Full .NET Interview Preparation Solutions

1. Interface & Inheritance

```
public interface IEmployee
{
    string Name { get; set; }
    decimal CalculateSalary();
}
public class FullTimeEmployee : IEmployee
{
    public string Name { get; set; }
    public decimal MonthlySalary { get; set; }
    public decimal CalculateSalary() => MonthlySalary;
}
public class PartTimeEmployee : IEmployee
{
    public string Name { get; set; }
    public int HoursWorked { get; set; }
    public decimal HourlyRate { get; set; }
    public decimal CalculateSalary() => HoursWorked * HourlyRate;
}
```

2. Multithreading: Even and Odd

```
static int count = 1;
static readonly object lockObj = new();
static void PrintOdd() {
    while (count <= 100) {
        lock (lockObj) {
            if (count % 2 != 0) Console.WriteLine("Odd: " + count++);
        }
    }
}
static void PrintEven() {
    while (count <= 100) {
        lock (lockObj) {
            if (count % 2 == 0) Console.WriteLine("Even: " + count++);
        }
    }
}
```

3. LINQ Challenge

```
var oldest = employees.OrderByDescending(e => e.Age).First();
var groupByDept = employees.GroupBy(e => e.Department);
var under30 = employees.Where(e => e.Age < 30).OrderBy(e => e.Age);
```

4. Events and Delegates

```
public class AlarmClock {
    public event Action OnAlarmRings;
    public void RingAlarm() {
        Console.WriteLine("Alarm ringing...");
        OnAlarmRings?.Invoke();
    }
}
```

5. ASP.NET Core API - CRUD

```
[HttpGet] public async Task<IEnumerable<Product>> Get() => await _context.Products.ToListAsync();
[HttpPost] public async Task<IActionResult> Post(Product product) { ... }
[HttpPut("{id}")] public async Task<IActionResult> Put(int id, Product product) { ... }
[HttpDelete("{id}")] public async Task<IActionResult> Delete(int id) { ... }
```

6. Model Binding - Nested Data

```
public class StudentViewModel {
    public string Name { get; set; }
    public AddressViewModel Address { get; set; }
}
```

7. Role-Based Authorization

```
[Authorize(Roles = "Admin")]
public class AdminController : Controller { ... }
```

8. Generic Repository + Unit of Work

```
public interface IGenericRepository<T> { ... }
public class UnitOfWork : IUnitOfWork { ... }
```

9. SQL - Duplicate Emails

```
SELECT Email, COUNT(*) FROM Users GROUP BY Email HAVING COUNT(*) > 1;
```

10. SQL - Second Highest Salary

```
SELECT MAX(Salary) FROM Employees WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

11. Stored Procedure with EF Core

```
await _context.Database.ExecuteSqlRawAsync("EXEC sp_UpdateProductStock @p0, @p1", productId, newStock);
```

12. Singleton Pattern

```
public sealed class Singleton {
    private static readonly Lazy<Singleton> _instance = new(() => new Singleton());
    public static Singleton Instance => _instance.Value;
    private Singleton() { }
}
```

13. Factory Pattern

```
public static class NotificationFactory {
    public static INotification Create(string type) { ... }
}
```

14. Dependency Injection

```
public class Application {
    private readonly ILoggerService _logger;
    public Application(ILoggerService logger) => _logger = logger;
}
```

15. System Design: Leave Management System

Modules: Employee, LeaveType, LeaveApplication, LeaveApproval.
ASP.NET Roles: Employee, Manager, HR

16. System Design: Cart Service (eCommerce)

CartItems Table: UserId, ProductId, Quantity.
POST /api/cart/add, GET /api/cart, POST /api/checkout