Review questions - Chapters 1-5

Task 1: Only one answer is correct

- (1P) In Java, variables that cannot change their value are
  a. named as consonants.
  b. actually kept constant by the linker.
  c. identified by the keyword `final`.

- (1P) Which of the following operators is a binary comparison operator?
  a. "=" Test for equality
  b. ">=" test for "greater than or equal to"
  c. "<<" test on "much bigger"

- (1P) The following applies for `switch` statements in Java:
  a. The default branch is mandatory.
  b. The `case` statement should be exited by a `break` statement, otherwise all of the following cases will be executed.
  c. `case`-targets must be floating-point literals that can be implicitly cast to `double`.

- (1P) The following applies for `if-` statements in Java:
  a. The `default` branch is mandatory.
  b. In the YES branch, only block statements are allowed, not simple statements.
  c. An existing `else-` part is always assigned to the last visible `if` that does not yet have an `else`.

Task 2: Short question - short answer (1 minute each, 5 points total)

1. Name two basic data types for floating-point numbers in Java.

float, double

2. What is the key difference between a `while` and a `do-while` loop?

While: executes instruction(s) only if condition is true.

Do-while: executes instruction(s) 1x in any case.

3. What does the signature of a method consist of?

Visibility identifier, static or non-static, return type, method name, list of Parameters

4. What does the term semantics mean in terms of programming?

Defines meaning of syntactically correct programme fragments

5. What is meant by the scope of a local variable?

The scope of a variable starts with the definition and ends with the block containing the definition.

Task 3: Troubleshooting (15 P)

The following program is intended to scale a `double`-array `f` to the value range [0; 1]. To do this, the entire array `f` is passed into the `scaling` method. The method creates a new `double`-array `res` to store the scaled values in. The `res` array is returned as a result by the `scaling` method. The array-variable `scaledValues` points to the scaled array. This is how the scaling works:

- The smallest value of the input array is a 0 in the `res` array of the scaling method.
- The largest value of the input array is 1 in the `res` array of the scaling method.
- The scaling of the other values should be linear.

In the main programme, the newly created field with the scaled values should be output; one value per line.

Find the syntactic errors and improve the semantics of the programme so that it is then an error-free Java programme that fulfills the task set. Try it with pen and paper. Do not rewrite the program!

```
public Troubleshooting {
    public static double scaling(double[] w) {
        double[] res = new double[w.length()];
        double minW = w[0];
        double maxW = w[0];

        for(int i = 1, i < w.length, i++) {
            if (w[i] > minW) {
                minW = w[i];
            } else {
                if (w[i] < maxW) {
                    maxW = w[i];
                }
            }
        }

        for(i = 0; i < w.length; i++) {
            res[i] = (w[i] - minW) / (minW - maxW);
        }
        return res;
    }

    public static void main(String args) {
        double[] f = { -50.0, 0x23, 25.0, 06, 0.7 };
        double[] scaledValues = scaling(f);
        System.out.println("\nThe scaled values are:");
        for (int i = 0; i < values.length; i++) {
            System.out.print(" " + scaledValues[i]);
        }
    }
}
```

Solution:

```
public class TroubleShootingCorrect { // keyword class

    // Method scaling
    public static double[] scaling(double[] w) {   // return-type == double[]
        double[] res = new double[w.length];    // .length instead of .length()
        // assume: Minimum/Maximum is @ first position
        double minW = w[0];
        double maxW = w[0];

        // iterate over double-Array w
        for(int i = 1; i < w.length; i++) {    // semi colons
            // new minimum?
            if (w[i] < minW) {        // (w[i] < minW) instead of (w[i] > minW)
                // store new minimum
                minW = w[i];
                // new maximum?
            } else if (w[i] > maxW) {// else if, (w[i] > maxW) instead of
(w[i]<maxW)
                // store new maximum
                maxW = w[i];
            }
        }

        // Scaling:
        // Iterate over double array w and scale it linearly
        for(int i = 0; i < w.length; i++) {  // missing int
            // store scaled values in res
            res[i] = (w[i] - minW) / (maxW - minW);// (maxW-minW) instead of (minW-
maxW)
        }
        // return res
        return res;
    }

    public static void main(String[] args) {     // String[]
        // Input array f
        double[] f = { -50.0, 0x23, 25.0, 06, 0.7 }; // 0x23 is 35
        double[] scaledValues = scaling(f);
        // Print-outs
        System.out.println("\nThe scaled values are:");
        for (int i = 0; i < scaledValues.length; i++) {//scaledValues instead of
values
            System.out.println(" " + scaledValues[i]); // .println instead of
.print
        }
    }
}
```

Task 4: Confusing, isn't it? (12P)

Specify exactly: What will be output in sequence on the console when the programme below is executed?

```
public class ConsoleOutput {

    public static boolean isRotated = false;
    public static boolean isMultiplied = false;

    public static int[] rotate(int[] a, int b) {

        int[] t = new int[a.length];

        for (int i = a.length-1; i>=0; i--) {
            t[i] = multipliziere(a[a.length-1-i],b);
        }
        isRotated = true;
        return t;
    }

    public static int multipliziere(int b, int a) {
        if (a!=1) {
            isMultiplied = true;}
        return a * b;
    }

    public static void main(String[] args) {

        int[] b = { 1000, 100, 10, 1 };
        b = rotate(b,b[b.length-1]);

        System.out.println("Multiply = " + isMultiplied);
        System.out.println("Rotate   = " + isRotated);

        for (int i = 0; i<b.length; i++) {
            System.out.println("b[" + i + "] = " + b[i]*4);
        }
    }
}
```

Solution:

```
Multiply = false
Rotate   = true
b[0] = 4
b[1] = 40
b[2] = 400
b[3] = 4000
```