

Exam – required part

Explanations

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

To pass the exam, the student must 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 earn a sufficient number of points from the extra part of the exam. Depending on this extra number of points, the exam grade can be D, C, B, or A.

To pass this part of the exam

To pass this required part of the exam the student must achieve at least two thirds of the total number of points in this part. These points do not contribute towards the higher grades. The highest grade in this required part is E.

Number of points

Total: 33 points

For the grade E, the requirement is at least: 22 points

Carefully formulate your answers, code, and images

Formulate your answers precise and to the point.

Code shall be written so that it is easy to follow and understand. In some situations suitable comments can contribute to understanding. Small syntactical errors can be tolerated. If some parts of code cannot be exactly produced, it is possible that well formed pseudocode may provide a solution. Do not write more code than necessary; if just a method is requested there is no need to create a whole class. All program code is to be written in Java.

When an array (vector) or an object is drawn, it must be clearly visible what data is located inside the array or object. When an array or object contains a reference, the resource that is referred to (an object or array) shall be drawn. All references shall have relevant labels.

Tasks

Task 1 (1 point + 3 points)

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

int[] v = new int[u.length];
int pos = 0;
for (int row = 0; row < u.length; row++)
{
    for (int column = 0; column < u[row].length; column++)
        v[pos] += u[row][column];
    pos++;
}
```

a) Draw the array referred to by reference u.

b) Draw the array referred to by reference v.

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

A static method `minLong` accepts four integers and returns the least of them:

```

public static int minLong (int n1, int n2, int n3, int n4)
{
    int    m = 0;
    if (n1 < n2)
    {
        if (n1 < n3)
        {
            if (n1 < n4)
                m = n1;
            else
                m = n4;
        }
        else
        {
            if (n3 < n4)
                m = n3;
            else
                m = n4;
        }
    }
    else
    {
        // code is missing here
    }

    return m;
}

```

- Complete the method `minLong`: write the missing code.
- Create another method, `minShort`, that determines the least of four given integers. Use an update strategy: start with the first integer as the least integer.
- How many comparisons are performed during an execution of method `minLong`? How many comparisons are performed during an execution of method `minShort`?

Task 3 (4 points + 3 points)

A class `Country` represents a country.

```

class Country
{
    // the country name and capital city
    private String    name;
    private String    capital;

    public Country (String name, String capital)
    {
        this.name = name;
        this.capital = capital;
    }

    public String name ()
    {
        return this.name;
    }

    public String toString ()
    {
        return "(" + this.name + ", " + this.capital + ")";
    }
}

```

A static method `filter` accepts an array of countries. The method returns another array that only contains those countries whose name begin with a given letter.

```
public static Country[] filter (Country[] countries, char firstLetter)
{
    // determine the number of acceptable countries
    // code is missing here

    // determine the acceptable countries
    // code is missing here

    // return the acceptable countries
    // code is missing here
}
```

- a) Create the method `filter`.
- b) Create an array of countries and use it in a call to method `filter`.

Task 4 (2 points + 2 points + 2 points + 2 points)

The class `Triangle` represents a planar triangle:

```
import java.awt.Point;

class Triangle
{
    // the vertices (corners) of the triangle
    private Point    vertex1;
    private Point    vertex2;
    private Point    vertex3;

    // Triangle creates a triangle from the given vertices
    public Triangle (Point[] vertices)
    {
        this.vertex1 = new Point (vertices[0]);
        this.vertex2 = new Point (vertices[1]);
        this.vertex3 = new Point (vertices[2]);
    }

    // toString returns the string representation of the triangle
    // code is missing here

    // perimeter returns the circumference of the triangle
    // code is missing here
}
```

A triangle is created and used like this:

```
Point[]    p = { new Point (0, 0), new Point (4, 0), new Point (0, 3) };
Triangle    t = new Triangle (p);
System.out.println (t);
System.out.println (t.perimeter ());
```

When this code is executed, the following printout is generated:

```
{ [x=0,y=0] [x=4,y=0] [x=0,y=3] }
12.0
```

- a) Implement the method `toString`.
- b) Implement the method `perimeter`.
- c) Draw the array referred to by reference `p`.
- d) Draw the object referred to by reference `t`.

Task 5 (2 points + 2 points + 2 points + 2 points)

The class `Circle` represents a planar circle:

```
import java.awt.Point;

class Circle
{
    // origin and radius of the circle
    private Point    centre;
    private double   radius;

    // Circle creates a circle: the origin and radius are given
    public Circle (Point centre, double radius)
    {
        this.centre = new Point (centre);
        this.radius = radius;
    }

    // toString returns the string representation of the circle
    public String toString ()
    {
        StringBuilder    sb = new StringBuilder ("[");
        sb.append (centre.toString ().substring (14) + ", ");
        sb.append (radius);
        sb.append ("]");

        return sb.toString ();
    }

    // area returns the area of the circle
    public double area ()
    {
        return Math.PI * radius * radius;
    }
}
```

The class `ColouredCircle` represents a coloured, planar circle:

```
import java.awt.Point;

class ColouredCircle extends Circle
{
    // The colour of the circle
    private String    colour;

    // ColouredCircle creates a coloured circle: the origin, radius
    // and colour are given
    // code is missing here

    // toString returns the string representation of the circle
    // code is missing here

    // getColour returns the colour of the circle
    public String getColour ()
    {
        return colour;
    }
}
```

Two circles are created and used like this:

```
Point    centre1 = new Point (3, 4);
Point    centre2 = new Point (0, 1);

Circle    c = new Circle (centre1, 2);
System.out.println (c);
// System.out.println (c.getColour ());    // (1)

ColouredCircle    cc = new ColouredCircle (centre2, 1, "yellow");
System.out.println (cc);
// System.out.println (cc.area ());    // (2)
```

When this code is executed, the following printout is generated:

```
[[x=3,y=4], 2.0]
{[[x=0,y=1], 1.0], yellow}
```

- a) Implement the constructor `ColouredCircle`.
- b) Implement the method `toString` in class `ColouredCircle`.
- c) What happens when statement (1) is included? What happens when statement (2) is included?
- d) Draw the object referred to by reference `cc`.