.NET Core Training/Review

Training Exam

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.1**
Effective Date	**20/11/2012**

Hanoi, 06/2024

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	12/06/2024	Create a newly issued	Create new	ToanHK2	VinhNV

Contents

General Requirements 4
Problem 01. Employee Salary Management 5
Objectives: 5
Problem Descriptions: 5
Input: 5
Questions to answer: 6
Output: 6
Evaluation Criteria: 6
Guideline: 6
Problem 02. Student Management 7
Objectives: 7
Problem Descriptions: 7
Questions to answer: 7
Testing on Program class: 8
Evaluation Criteria: 8
Guideline: 9

CODE	**:**	**NCTR_Practice_T02**
TYPE	**:**	**Long**
LOC	**:**	**n/a**
DURATION	**:**	**180 minutes**

General Requirements

Require 01: Working tools and Delivery requirements

- **Working tools**: Visual Studio 2013 or higher.
- **Delivery**: Source code and test results in a compressed archive.

Require 02: Technologies

The product illustrates:

- C# Programming Language.
- Object Oriented Programming paradigm
- String, DateTime, Generic and Collections, Exception
- LINQ, Coding convention, Unit Test

Require 03: Technical Requirements

- Use StartCode solution to put your code in marked labels.
- Use Object-Oriented programming style.
- Apply the naming and coding convention.
- Remember to comment each class, method, and attribute (if needed).
- Clean source code and project before release

Mark scale:

- Problem 1: 4 marks
- Problem 2: 6 marks (1 mark for task 1 and 3, 2 marks for task 2 and 4)

In Visual Studio, create solution name: **NCTR.Practice.T02**

Problem 01. Employee Salary Management

Objectives:

Assess understanding of Array, Loop, Conditional Statements and using Method in C#

Problem Descriptions:

This program calculates the net income and tax for an employee based on their monthly salary and the number of dependents. The non-taxable income is 11,000,000 VND, and each dependent reduces the taxable income by 4,400,000 VND. The taxable income is calculated as total income minus non-taxable income minus the number of dependents multiplied by 4,400,000 VND. The tax rates are as follows:

Table of taxable income and tax rates		
Level	Taxable income/month(million VND)	Tax rates
1	Over 0 to 5	5%
2	Over 05 to 10	10%
3	Over 10 to 18	15%
4	Over 18 to 32	20%
5	Over 32 to 52	25%
6	Over 52 to 80	30%
7	Over 80	35%

Input:

- Allows users to enter employee name from the keyboard.

  If the user enters an empty name or null, report an error message and ask the user to re-enter.

  Message: Employee name is required!
- Allows users to enter employee salary from the keyboard

  If the user enters an invalid salary value or salary less than 0, report an error message and ask the user to re-enter.

  Message: Invalid salary! Please enter again.
- Allows the user to enter the employee's dependent number from the keyboard.

  If the user enters an invalid salary value or salary less than 0, report an error message and ask the user to re-enter.

  Message: Invalid number of dependent! Please enter again.
- Taxable income levels - The default value is already available in the starter code:

  double[] taxableIncomeLevel = {0, 5000000, 10000000, 18000000, 32000000, 52000000, 80000000};
- Tax rates corresponding to the taxable income levels (unit: %) - The default value is already available in the starter code: double[] taxLevel = {5, 10, 15, 20, 25, 30, 35};

Questions to answer:

The program utilizes two separate methods:

- ✓ CalculateTaxByMonth:
  - Takes the salary, number of dependents, taxable income levels, and tax rates as input..
  - Calculates and returns the tax to be paid for the month
- ✓ GetNetIncome:
  - Takes the salary.
  - Calculates and returns the net income after tax.

Output:

- Tax to be paid for the month.
- Net income after tax.

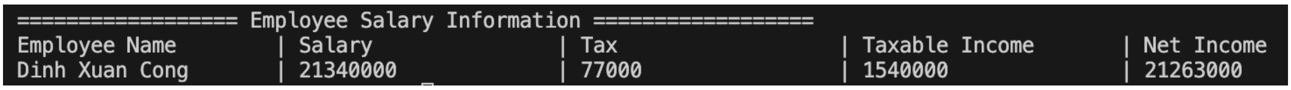
Evaluation Criteria:

- ✓ ***Correctness:***
  - The program should compile and run without errors.
  - Remember to validate user input to avoid causing errors when running the program.
- ✓ ***Functionality:***
  - The program should correctly calculate the total salary, identify employees above the threshold, and increase salaries.
- ✓ ***Formatting:***
  - The output should be neatly formatted, with columns aligned as shown in the example output.
- ✓ ***Code Quality:***
  - The code should be well-organized and modular.
  - Methods should have clear and appropriate signatures.
  - Variables and methods should have descriptive names.
  - The code should include comments explaning the logic where necessary

