* **IQuizGame**: Là trung tâm điều khiển, gọi các phương thức để bắt đầu, gửi câu trả lời, điều hướng, và kết thúc trò chơi. Nó tương tác với IQuizState để quản lý trạng thái.
* **IQuizState**: Lưu trữ trạng thái trò chơi, bao gồm danh sách câu hỏi (questions), câu trả lời của người dùng (userAnswers), và các thông tin khác (score, startTime, v.v.).
* **IQuestion**: Định nghĩa cấu trúc và hành vi của mỗi câu hỏi, bao gồm options (lựa chọn), correctAnswer (câu trả lời đúng), và handleAnswer() (xử lý câu trả lời).
* **IOption**: Biểu diễn các lựa chọn cho câu hỏi (nếu có), ví dụ: các đáp án trong câu trắc nghiệm.
* **IAnswer**: Lưu trữ câu trả lời của người dùng, với value thuộc kiểu AnswerValue, và được lưu trong IQuizState.userAnswers.

### **Ví dụ minh họa**

Giả sử trò chơi có 3 câu hỏi:

1. Câu trắc nghiệm: "OOP là gì?" (IMultipleChoiceQuestion).
2. Câu điền chỗ trống: "console.log() là \_\_\_" (IFillInTheBlankQuestion).
3. Câu hotspot: "Chọn thuật toán thích hợp cho giải thuật sau" (IHotspotQuestion).

* **Bắt đầu**: IQuizGame.startQuiz() khởi tạo IQuizState, đặt currentQuestionIndex = 0.
* **Câu 1 (Trắc nghiệm)**:
  + Hiển thị câu hỏi và các lựa chọn (IOption[]): ["Paris", "London"].
  + Người dùng chọn "Paris", hệ thống tạo IAnswer: { questionId: "q1", value: { type: 'text', value: "Paris" } }.
  + handleAnswer() kiểm tra, thấy đúng, tăng score và lưu vào userAnswers.
* **Chuyển tiếp**: Người dùng gọi nextQuestion(), currentQuestionIndex tăng lên 1.
* **Câu 2 (Điền chỗ trống)**:
  + Hiển thị ô nhập liệu, người dùng nhập "Jupiter".
  + Tạo IAnswer: { questionId: "q2", value: { type: 'text', value: "Jupiter" } }.
  + handleAnswer() kiểm tra, đúng, tăng score.
* **Câu 3 (Hotspot)**:
  + Hiển thị bản đồ, người dùng nhấp vào tọa độ { x: 100, y: 200 }.
  + Tạo IAnswer: { questionId: "q3", value: { type: 'coordinates', value: { x: 100, y: 200 } } }.
  + handleAnswer() kiểm tra, đúng, tăng score.
* **Kết thúc**: Người dùng gọi endQuiz(), tính điểm cuối (3/3), hiển thị kết quả.

### 1. **Core Types**

These foundational types define the data structures used across the quiz system.

* AnswerValue
  + **Purpose**: Represents the value of a user's answer for different question types.
  + **Structure**: A discriminated union that supports various answer formats:
    - text: For questions like multiple-choice, fill-in-the-blank, dropdown, survey, or math input (e.g., { type: "text", value: "A" }).
    - number: For numeric answers.
    - boolean: For true/false answers.
    - array-reorder: For reorder questions (e.g., { type: "array-reorder", value: ["item1", "item2"] }).
    - array-drag-and-drop: For drag-and-drop questions (e.g., { type: "array-drag-and-drop", value: [{ itemId: "1", targetId: "A" }]).
    - array-categorize: For classify questions (e.g., { type: "array-categorize", value: [{ itemId: "1", categoryId: "cat1" }]).
    - array-labeling: For image tagging (e.g., { type: "array-labeling", value: [{ labelId: "1", position: { x: 10, y: 20 } }]).
    - array-match: For matching questions (e.g., { type: "array-match", value: [{ left: "A", right: "1" }]).
    - array-graphing: For graph plotting (e.g., { type: "array-graphing", value: [{ x: 1, y: 2 }]).
    - coordinates: For image hotspot (e.g., { type: "coordinates", value: { x: 50, y: 60 } }).
  + **Usage**: Ensures answers are structured consistently for each question type, enabling automatic grading for objective questions and data collection for surveys.
* OptionValue
  + **Purpose**: Defines the value of an option in a question (e.g., a choice in multiple-choice).
  + **Structure**: Supports text, number, or coordinates (e.g., { type: "text", value: "Option A" }).
  + **Usage**: Used in questions with selectable options, like multiple-choice or matching.
* LocalizedText
  + **Purpose**: Supports multilingual text for questions, options, or hints.
  + **Structure**: Either a string or an object mapping languages to strings (e.g., { "en": "Hello", "vi": "Xin chào" }).
  + **Usage**: Ensures the quiz is adaptable to different languages.

