

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 (2 points + 2 points)

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

```
int[]    k = {1, 2, 3, 4};
Integer[] v = new Integer[4];
for (int pos = 0; pos < v.length; pos++)
    v[pos] = new Integer (k[pos] % 2);
```

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)

If the set of real numbers is denoted R and the set of integers Z , the mathematical functions *signum*, *abs*, and *floor* can be defined like this:

signum: $R \rightarrow Z$

$x < 0$: *signum* (x) = -1

$$x = 0: \text{signum}(x) = 0$$

$$x > 0: \text{signum}(x) = 1$$

$$\text{abs}: \mathbb{R} \rightarrow \mathbb{R}$$

$$x < 0: \text{abs}(x) = -x$$

$$x \geq 0: \text{abs}(x) = x$$

$$\text{floor}: \mathbb{R} \rightarrow \mathbb{Z}$$

$$\text{floor}(x) = \max \{ m \in \mathbb{Z} \mid m \leq x \}$$

Create a class `Math`, that contains the three static methods `signum`, `abs`, and `floor`. These methods implement the corresponding mathematical functions.

Task 3 (3 points + 4 points)

A class `Word` represents a word, in both Swedish and English:

```
public class Word
{
    // the word in Swedish
    private String sWord;

    // the word in English
    private String eWord;

    public Word (String sWord, String eWord)
    {
        this.sWord = sWord;
        this.eWord = eWord;
    }

    public String wordSw ()
    {
        return this.sWord;
    }

    public String wordEn ()
    {
        return this.eWord;
    }

    public String toString ()
    {
        return this.sWord + " | " + this.eWord;
    }
}
```

A static method, `merge`, accepts two arrays of words. The method returns a third array that contains all the supplied words.

```
public static Word[] merge (Word[] words1, Word[] words2)
{
    // code is missing here
}
```

a) Create the method `merge`.

b) Create two arrays of words, and call method `merge` with them. Then show the array that is returned. The listing should look like this:

```
ett | one
två | two
tre | three
fyra | four
fem | five
sex | six
sju | seven
åtta | eight
```

Task 4 (2 points + 3 points + 2 points + 1 point)

The class `StringCreator` represents a mutable character string.

```
public class StringCreator
{
    // characters in the string
    private char[] chars;
    // the number of characters
    private int charCount;

    // StringCreator creates a character string from
    // a given character array
    public StringCreator (char[] chars)
    {
        this.chars = new char[chars.length + 1];

        // copy characters
        // code is missing here

        // adjust the charCount variable
        // code is missing here
    }

    // insert inserts a character at a given position in
    // this character string
    public void insert (char c, int pos)
        throws ArrayIndexOutOfBoundsException
    {
        if (pos > charCount)
            throw new ArrayIndexOutOfBoundsException ("bad index: " + pos);

        // code is missing here
        // do not extend the character array
    }

    public String toString ()
    {
        return new String (chars, 0, charCount);
    }
}
```

A character string is created and used like this:

```
char[] digits = {'1', '2', '3', '4'};
StringCreator sc = new StringCreator (digits);
int pos = 0;
// pos = 5; // (1)
sc.insert ('+', pos);
System.out.println (sc);
```

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

+1234

- Implement the constructor `StringCreator`.
- Implement the method `insert`.
- Draw the object referred to by reference `sc`.
- What happens if statement (1) is included in the given code section?

Task 5 (4 points + 2 points + 2 points)

The class `Shape` represents a geometrical figure:

```

public abstract class Shape
{
    // the colour of the shape
    private String    colour;

    public Shape (String colour)
    {
        this.colour = colour;
    }

    public String getColour ()
    {
        return colour;
    }

    // area returns the area of the shape
    public abstract double area ();
}

```

The classes `Rectangle` and `Circle` represent two subclasses to class `Shape`. Class `Circle` is implemented like this:

```

class Circle extends Shape
{
    // the radius of the circle
    private double    radius;

    // Circle creates a circle - the colour and radius are given
    public Circle (String colour, double radius)
    {
        super (colour);
        this. radius = radius;
    }

    // code is missing here
}

```

Several shapes are created and used like this:

```

Shape[]    shapes = new Shape[4];
shapes[0] = new Circle ("blue", 10);
shapes[1] = new Rectangle ("yellow", 3, 4);
shapes[2] = new Rectangle ("yellow", 4, 5);
shapes[3] = new Circle ("blue", 5);
// shapes[0] = new java.lang.String ("red"); // (1)

for (Shape shape : shapes)
{
    double    ar = shape.area ();
    System.out.println (shape + " : " + ar);
}

```

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

```

[blue | (10.0)] : 314.1592653589793
[yellow | (3.0, 4.0)] : 12.0
[yellow | (4.0, 5.0)] : 20.0
[blue | (5.0)] : 78.53981633974483

```

- Make complete class `Circle`: write the missing code.
- What happens when statement (1) is included? Why?
- Draw the object referred to by reference `shapes[1]`.