KAZAKH KBTU BRITISH TECHNICAL UNIVERSITY

Object-Oriented programming

Lab #4 (4 points)

#1

a) Write a program that reads student scores from the file "**scores.txt"**, stores the score in some appropriate collection (guess which one), finds the best score, and then assigns grades for all the students according to these plan:

- Grade is A if score is >= best–10;
- Grade is B if score is >= best–20;
- Grade is C if score is >= best–30;
- Grade is D if score is >= best–40;
- Grade is F otherwise.

The result must be stored in a file **"grades.txt"** in exactly the following format format:

1) Ivanov Ivan – "A"

Format of scores.txt:

Ivanov Ivan 100
Aliyev Ali 22
Menshikov Sergey 65
…

b) Now write the second program. Suppose you don't need to store students' names, all you need is just the maximum, minimum and average mark. **Think carefully, which collection needs to be used with new conditions.**

Print the answer to the same **"grades.txt"** file. Take into account the fact that you need to ALTER (update) the file, not override it, so the previously printed information must not be arisen. So, the **"grades.txt**" may look like:

1) Ivanov Ivan – "A"
…
…
Average – 60
Maximum – 100
Minimum – 25

For I/O you can use any classes, except Scanner.

#2

Develop an application for a university course offering. There are to be classes for `Textbook`, `Instructor`, and `Course`. This application should also have a `Driver` class to test the classes constructed.

Classes:

1. **`Textbook`**: this class is to have variables (fields) for isbn, title, and author(s). And, it should include a constructor, accessor and mutator methods for the fields, as well as toString and equals methods.
2. **`Instructor:`** variables to include are firstName, lastName, department, and email. Define a constructor and methods as done for Textbook.
3. **`Course:`** this class is an aggregation of data which includes a variable for courseTitle, and variables for textbook and instructor. The Course constructor would initialize the courseTitle and reference the textbook and instructor constructors to initialize fields for those classes. Include accessors and mutators and a toString, equals methods.

Driver or test program:

The Driver application should be run in 2 modes: *user mode* and *admin node*.

The user mode should offer the user choices to :

a) View the list of available courses
b) Display information about the course

For the second choice (b) , the program should be capable of retrieving values associated with course objects then printing those details.

Textbooks, courses and instructors need to be inputted to a system by admin (admin mode). Admin's username and hashed password need to be stored in a file "admin.text" . Each time admin enters the system, it should be logged to "admin.txt" (you need to alter the file, not to override). So, format of the "admin.txt" is the following:

Username: root
Password: z53h /// it is a hash!

27.10.12 13:44 admin logged in to a system
27.10.12 13:47 admin added new course "Natural language processing"
27.10.12 14:05 admin added new textbook "Data Mining – tools and applications"
…

For console input, include user prompts. Include descriptive headings or labels to identify output data.

Data for several textbook, instructor and course objects inputted by the admin need to be saved by using serialization. So, for displaying the information to users you need to deserialize it.

Developed by: Shamoi Pakita