### 2. **Enums**

These define fixed sets of values for question types, answer types, difficulty levels, and graph types, ensuring consistency with the backend JSON.

* QuestionType
  + Lists all supported question types, matching the code field in the JSON (e.g., MULTI\_CHOICE, SURVEY, GRAPH\_PLOTTING).
  + Only includes objective (answer\_type: "objective") and survey (answer\_type: "non\_evaluated") types, excluding subjective types like video/audio response.
* AnswerType
  + Defines grading types: OBJECTIVE (automatic grading) and NON\_EVALUATED (survey, no right/wrong).
  + Matches the answer\_type field in the JSON.
* DifficultyLevel
  + Defines difficulty levels: EASY, MEDIUM, HARD.
  + Used in question metadata for categorization.
* GraphType
  + Defines graph types for graph plotting questions: LINE, BAR, SCATTER.

### 3. **Interfaces**

These define the structure for quiz components, from answers to the overall game logic.

* IResult
  + **Purpose**: Standardizes the result of operations (e.g., submitting an answer).
  + **Structure**: { success: boolean, error?: string }.
  + **Usage**: Provides feedback on operations, like whether an answer submission was valid.
* IAnswerValidator
  + **Purpose**: Defines a method to validate user answers against correct answers for objective questions.
  + **Structure**: Has a validate method that takes userAnswer and correctAnswer (both IAnswer) and returns { isCorrect: boolean, score: number }.
  + **Usage**: Used for automatic grading in objective questions (e.g., checking if a multiple-choice answer matches the correct option).
* IOption
  + **Purpose**: Represents a single option in a question (e.g., a choice in multiple-choice).
  + **Structure**: { id: string, value: OptionValue, label?: LocalizedText }.
  + **Usage**: Used in questions like multiple-choice, matching, or dropdown to present selectable options.
* IAnswer
  + **Purpose**: Represents a user's answer to a question.
  + **Structure**:
    - questionId: Links the answer to a specific question.
    - value: The answer content (type AnswerValue).
    - isCorrect: Indicates if the answer is correct (for objective questions; false for surveys).
    - submittedAt?: Timestamp of submission.
    - timeSpent?: Time spent answering (in seconds).
  + **Usage**: Stores user responses and supports grading or data collection.
* IQuestion<T extends AnswerValue>
  + **Purpose**: Base interface for all question types, using generics to enforce specific answer types.
  + **Structure**:
    - id: Unique question identifier.
    - type: Question type (from QuestionType).
    - answerType: Grading type (OBJECTIVE or NON\_EVALUATED).
    - text: Question text (supports localization).
    - options?: List of selectable options (if applicable).
    - correctAnswer: The predefined correct answer (for objective questions; may be a dummy for surveys).
    - validator?: Optional validator for objective questions.
    - metadata?: Includes difficulty, points, time limit, tags, and hints.
    - handleAnswer: Method to process a submitted answer, returning IResult.
  + **Usage**: Provides a flexible structure for all question types, ensuring type-safe answer handling.
* IQuizState
  + **Purpose**: Tracks the state of a quiz session.
  + **Structure**:
    - currentQuestionIndex: Current question being answered.
    - score: User's current score (for objective questions).
    - userAnswers: Map of question IDs to user answers.
    - questions: Array of questions (type IQuestion<AnswerValue>).
    - questionLoader?: Optional async loader for large quizzes.
    - isCompleted: Indicates if the quiz is finished.
    - startTime and endTime?: Tracks quiz duration.
    - metadata?: Includes quiz ID, user ID, and total points possible.
  + **Usage**: Maintains the quiz's progress and state.
* IQuizGame
  + **Purpose**: Defines the logic for managing a quiz.
  + **Structure**: Methods like:
    - startQuiz, endQuiz: Begin or end the quiz.
    - submitAnswer: Submit an answer for a question.
    - nextQuestion, previousQuestion, skipQuestion: Navigate questions.
    - calculateScore: Compute the score (for objective questions).
    - getCurrentQuestion: Retrieve the current question.
    - pauseQuiz, resumeQuiz: Pause/resume the quiz.
    - shuffleQuestions: Randomize question order.
    - getHint: Retrieve a question's hint.
  + **Usage**: Provides the core functionality for running a quiz.