Guideline:

- In created solution above, create Console Application project name **NCTR.Practice.T02.Problem01**
- Complete code in the method to implement requirement
- Add comments to explain your code
- Test your method by completing the Todo task in the Main method

  Image Demo in Main method:



```
================== Employee Salary Information ==================
Employee Name        | Salary               | Tax                  | Taxable Income       | Net Income
Dinh Xuan Cong       | 21340000             | 77000                | 1540000              | 21263000
```

```
================== Employee Salary Information ==================
Employee Name        | Salary               | Tax                  | Taxable Income       | Net Income
Dinh Xuan Cong       | 21340000             | 77000                | 1540000              | 21263000
```

Problem 02. Student Management

Objectives:

- Assess understanding of Class/Object, OOP, LinQ, Exception handling
- Test students' code organization and management skills

Problem Descriptions:

This exam tests your ability to design and implement a basic Student Management System using C# and LINQ. The system will manage student information.

Questions to answer:

1. **Student Class:** *Create a class named Student with the following properties:*

- Id (int): Unique identifier for the student.
- Name (string): Name of the student.
- Email (string): Email of the student.
- DateOfBirth (date): Birth date of the student.
- Grade (double): Current grade of the student.
- Major (string): Major of the Student ("Computer Science", "Mathematics", "Physics", or "Biology").

2. **Data Validation:** *Implement data validation in the Student class to ensure:*

- Email must contain '@' and end with fpt.com.
  Create a custom exception InvalidEmailException
  Throw an *InvalidEmailException* with a clear message if validation fails.
  Message: "Email address is not correct!"
- Grade value must be between 0 and 4.

  Create a custom exception InvalidGradeException
  Throw an *InvalidGradeException* with a clear message if validation fails.
  Message: "Grade value is not valid!"
- Major is one of the allowed values ("Computer Science", "Mathematics", "Physics", or "Biology").

  Create a custom exception InvalidMajorException
  Throw an *InvalidMajorException* with a clear message if validation fails.
  Message: "Major value is not valid!"

3. StudentManagement Class:

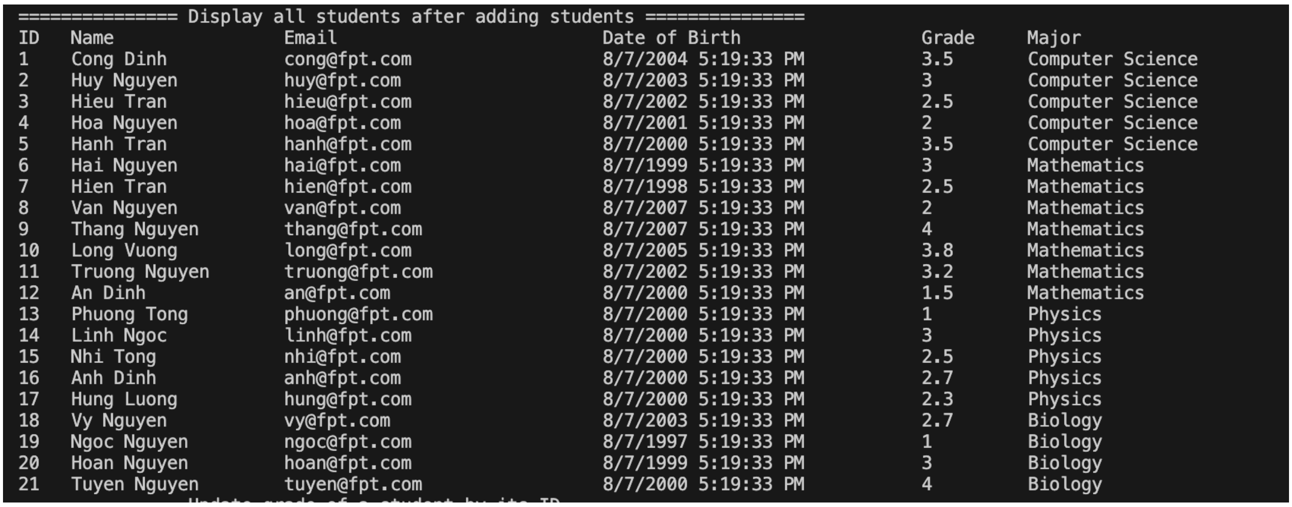
- Create an interface IStudentManagement including the following methods:
  List<Student> GetAll();
  void Add(Student student);
  void Update(int id, double grade);
  void Update(int id, Student student);
  List<Student> SearchByMajor(string major, int topN);
  List<Student> SearchByMajor(string major, double? minGrade, int pageNumber, int pageSize);

- Create a class named StudentManagement that implements IStudentManagement to manage the students list.
- In StudentManagement class, add a property named Students (List<Student>) to store the list of students.

