**1. Quy ước đặt tên (Naming Conventions)**

* **PascalCase** cho tên lớp, phương thức, và thuộc tính:

csharp

Copy

public class StockService { }

public void CalculateStockPrice() { }

public decimal CurrentPrice { get; set; }

* **camelCase** cho tên biến và tham số:

csharp

Copy

int totalStock = 100;

public void UpdateStock(int newStock) { }

* **UPPER\_CASE** cho hằng số:

csharp

Copy

public const int MAX\_STOCK = 1000;

* **Tiền tố "I"** cho tên interface:

csharp

Copy

public interface IStockRepository { }

* **Tên có ý nghĩa**:
  + Tránh đặt tên chung chung như x, data, temp.
  + Ví dụ:

csharp

Copy

// Tốt

int numberOfShares;

decimal stockPrice;

// Không tốt

int n;

decimal p;

**2. Quy ước định dạng code (Formatting Conventions)**

* **Thụt lề (Indentation)**:
  + Sử dụng 4 khoảng trắng (không dùng tab).
  + Ví dụ:

csharp

Copy

if (condition)

{

// Code here

}

* **Khoảng trắng**:
  + Thêm khoảng trắng xung quanh toán tử và sau dấu phẩy.
  + Ví dụ:

csharp

Copy

int result = a + b;

UpdateStock(stockId, newStock);

* **Xuống dòng**:
  + Giữ các dòng code ngắn (không quá 120 ký tự).
  + Xuống dòng sau dấu phẩy hoặc toán tử khi cần.
  + Ví dụ:

csharp

Copy

var stockList = new List<Stock>

{

new Stock { Symbol = "AAPL", Price = 150.50m },

new Stock { Symbol = "GOOGL", Price = 2800.00m }

};

**3. Quy ước viết code (Coding Practices)**

* **Sử dụng var khi kiểu dữ liệu rõ ràng**:

csharp

Copy

var stock = new Stock();

* **Tránh lồng quá nhiều câu lệnh if**:
  + Sử dụng guard clauses hoặc early return để giảm độ phức tạp.
  + Ví dụ:

csharp

Copy

public void ProcessStock(Stock stock)

{

if (stock == null)

throw new ArgumentNullException(nameof(stock));

// Logic chính

}

* **Sử dụng string interpolation thay vì string.Format**:

csharp

Copy

string message = $"Stock {stock.Symbol} has price {stock.Price}";

* **Tránh "magic numbers"**:
  + Sử dụng hằng số hoặc biến có ý nghĩa thay vì số trực tiếp.
  + Ví dụ:

csharp

Copy

const int MaxStock = 1000;

if (stock > MaxStock) { }

**4. Quy ước về comment và tài liệu**

* **Comment có ý nghĩa**:
  + Giải thích "tại sao" thay vì "cái gì".
  + Ví dụ:

csharp

Copy

// Tính toán giá cổ phiếu dựa trên công thức Black-Scholes

decimal price = CalculateOptionPrice();

* **XML Documentation** cho các phương thức và lớp:

csharp

Copy

/// <summary>

/// Cập nhật số lượng cổ phiếu.

/// </summary>

/// <param name="newStock">Số lượng cổ phiếu mới.</param>

public void UpdateStock(int newStock) { }

**5. Quy ước về kiến trúc và tổ chức code**

* **Phân tách lớp và phương thức theo nguyên tắc Single Responsibility**:
  + Mỗi lớp/phương thức chỉ nên có một trách nhiệm duy nhất.
  + Ví dụ:

csharp

Copy

public class StockService

{

public decimal CalculatePrice() { }

public void SaveStock(Stock stock) { }

}

* **Sử dụng Dependency Injection (DI)**:
  + Tránh tạo đối tượng trực tiếp trong lớp, thay vào đó sử dụng DI.
  + Ví dụ:

csharp

Copy

public class StockService

{

private readonly IStockRepository \_repository;

public StockService(IStockRepository repository)

{

\_repository = repository;

}

}

**6. Quy ước về xử lý lỗi**

* **Sử dụng try-catch một cách hợp lý**:
  + Chỉ bắt các ngoại lệ mà bạn có thể xử lý.
  + Ví dụ:

csharp

Copy

try

{

// Code có thể gây lỗi

}

catch (SpecificException ex)

{

// Xử lý lỗi

}

* **Sử dụng throw thay vì throw ex**:

csharp

Copy

catch (Exception ex)

{

// Ghi log

throw; // Giữ nguyên stack trace

}

**7. Công cụ hỗ trợ**

* **StyleCop**:
  + Công cụ phân tích code để đảm bảo tuân thủ quy ước.
  + Cấu hình trong .editorconfig hoặc StyleCop.json.
* **ReSharper**:
  + Công cụ mạnh mẽ để refactor và kiểm tra code.
* **SonarQube**:
  + Phân tích chất lượng code và đưa ra gợi ý cải thiện.

**8. Ví dụ tổng hợp**

csharp

Copy

public class StockService

{

private readonly IStockRepository \_repository;

public StockService(IStockRepository repository)

{

\_repository = repository;

}

/// <summary>

/// Tính toán giá cổ phiếu.

/// </summary>

/// <param name="symbol">Mã cổ phiếu.</param>

/// <returns>Giá cổ phiếu.</returns>

public decimal CalculateStockPrice(string symbol)

{

if (string.IsNullOrEmpty(symbol))

throw new ArgumentNullException(nameof(symbol));

var stock = \_repository.GetStockBySymbol(symbol);

return stock?.Price ?? 0;

}

}