### 4. **Specific Question Interfaces**

Each question type extends IQuestion with specific properties tailored to its format. All match the JSON structure and support either objective or non\_evaluated answer types.

* IMultipleChoiceQuestion
  + For MULTI\_CHOICE (objective).
  + Has options (list of choices) and a text answer (e.g., selecting "A").
* IFillInTheBlankQuestion
  + For FILL\_IN\_THE\_BLANK (objective).
  + Expects a text answer matching the correct answer.
* IReadingComprehensionQuestion
  + For READING\_COMPREHENSION (objective).
  + Includes a passage and subQuestions (other question types).
* IMatchingQuestion
  + For MATCHING (objective).
  + Has pairs (left/right options) and an array-match answer.
* IReorderQuestion
  + For REORDER (objective).
  + Has items to reorder and an array-reorder answer.
* IDragAndDropQuestion
  + For DRAG\_N\_DROP (objective).
  + Has items and targets, with an array-drag-and-drop answer.
* IDropDownQuestion
  + For DROPDOWN (objective).
  + Has options and a text answer.
* IImageHotspotQuestion
  + For IMAGE\_HOTSPOT (objective).
  + Has an imageUrl, hotspots, and a coordinates answer.
* IImageTaggingQuestion
  + For IMAGE\_TAGGING (objective).
  + Has an imageUrl, labels, and an array-labeling answer.
* IClassifyQuestion
  + For CLASSIFY (objective).
  + Has items, categories, and an array-categorize answer.
* ISurveyQuestion
  + For SURVEY (non\_evaluated).
  + Has options and a text answer; no correct/incorrect evaluation.
* IMathInputQuestion
  + For MATH\_INPUT (objective).
  + Has an equation and a text answer (math notation).
* IGraphPlottingQuestion
  + For GRAPH\_PLOTTING (objective).
  + Has a graphType, dataPoints, and an array-graphing answer.

### 5. **Key Changes and Alignment with Requirements**

* **Objective and Survey Only**: Removed subjective question types (e.g., VIDEO\_RESPONSE, AUDIO\_RESPONSE) and their interfaces, keeping only types from the JSON with answer\_type: "objective" or "non\_evaluated".
* **JSON Alignment**: QuestionType enum values match the code field in the JSON (e.g., MULTI\_CHOICE instead of MULTIPLE\_CHOICE). answerType field in IQuestion uses AnswerType enum to reflect objective or non\_evaluated.
* **No** any **Usage**: Replaced any in IQuizState and IQuizGame with AnswerValue, ensuring type safety.
* **Backend Mapping**: Naming and structure (e.g., code, answer\_type) align with the JSON for seamless backend integration.

### 6. **How It Works Together**

* **Quiz Flow**: The IQuizGame manages the quiz, using IQuizState to track progress. Questions (IQuestion) are loaded, and users submit answers (IAnswer) via submitAnswer.
* **Objective Grading**: For objective questions, IAnswerValidator checks answers against correctAnswer, updating the score in IQuizState.
* **Survey Handling**: Survey questions collect answers without grading, storing them in userAnswers for statistical analysis.
* **Type Safety**: Generics in IQuestion<T> ensure each question type has the correct AnswerValue type, preventing errors.