C# New Features Coding Test

You are tasked with building a decoupled service for a Human Resources application that calculates an employee's total compensation, which is the sum of their base salary and a dynamic bonus.

Your solution must use a dedicated `EmployeeRecord` class for data transport and integrate the four required features.

Requirements and Feature Integration

1. **Model Definition (Automatic Properties):**

   Define a public class named **`EmployeeRecord`** with the following public **automatic properties**:

   - `Id` (int)
   - `Name` (string)
   - `Role` (string)
   - `IsVeteran` (bool)

2. **Dependency Injection (DI) Setup:**

   - Define an interface `IEmployeeDataReader`.
   - The `IEmployeeDataReader` must define a method `EmployeeRecord GetEmployeeRecord(int employeeId)` that returns an instance of the class defined in Requirement 1.
   - Define a concrete class `MockEmployeeDataReader` that implements this interface, returning a hardcoded `EmployeeRecord` for a given ID (e.g., ID 102 returns a Manager record).
   - The primary service, **`PayrollProcessor`**, must receive an instance of `IEmployeeDataReader` via **Constructor Injection**.

3. **Dictionary Initializer:**

   - The `PayrollProcessor` class must contain a private, static, read-only field `BaseSalaries` of type `Dictionary<int, decimal>`. This dictionary must be initialized inline using the **Dictionary Initializer** feature with at least three sample employee ID-to-salary pairs (e.g., `{ [101] = 65000m, [102] = 120000m }`).

4. **Pattern Matching Logic:**

    - Implement a method `CalculateTotalCompensation(int employeeId)` in `PayrollProcessor`.
    - Inside this method, retrieve the `EmployeeRecord` using the injected data reader.
    - Use the **C# Pattern Matching `switch` expression** (C# 8+) combined with a **Property Pattern** to apply a dynamic bonus based on the employee's **`Role`** and **`IsVeteran`** properties, following this logic:

Property Pattern Match	Bonus (Applied)
{ Role: "Manager", IsVeteran: true }	**$10,000**
{ Role: "Manager", IsVeteran: false }	**$5,000**
{ Role: "Developer" }	**$2,000**
{ Role: "Intern" }	**$500**
_ (Discard pattern for all others)	**$0**

5. **Final Output:**
    1. The `CalculateTotalCompensation` method must return the calculated total compensation (`decimal`), which is the `BaseSalaries` lookup combined with the bonus amount determined by Pattern Matching.

Deliverables

Provide the complete C# code demonstrating all components, including the new `EmployeeRecord` class, and a working demonstration in the `Main` method.