Demo 6 / 20.10

Tasks that relate to each other can be done in a same file (class). If you want to carry out tasks in separate files you can call the method (subprogram in class Calculations.java for example) `mide` from another file as follows:

```
double m1 = Calculations.mide(dots);
```

An another way is to put static `import` in the beginning of the file:

```
import static demo6.Calculations.mide;
...
double m1 = mide(dots);
```

Tasks

1. **Ville:** Look for our wiki page for further instructions: https://trac.cc.jyu.fi/projects/ohj1/wiki/villeEn

2. **Real number comparison:** Real numbers are not allowed to be compared with == operator. Create two function subprograms to help the equality comparison of real numbers and these subprograms must work as follows:

```
double a = 7.1001;
double b = 7.1002;
double c = 7.2002;

if ( same(a,b,0.01) )  System.out.println("They are almost equal");
if ( !same(a,b) )      System.out.println("They are unequal");
if ( !same(a,c,0.01) ) System.out.println("They are unequal");
if ( same(a,c,0.2) )   System.out.println("More or less equal");
```

Note that the second call has less parameters than the other ones (this is called function overloading). Put that in comments first. It is advisable to create a subprogram which has a right syntax but doesn't do anything smart. An example for the subprogram with two parameters:

```
public static boolean same(double a, double b) {
  return false;
}
```

Then run the program and find out that the syntax is correct. Then alter the program to work as asked. Pay attention to the commenting and names of variables. Names that Eclipse gives aren't usually that informative.

3a. **Absolute value:** Start from scratch creating a function subprogram absolute(double number) which returns the number every time positive. Begin with proper main program in which you call the function with different test values.

3b. **Distance:** Create a function subprogram which returns the distance between two real numbers. An example:

```
distance(3.2,8.5) is almost 5.3 as well as distance(8.5,3.2);
```

4. **Loops and arrays:** Mide is the dot nearest to the average. Create a function subprogram `mide(numbers)` which returns the mide of an array of real numbers. Unfortunately you are not able to use (to call) the function `average` in the previous demo because it was made for an array of integers. Instead, you have to copy it and change the type of the array. First you have to create a function subprogram `average` which calculates and returns the average of an array of real numbers. Test with following:

```
double[] numbers = {1,2,3,2,5}; // average == 2.6
double m1 = mide(luvut);              // 3
double m2 = mide(new double[]{1});    // 1
double m3 = mide(new double[]{3,3}); // 3
double m4 = mide(new double[]{});     // 0
```

Hint: Forget the Java first and begin with doing the task with paper and pencil. Think in stages what you need to do and which auxiliary variables to use.

5a. **Scaling:** Create a function subprogram scale(number,min,max) which scales a number between [0,1] to the range between [min,max] . Examples:

```
scale(0.2,-3,3) => -1,8;
scale(0.2,1,6) => 2.0;
scale(0.0,1,6) => 1.0;
scale(1.0,1,6) => 6.0;
```

A hint: if you have a number between [0,1[and you want to get a number from it to between `[a,b[` then you need to think what to do so that 0 would become `a` and 1 would become `b`. (`f(x) = a + (b-a)*x`)

5b. **Loops and arrays:** Create a subprogram `randomNumbers(n,min,max)` which creates a size n of an array of real numbers and draws it full of numbers out of the hat (look for `Math`.random). These numbers are between `min` and `max`, but `max` is excluded: `[min,max[`. Print the array to test the subprogram.

6. **Geometry:** Think back to school or check Wikipedia for how to calculate the hypotenuse of a right-angled triangle using the Pythagorean method. Using this, figure out how to calculate the distance of two spots on a level. Then write a function

```
distance(x1,y1,x2,y2)
```

that calculates the eucleidian distance of two spots.

B1. **String management:** Write a function subprogram `askNumbers(question)` that makes the following main program (hint `StringTokenizer`, `Mjonot.erotaDouble` from Ali.jar):

```
public static void main(String[] args) {
    double numbers[] = askNumbers("Enter numbers");
    print(numbers);
}
```

work as follows:

```
Enter numbers >2 3 4 5 k      9 ;5
 2,00  3,00  4,00  5,00  0,00  9,00  5,00
```

B2. **Matrixes:** Create a function subprogram which searches for the greatest element in a two dimensional array. An example of use:

```
public static void main(String[] args) {
    double mat1[][] = {{1,2,3},{2,2,2},{4,2,3}};
    double mat2[][] = {{9,2,8},{1,2,5},{3,19,-3}};

    double greatest1 = greatest(mat1);
    double greatest2 = greatest(mat2);
    System.out.printf("%5.2f %5.2f%n",greatest1,greatest2);
}
```

B3. **Comparing real numbers:** In task 2 you may have compared numbers' absolute size. Still for example 1000 and 1100 are equal give or take 10% but not to a decimal place. As such it is often sensible to talk about comparative equality instead of absolute equality. Write one more function subprogram that works as follows:

```
nearSame(0.10,0.12,0.1) == false
nearSame(0.10,0.11,0.1) == true
nearSame(1.0,1.2,0.1)   == false
nearSame(1.0,1.1,0.1)   == true
nearSame(10,12,0.1)     == false
nearSame(10,11,0.1)     == true
nearSame(1000,1200,0.1) == false
nearSame(1000,1100,0.1) == true
```

GURU-tasks

G1-2: Plan a Gallow game. Do not realize it! Plan a console based version and think what would you print on each screen in each stage. Return as an answer a text file with illustrations what will be printed on the screen.