4. **Student Management Methods:**
   - GetAll(): This method to get and return a list of all students.
   - Add(Student student): This method adds a new Student object to the Students list.
   - Update(int id, double grade): This method update grade of a student by its ID. If the student is not found, throw a *StudentNotFoundException*. Handle exceptions and print error messages when testing on Main.
   - Update(int id, Student student): This method update all properties of student by id. If the student is not found, throw a *StudentNotFoundException*. Handle exceptions and print error messages when testing on Main
   - SearchByMajor(string major, int topN): This method retrieves the top N students with the highest grades within a specified major. It returns a list of matching students based on the provided criteria
   - SearchByMajor(string major, double? minGrade, int pageNumber, int pageSize): This method offers an extended search functionality. It enables filtering students by their major, a minimum grade, pagination parameters (page number and page size) and order by grade descending. The method returns a list of students matching the specified criteria.

Testing on Program class:

1. Complete all Todo tasks in Main method.
2. Implement the display method in the Program class. The method is used to display the student list for the Todo tasks in the Main method. The student list display format is as shown below:



```
=============== Display all students after adding students ===============
ID   Name                Email                         Date of Birth                 Grade     Major
1    Cong Dinh           cong@fpt.com                  8/7/2004 5:19:33 PM           3.5       Computer Science
2    Huy Nguyen          huy@fpt.com                   8/7/2003 5:19:33 PM           3         Computer Science
3    Hieu Tran           hieu@fpt.com                  8/7/2002 5:19:33 PM           2.5       Computer Science
4    Hoa Nguyen          hoa@fpt.com                   8/7/2001 5:19:33 PM           2         Computer Science
5    Hanh Tran           hanh@fpt.com                  8/7/2000 5:19:33 PM           3.5       Computer Science
6    Hai Nguyen          hai@fpt.com                   8/7/1999 5:19:33 PM           3         Mathematics
7    Hien Tran           hien@fpt.com                  8/7/1998 5:19:33 PM           2.5       Mathematics
8    Van Nguyen          van@fpt.com                   8/7/2007 5:19:33 PM           2         Mathematics
9    Thang Nguyen        thang@fpt.com                 8/7/2007 5:19:33 PM           4         Mathematics
10   Long Vuong          long@fpt.com                  8/7/2005 5:19:33 PM           3.8       Mathematics
11   Truong Nguyen       truong@fpt.com                8/7/2002 5:19:33 PM           3.2       Mathematics
12   An Dinh             an@fpt.com                    8/7/2000 5:19:33 PM           1.5       Mathematics
13   Phuong Tong         phuong@fpt.com                8/7/2000 5:19:33 PM           1         Physics
14   Linh Ngoc           linh@fpt.com                  8/7/2000 5:19:33 PM           3         Physics
15   Nhi Tong            nhi@fpt.com                   8/7/2000 5:19:33 PM           2.5       Physics
16   Anh Dinh            anh@fpt.com                   8/7/2000 5:19:33 PM           2.7       Physics
17   Hung Luong          hung@fpt.com                  8/7/2000 5:19:33 PM           2.3       Physics
18   Vy Nguyen           vy@fpt.com                    8/7/2003 5:19:33 PM           2.7       Biology
19   Ngoc Nguyen         ngoc@fpt.com                  8/7/1997 5:19:33 PM           1         Biology
20   Hoan Nguyen         hoan@fpt.com                  8/7/1999 5:19:33 PM           3         Biology
21   Tuyen Nguyen        tuyen@fpt.com                 8/7/2000 5:19:33 PM           4         Biology
```

Evaluation Criteria:

1. Code Structure and Clarity:
   - Is the code well-organized, using appropriate classes and methods?
   - Are variable and method names descriptive and easy to understand?
   - Are there comments explaining the logic of different functionalities?

2. Data Validation:
    - Does the Student class enforce data validation for Email and Major?
    - Are appropriate exceptions thrown for invalid data?
3. Student Management Functionality:
    - Does the StudentManagement class provide methods for adding, update, and searching students?
    - Do the methods work correctly as specified?
4. LINQ Usage:
    - Are LINQ methods effectively used for search functionalities?
    - Do the search methods demonstrate filtering, sorting, and pagination capabilities (for advanced search)?
5. Error Handling:
    - Does the program handle potential errors gracefully, such as invalid inputs or student not found?
    - Are informative messages displayed for errors?

Guideline:

- In created solution above, create Console Application project name **NCTR.Practice.T02.Problem02**
- Complete code in the method to implement requirement
- Add comments to explain your code
- Test your method with test cases in Main method

-- THE END --