C# BASICS

Training Assignments

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

Hanoi, 06/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1.	01/Oct/2018	Create new	Draft		
2.	01/Jun/2019	Update template	Fsoft template	DieuNT1	

Contents

TypeScript Basic: Quiz Application 4
Objectives: 4
Prerequisites: 4
Problem Requirements: 4
Mark Scale: 7

CODE:	ANG.M.A001.Opt1
TYPE:	MEDIUM
LOC:	190
DURATION:	180 MINUTES

TypeScript Basic: Quiz Application

Objectives:

- Gain a solid understanding of TypeScript syntax, including basic types, interfaces, enums, arrays, objects, and classes.
- Learn how to develop a simple application using TypeScript, applying fundamental programming concepts to manage data and functionality.
- Practice defining and using data models to structure and organize data within an application.
- Understand the concept of services and interfaces in TypeScript to encapsulate and manage application logic and data operations.
- Explore asynchronous programming using promises to handle data retrieval and manipulation operations.
- Develop skills in testing and debugging TypeScript code, including proper error handling and logging techniques.
- Learn how to organize code into modules and classes to promote modularity, reusability, and maintainability.
- Apply fundamental programming concepts to build a basic quiz application, mimicking real-world scenarios encountered in software development.
- Practice structuring a TypeScript project, including organizing code into separate files and adding comments for clarity and documentation
- Enhance problem-solving skills by implementing proper error handling strategies to manage potential exceptions and issues.
- Learn and apply coding best practices, adhering to conventions, and writing clean, readable, and maintainable TypeScript code.

Prerequisites:

- Working environment: Visual Studio Code/Visual Studio 2013 or higher.
- Delivery: Source code packaged in a compress archive.

Problem Requirements:

This assignment challenges you to explore the fundamentals of TypeScript by building a basic quiz application. You'll practice using basic syntax, data types, arrays, objects, classes, interfaces, and modules to manage quiz data and functionality.

Function Requirements:

- **Data Model:**

  Quiz:

  - id: number
  - title: string
  - description: string (optional)
  - duration: number

- questions: Array<Question> (array of Question objects)

Question:

- id: number
- content: string (question content)
- questionType: enum (MultipleChoice,SingleChoice,TrueFalse, FillInTheBlanks, ShortAnswer, LongAnswer,)
- answers: Array<Answer> (array of Answer objects)

Answer:

- id: number
- text: string (answer text)
- isCorrect: boolean

- **Service class and interface:**

IQuizService:

- getQuizzes(): Promise<Quiz[]> (returns a promise that resolves to an array of all quizzes)
- getQuizById(id: number): Promise<Quiz | null> (returns a promise that resolves to a specific quiz by ID or null if not found)
- addQuiz(quiz: Quiz): Promise<void> (adds a new quiz to the system)
- updateQuiz(quiz: Quiz): Promise<void> (updates an existing quiz)
- deleteQuiz(id: number): Promise<void> (deletes a quiz by ID)

QuizService:

- Implement IQuizService
- Implement in-memory storage for quizzes using an array initially (consider using a more persistent storage solution in a real application)

- **Main.ts:**
  - Serves as the entry point for testing the functionalities of the QuizService. Imports the QuizService class.
  - Creates an instance of the QuizService.
  - Calls the service methods (addQuiz, getQuizzes, updateQuiz, deleteQuiz, etc.) to test their functionality.

```
congdinh@192 ANG.M.A001.Opt1 % tsc --target ES2015 --lib dom,es2015 --module CommonJS main.ts
congdinh@192 ANG.M.A001.Opt1 % node main.js
[
  {
    "id": 1,
    "title": "Capitals of Europe",
    "description": "Test your knowledge of European capitals",
    "duration": 5,
    "questions": [
      {
        "id": 1,
        "content": "What is the capital of France?",
        "questionType": 1,
        "answers": [
          {
            "id": 1,
            "text": "Paris",
            "isCorrect": true
          },
          {
            "id": 2,
            "text": "Berlin",
            "isCorrect": false
          },
          {
            "id": 3,
            "text": "Madrid",
            "isCorrect": false
          },
          {
            "id": 4,
            "text": "Rome",
            "isCorrect": false
          }
        ]
      },
      {
        "id": 2,
        "content": "What is the capital of Spain?",
        "questionType": 1,
        "answers": [
          {
            "id": 5,
            "text": "Paris",
            "isCorrect": false
          },
          {
            "id": 6,
            "text": "Berlin",
            "isCorrect": false
          },
          {
            "id": 7,
            "text": "Madrid",
            "isCorrect": true
          },
          {
            "id": 8,
            "text": "Rome",
            "isCorrect": false
          }
        ]
      }
    ]
  }
]
```

Hints:

- Utilize appropriate data types for each property in the classes.
- Leverage interfaces to define contracts for the QuizService methods.
- Implement methods within the QuizService class to handle data access and manipulation.
- Consider using helper functions for common tasks like finding a quiz by ID within the service.
- Remember to handle potential errors during data access operations (e.g., quiz not found).
- Ensure that error handling is properly implemented within each method call to manage potential exceptions or promise rejections.
- Use console.log and console.error statements to output relevant information and error messages during testing.
- Using the following code to use specific target, libraries and module:
  tsc --target ES2015 --lib dom,es2015 --module CommonJS main.ts

Business Rules:

- Ensure data consistency by maintaining the relationship between quizzes, questions, and answers.
- Implement proper error handling to gracefully handle situations like missing quizzes or invalid data.
- The QuizService should encapsulate the logic for managing quiz data, promoting modularity and reusability.

Evaluation Criteria:

- The submission should be a complete TypeScript program with well-defined classes, interfaces, and a service implementation.
- The code should demonstrate a clear understanding of basic TypeScript concepts.
- Correct usage of data types, arrays, objects, classes, interfaces, and modules will be evaluated.
- Functionality of the QuizService methods for managing quiz data will be assessed.
- Adherence to coding conventions and proper error handling will be considered.
- Organize the code into a structured project or individual files.
- Include comments to explain each code segment and its purpose.

Submission file:

- Zip solution folder to a zip file
- File: FullName_ANG_M_A01_v1.0.zip

Estimated Time: 180 minutes.

Mark Scale:

OOP design	10%	Function requirements	60%
Business rules	15%	Main function	15%