Homework Programming Languages, 16.11.2011

Use the book: L.D.Serbanati, C.Bogdan, "Programare orientata spre obiecte cu exemplificari in limbajul Java", vol.1, Politehnica Press, 2010!

Book, page 111, ex. 3
Use a one-dimensional array that stores 20 randomly generated numbers, each between <u>10 and 100, inclusive</u>. After a number is generated and stored in the array, it is displayed only if it is not a duplicate of a previously stored number.

<u>Generating numbers in a range:</u>

- int randomNum = Min + (int)(Math.random() * ((Max - Min) + 1));
- Random rand = new Random();
  int randomNum = rand.nextInt(max - min + 1) + min;

Attention:

- random() returns numbers between 0 (inclusive) and 1 (exclusive)
- nextInt(int n) returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Book, page 112, ex. 9
A company has four sellers (1 .. 4) selling 5 (1 .. 5) different type of products. For each different type of product sold, a seller completes a sheet. Each sheet contains:

- the seller id (1..4);
- the product id (1..5);
- total value of sales of that product in that day.

So, each seller fills in between 0 and 5 sheets per day. We assume that we have all sellers' sheets from the last month. Write a Java program which reads all this information and calculates a total, representing the value of each product sold by each seller. All these totals should be stocked in a 2-dimmensional array, named *sales*. Display the results as a table, in which each column represents a seller and each row represents a product. The total sales for each product can be obtained by going through each row of the table. The total sales for each seller can be obtained by going through each column of the table. Display these totals in a separate column (products' totals)/ separate line (sellers' totals).

Steps to create the program:

1. Create a Sheet class, with 3 instance variables (sellerId, productId, totalSale), a constructor and other useful methods (getters for the variables).

```
class Sheet
{
   // instance variable
   int sellerId;
```

```
    int productId;
    double totalSale;

    // constructor
    public Sheet(int sellerId,int productId,double totalSale)
    {
        // we have to use "this", because the params of the constructor have the same
        // name as the instance variables and the differentiation has to be done
        this. sellerId = sellerId;
        this. productId = productId;
        this.totalSale = totalSale;
    }

    //getters
    public int getProductId()
    {return productId;}

    public int getSellerId()
    {return sellerId;}

    public double getTotalSale()
    {return totalSale;}
}
```

Observation: The class Sheet doesn't have to contain the method "main".

2. Create a TestSales class, with a main method. Within the main method, do the following:
    - create an array of 20 Sheet objects (use the constructor of class Sheet!); the total sales of each product, for each seller will be generated with Math.random();
    - create the sales matrix, using the array of Sheet objects (use the getters of class Sheet!);
    - display the sales matrix: go through all the rows and columns of the matrix and display its elements;
    - display the total sales per product (in a column) and per seller (in a row): see the red marked column and row.

Sales matrix:

	Seller0	Seller1	Seller2	Seller3	Total sales per product
Product0					Total sales for Product 0
Product1					Total sales for Product 1

Product2		Total sale of product 2, sold by Seller 1			Total sales for Product 2
Product3					Total sales for Product3
Product4					Total sales for Product 4
Total sales per seller	Total sales for Seller 0	Total sales for Seller 1	Total sales for Seller 2	Total sales for Seller 3	

The structure of your java program should look like this:
<u>TestSales.java:</u>

```
package testSales;
class Sheet { } //without main method
public class TestSales { } // with main method
```

<u>Packages in JAVA:</u>

- Every class is part of some *package*.
- All classes in a file are part of the same package.
- You can specify the package using a *package declaration*:

  package *name* ;

  as the first (non-comment) line in the file.
- Multiple files can specify the same package name.
- If no package is specified, the classes in the file go into a special unnamed package (the same unnamed package for all files).
- If package *name* is specified, the file must be in a subdirectory called *name* (i.e., the directory name must match the package name).
- You can access public classes in another (named) package using:

  package-name.class-name

  You can access the public fields and methods of such classes using:

  package-name.class-name.field-or-method-name

Book, page 91, ex. 2
Implement a java class, named Product. Each product has a name and a price. Write the constructor of the class Product and the following methods: displayProduct(); getPrice() and setPrice(). Write a program which creates two products and display them, then reduce the prices of the products with 10% and display them again.

Book, page 91, ex. 3

Create a BankAccount class, with two private instance variables, named accountNumber and accountSum. These variables are initialized through the parameters of a constructor defined by the programmer. Implement public methods for adding and extracting an amount of money in/from the account. Then write a Test class, which, in the main method, creates a BankAccount object, having 40 as its number and 10 RON as its minimum value, then adds and extracts an amount of money from the account. After each method call, display the account number and the current balance, by rewriting the toString() method.