CommandApp Console Tool

A simple C# console application that accepts a number as input and allows the user to apply various operations in an infinite loop.
Supports commands like `increment`, `decrement`, `double`, `randadd`, and `undo`.

Technologies Used

- Language: C#
- Framework: .NET 10.0 (Preview)
- SDK Version: `10.0.100-preview.2.25164.34`
- IDE: Visual Studio 2022 / Visual Studio Code

Getting Started

Make sure the correct .NET SDK is installed on your machine.

```
dotnet --version
# Expected output: 10.0.100-preview.2.25164.34
```

🧠 Commands Supported

- `increment` → Adds 1 to the result
- `decrement` → Subtracts 1 from the result
- `double` → Multiplies the result by 2
- `randadd` → Adds a random number (e.g., from -10 to +10)
- `undo` → Reverts the last valid command (not undo itself)

🚀 How to Run

`dotnet run --project ./CommandApp -- 1`

🛠️ Example

- Current result: 1

> increment

- Current result: 2

> double

- Current result: 4

> undo

- Current result: 2

> Decrement

- Current result: 1

> RandAdd

- Current result: 1 + random number (6): 7

Design Patterns & Data Structures

This application uses the **Command Pattern** and a **Stack** to manage user commands and support the `undo` functionality.

Command Pattern

Each operation (increment, decrement, etc.) is implemented as its own class following the `ICommand` interface.

Benefits:

- Easy to add new commands
- Encapsulates execution and undo logic per command
- Clean structure following SOLID principles (especially Open/Closed)

Stack (LIFO)

We use a stack to track executed commands.
This is perfect for implementing the **undo** feature, as it allows us to **revert the last command** in O(1) time.

Why a stack?

- Fits the *Last-In-First-Out* logic of undo
- Simple, fast, and memory-efficient
- Well-suited for linear command history

Progress Status

- ☑ Initial project and solution structure created
- ☑ Projects added: `CommandApp` and `CommandApp.Tests`
- ☑ Command classes implemented: `IncrementCommand`, `DecrementCommand`, `DoubleCommand`, `RandAddCommand`
- ☑ `ICommand` interface added
- ☑ `CommandContext.cs` added to manage execution and command stack
- ☑ `Program.cs` updated to use CommandContext
- ☑ Uploaded to GitHub: AntoniousShehata/EKVIP_APP

☑ Unit Testing

Unit tests have been added using xUnit to ensure correct behavior of all commands.

To create the test project run this command

```
dotnet new xunit -n CommandApp.Tests
```

🧹 Covered Commands

- `IncrementCommand`
  - ✔ Execute: Increases the value by 1
  - 🔁 Undo: Decreases the value by 1
- `DecrementCommand`
  - ✔ Execute: Decreases the value by 1
  - 🔁 Undo: Increases the value by 1
- `DoubleCommand`
  - ✔ Execute: Multiplies the value by 2
  - 🔁 Undo: Divides the value by 2
- `RandAddCommand`
  - ✔ Execute: Adds a random number to the value
  - 🔁 Undo: Subtracts the same random number to restore the original value

▶ Run Tests

To run all unit tests, use the command:

```
cd CommandApp.Tests
dotnet test
```

☑ Sample Output

```
Test summary: total: 8, failed: 0, succeeded: 8, skipped: 0, duration: 3.8s
Build succeeded with 2 warning(s) in 10.3s
```

 Author

Antonious Shehata || April 2025