

Exam

Explanations

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 by which reference the array or object is referred to, and what data is located inside it. 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.

Points and grading

In total: 41 points

For grade E at least: 21 points

For grade D at least: 25 points

For grade C at least: 29 points

For grade B at least: 33 points

For grade A at least: 37 points

.

Tasks

Task 1 (2 points + 3 points)

```
int[]    u = new int[5];
for (int pos = u.length - 1; pos >= 0; pos--)
    u[pos] = (pos == u.length - 1)? 10 : u[pos + 1] - 2;

int[][]  v = new int[3][];
v[0] = new int[4];
v[1] = new int[3];
v[2] = new int[2];
for (int row = 0; row < v.length; row++)
    for (int column = 0; column < v[row].length; column++)
        v[row][column] = column;
```

a) Draw the array referred to by reference u.

b) Draw the array referred to by reference v.

Task 2 (1 point + 2 points + 2 points)

Draw the references, arrays, and objects that are created in the following cases:

a)

```
String    s1 = new String ("abcde");
String    s2 = s1;
```

b)

```
StringBuilder    sb1 = new StringBuilder ("01234");
StringBuilder    sb2 = new StringBuilder ();
sb2.append ("01234");
```

c)

```
int[]    v1 = {0, 1, 2, 3, 4};
int[]    v2 = v1;
v2[0] = 5;
```

Task 3 (3 points + 2 points + 2 points)

A class `Point` represents a planar point:

```
class Point
{
    public static final Point    ORIGIN = new Point (0, 0);

    // the coordinates of the point
    private double    x;
    private double    y;

    public Point (double x, double y)
    {
        this.x = x;
        this.y = y;
    }

    public double distance (Point p)
    {
        return Math.sqrt ( (this.x - p.x) * (this.x - p.x) +
                           (this.y - p.y) * (this.y - p.y) );
    }
}
```

A static method, `selectPoint`, accepts two points and returns the point closest to the origin:

```
public static Point selectPoint (Point p1, Point p2)
{
    // code is missing here
    // use the method distance
}
```

A static method, `selectPoint`, accepts four points and returns the point closest to the origin:

```
public static Point selectPoint (Point p1, Point p2, Point p3, Point p4)
{
    // code is missing here
    // use the method selectPoint for two points
}
```

- Create the method `selectPoint` for two points.
- Create the method `selectPoint` for four points.
- Create four points, and call method `selectPoint` with them.

Task 4 (3 points + 3 points + 3 points)

The class `Student` manages a student's name and grades.

```
class Student
{
    // the number of grades
    public static final int    GRADE_COUNT = 4;

    // the student's name and grades
    public String    name;
    public int[]    grades;
```

```

public Student (String name)
{
    this.name = name;
    grades = new int [GRADE_COUNT];
}

// setGrade sets the grade on the assessment with the given index
public void setGrade (int gradeIndex, int grade)
{
    this.grades[gradeIndex] = grade;
}

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

// averageGrade returns the student's average grade
// code is missing here
}

```

An instance of class Student is created and used like this:

```

Student student = new Student ("Anna Ericsson");
student.setGrade (0, 3);
student.setGrade (1, 4);
student.setGrade (2, 5);
student.setGrade (3, 5);

System.out.println (student);
System.out.println (student.averageGrade ());

```

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

```

{Anna Ericsson: [3, 4, 5, 5]}
4.25

```

- a) Implement the method toString.
- b) Implement the method averageGrade.
- c) Draw the object referred to by the reference student.

Task 5 (1 point + 2 points + 2 points + 1 point)

The interface LetterSupplier, and the classes LetterSupplierA, LetterSupplierB and LetterSupplierC, are defined in the following way:

```

interface LetterSupplier
{
    char lowerCaseLetter ();
    char upperCaseLetter ();
}

class LetterSupplierA implements LetterSupplier
{
    public char lowerCaseLetter ()
    {
        return 'a';
    }

    public char upperCaseLetter ()
    {
        return 'A';
    }
}

class LetterSupplierB implements LetterSupplier

```

```

{
    public char lowerCaseLetter ()
    {
        return 'b';
    }

    public char upperCaseLetter ()
    {
        return 'B';
    }
}

class LetterSupplierC extends LetterSupplierA
{
    public char upperCaseLetter ()
    {
        return 'C';
    }

    public char letter ()
    {
        return 'X';
    }
}

```

An array with objects of the classes LetterSupplierA, LetterSupplierB and LetterSupplierC is created:

```

LetterSupplier[] ls = new LetterSupplier[3];
ls[0] = new LetterSupplierA ();
ls[1] = new LetterSupplierB ();
ls[2] = new LetterSupplierC ();

```

- a) The array (referred to by reference) `ls` contains objects from different classes. Why is that possible?
- b) Which printout is generated when the following code fragment is executed?

```

for (int pos = 0; pos < ls.length; pos++)
{
    System.out.print (ls[pos].lowerCaseLetter () + " | ");
    System.out.println (ls[pos].upperCaseLetter ());
}

```

c)

```

LetterSupplierA lsa1 = new LetterSupplierC ();
System.out.println (lsa1.upperCaseLetter ());
// System.out.println (lsa1.letter ()); // (1)

```

What happens when this code fragment is executed? Why?

What happens if statement (1) is included? Why?

d)

```

LetterSupplierA lsa2 = new LetterSupplierB ();
System.out.println (lsa2.upperCaseLetter ());

```

Is this code fragment correct? Why?

Task 6 (2 points + 2 points + 5 points)

The Fibonacci numbers are:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

The integer in position n in this sequence of integers can be defined like this:

$$f(n) = 0, \quad \text{if } n = 0$$

$$f(n) = 1, \quad \text{if } n = 1$$

$$f(n) = f(n-2) + f(n-1), \quad \text{if } n > 1$$

To determine the Fibonacci number at a given position n , the following method can be used:

```
public static long fibonacciNumber (int n)
{
    long[] f = new long[n + 1];
    f[0] = 0;
    if (n > 0)
    {
        f[1] = 1;
        for (int pos = 2; pos <= n; pos++)
            f[pos] = f[pos - 2] + f[pos - 1];
    }

    return f[n];
}
```

- a) Determine the memory complexity for the algorithm used in the method `fibonacciNumber`. Categorize the corresponding complexity function: to which Θ -set does it belong?
- b) Determine the time complexity for the algorithm used in the method `fibonacciNumber`. An addition is to be regarded as an elementary operation in the algorithm. Categorize the corresponding complexity function.
- c) Create a new, more memory efficient algorithm, to determine the Fibonacci number at a given position. The algorithm shall only use a small number of memory cells. Present the algorithm in the form of a Java method. Determine the memory complexity of the algorithm, and categorize the corresponding complexity function.