Exercise 1: The Basic Method

Goal: Convert a simple calculation method into a one-liner.

Traditional Code:

C#

```
public double CalculateCircleArea(double radius)
{
    return Math.PI * Math.Pow(radius, 2);
}
```

Your Task: Refactor this into an expression-bodied method.

Exercise 2: The Read-Only Property

Goal: Simplify a property that calculates a value based on other properties.

Traditional Code:

C#

```
public class Product
{
    public decimal Price { get; set; }
    public decimal Discount { get; set; }

    public decimal FinalPrice
    {
        get
        {
            return Price - Discount;
        }
    }
}
```

Your Task: Convert FinalPrice into an expression-bodied property.

Exercise 3: Constructor and Local Function

Goal: Use expression bodies for both a constructor and a local helper function.

Traditional Code:

C#

```
public class User
{
    private string _username;

    public User(string name)
    {
        _username = name.ToLower();
    }

    public void PrintGreeting()
    {
        string GetMessage()
        {
            return $"Hello, {_username}";
        }
        Console.WriteLine(GetMessage());
    }
}
```

Your Task: Refactor the **constructor** and the **local function** GetMessage() to use expression bodies.

Exercise 4: The Property "Set" Accessor

Goal: Apply the syntax to a property that has logic in the setter.

Traditional Code:

C#

```
private string _email;
public string Email
{
    get => _email;
    set
    {
        _email = value.Trim();
    }
}
```

Your Task: Refactor the set accessor to be expression-bodied.

Exercise 5: Overriding Methods

Goal: Simplify standard overrides like ToString().

Traditional Code:

C#

```
public class Car
{
    public string Make { get; set; }
    public string Model { get; set; }

    public override string ToString()
    {
        return $"{Make} {Model}";
    }
}
```

Your Task: Refactor the ToString() override to use the => operator.