector look like when this section of code has been executed? Draw both the vector and the corresponding reference.

b)

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

What does the created vector look like when this section of code has been executed? Draw both the vector and the corresponding reference.

Task 2 (3 points)

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

int     row = 0;
int     col = 0;
for (col = 0; col < v[row].length; col++)
    v[row][col] = 5 - col;
```

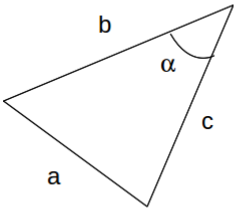
```
row = 1;
for (col = 0; col < v[row].length; col++)
    v[row][col] = col % 2;

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

Draw the created vector to show how the vector is stored in computer memory. Both the vector's cells, the corresponding data, and all references are to be included. The references shall be labelled accordingly.

Task 3 (2 points + 1 point)

The lengths of the sides of a triangle are a, b, and c. The angle that corresponds to side a is α.



The law of cosines states that:

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

a) Create a static method `angle` that accepts the lengths a, b, and c, and returns the angle α in degrees.

b) Call the `angle` method to determine the angle α in the case where $a = 2.5, b = 4.0, c = 3.5$.

Task 4 (2 points)

A static method `min` accepts a non-empty, twodimensional vector of integers, and returns the smallest integer in the vector. Create that method.

Task 5 (3 points)

An algorithm determines the greatest element in a set of integers. The algorithm can be described like this:

Algorithm: max

Preconditions:
$n \in N, n \geq 1, X = \{x_1, x_2, \ldots, x_n\} \subset Z$
(N is the set of all natural integers, Z is the set of all integers)
Postconditions:
$m \in N, 1 \leq m \leq n : x_m = \text{maximum} X$

```
max (n, X)
{
  i = 1
  m = 1
  while i < n
  {
    i++
    if ($x_1 > x_m$)
      m = i
  }

  return $x_m$
}
```

Trace this algorithm for the following set:
$X = \{7, 3, 2, 8, 9, 10, 4, 1\}$

Collect relevant data in a table of the following format:

i	x_i	m	X_m

Task 6 (2 points + 1 point)

A static method `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, for example:

`"00045763564667004357777"`

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.

a) Create the method `removeLeadingZeros`. The method is to be memory efficient: do not convert the character string to an array of char.

b) Call the method `removeLeadingZeros` in some way.

Task 7 (3 points)

```
class V
{
    public static void main (String[] args)
    {
        StringBuilder    str = new StringBuilder ("aaaa");
        str.insert (0, "X");
        System.out.println (str);

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


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

        return s;
    }
}
```

Which printout is generated when this program is executed?

Task 8 (3 points)

A class *Point* represents a planar point:

```
public class Point
{
  // The point's coordinates
  private double    x;
  private double    y;
  // Point initializes the point from the
  // given coordinates
  public Point (double x, double y)
  {
    this.x = x;
    this.y = y;
  }

  // toString returns the string representation
  // of the point on the form: [3.0, 4.0].
  // code here

  // equals returns true if this point has coordinates
  // equal to a given point, otherwise false
  // code here
}
```

a) Implement the method `toString`.

b) Implement the method `equals`.

c) Create two points of type `Point`, and then call the methods `toString` and `equals` on the two points.

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

The class *PointCollection* represents a collection of points:

```
import java.awt.Point;

class PointCollection
{
  // Points in the collection
  private Point[]    points;

  // PointCollection creates an empty collection of points
  public PointCollection ()
  {
    this.points = new Point[0];
  }

  // toString returns the string representation of the collection
  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 adds a given point to the collection
  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) Create an instance of `PointCollection` that contains the points (3,4) and (5,6).

b) Which printout is created when the instance is shown using the method System.out.println?

c) Draw the instance. All objects, vectors, and their references are to be included.

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

Review the following code:

```
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) The vector referred to by `v` stores objects of both type `String` and type `Color`. Why is this possible?

b) Which output is created when the code is executed?

c) You can include statement (1) in the code, but not statement (2) (there would be a compilation error). Why is that so?