Assignment 6

Due date 27 April, 2016

In this assignment you will implement a program that consists of **three** Java classes; **BoardGame**, **Store**, and a **StoreTestApp** class to test your code.

It is especially **important** for you not to use any package definitions and to use the **exact** same definitions for classes provided below. If you have to define a package, please ask us before you submit your code. Your code will be tested automatically, so if you do not use the right method signatures and class definitions, your code will **fail** even if it works **correctly**.

Store class represents a store that buys/sells 10 different board games. It should be clear from this statement that you will use an **array** to store **the board games**.

A store has an initial budget. This budget will be used to buy some board games. A store can buy a board game as long as it can afford. Similarly, a store can sell a board game as long as it has enough stocks. For instance, you cannot sell 3 Monopoly games if you only have 2 in stock. The budget of the store should be updated after each transaction (e.g. will decrease when the store buys a board game and will increase when the store sells a board game from its store). Please take **extra** attention to array index out of bounds exception.

BoardGame class should have the following attributes (i.e. instance variables): a title, purchase price, selling price, and quantity.

Finally, you should implement a console menu for stock clients to be able to buy and sell board games. This method will be implemented in **StoreTestApp.** You can set the initial budget of a store to any amount you prefer.

1- Buy boardgame to their stock
2- Sell boardgame from their stock
3- See the current budget
4- Exit

You should implement methods of your Store and BoardGame classes according to the code below. You should figure out the return types and the types of arguments of these methods. We remind you again that **method names and signatures must be exactly the same**. For your convenience you may use store.addBoardGame part of the code in run method of your **StoreTestApp** class. However, you should start printing the menu and taking the commands in a while loop, after the last board game addition.

```
// Initial budget is set to 10,000
	Store store = new Store(10000);

	// We should add BoardGames available in our program. At the beginning
we do not have any board games at the stock; therefore, the quantity is
specified as zero.
	store.addBoardGame(new BoardGame("Monopoly", 20, 30, 0));
	store.addBoardGame(new BoardGame("Taboo", 10, 15, 0));
	store.addBoardGame(new BoardGame("King Of Tokyo", 30, 40, 0));
	store.addBoardGame(new BoardGame("Jungle Speed", 25, 28, 0));
	store.addBoardGame(new BoardGame("Stone Age", 15, 22, 0));
	store.addBoardGame(new BoardGame("Blokus", 12, 24, 0));
	store.addBoardGame(new BoardGame("Carcassone", 22, 40, 0));
	store.addBoardGame(new BoardGame("Citadel", 50, 60, 0));
	store.addBoardGame(new BoardGame("Settlers of Catan", 50, 52, 0));
	store.addBoardGame(new BoardGame("Hanabi", 10, 12, 0));

	/* In case of an error each method should return -1. e.g. array is
full size is 10 */
	int result = store.addBoardGame(new BoardGame("Bang", 15, 30, 0));
	if (result == -1)
		System.out.println("BoardGame array is full!");

	/* This method buys 2 x Monopoly, it returns -1 in case of an error,
	* otherwise it should return the total cost 20 * 2 = 40 */
	result = store.buyBoardGame(0, 2);

	/* Some test cases to consider */
	result = store.buyBoardGame(-1, 30);
	result = store.buyBoardGame(10, 30);
	result = store.buyBoardGame(2, 1000);

	/* Print the available board games. This method will display the
following information for available board games: the index of the board game
in the store, title, selling price and quantity */
	store.listBoardGames();


	/* This should sell 2 x Monopoly, it returns -1 in case of an error
	* otherwise it should return the total earned amount 30 * 2 = 60 */
	result = store.sellBoardGame(0, 2);

	/* what happens now? */
	result = store.buyBoardGame(10, 300000);

	/* Finally we should be able to get the current budget */
	int currentBudget = store.getBudget();
```

Coding Instructions:

- As mentioned above, submit your source files to the LMS **submissions with different names will be disregarded!**
- Make sure your program **compiles and runs before submitting otherwise you will get 0 from your homework (no exceptions).**
- The first lines of your code must include your name, surname, student number, and department as a comment. An example comment is as follows:

```
/* John Smith S0001 Department of Computer Science */
```

- Submit .java files only. Do NOT submit .rar, .zip, .doc, .class, etc. files.

- **IMPORTANT : Add comments to your code that briefly explains what your code does such as :**

```
int n;   // n holds the number of square
if ( n > 0)  // test whether the value of n is greater than zero
```