BOOSE Interpreter

This Project is a custom drawing language interpreter created as part of the Advanced Software Engineering module. It Demonstrates the use of Object-Oriented design principles and software architecture patterns like Factory and Command

Quick Start

- View API Documentation - Full reference for all classes and methods
- BOOSE Application - Main source code and components
- BOOSE Commands - Command Classes
- Unit Tests - Documentation for the Unit Tests

Key Features

- **Custom Drawing Language** - Interprets user-written programs to draw shapes
- **Command Pattern** – Each drawing instruction is implemented as a command
- **Error Handling** – Displays meaningful error messages with line numbers
- **Unit Testing** – Over 50 unit tests to verify functionality
- **Graphical Output** – Draws shapes such as circles, rectangles, and lines on a canvas

Core Components

- MyParser – Reads and validates user programs (supports multi-line comments)
- MyCommandFactory – Responsible for creating command objects dynamically
- ICommandHandler – Interface implemented by all command classes
- AppCanvas – Custom canvas used for drawing shapes
- Command Classes – Includes MoveTo, DrawTo, Circle, Rectangle, and PenColour

Example BOOSE Program

```
// Draw a red Rectangle
pen 255,0,0
moveto 50,50
rect 200,100

// Draw a blue circle
pen 0,0,255
circle 40
```

About

This project was developed by **Abdullah Alromaihi** as part of the **Advanced Software Engineering** module at **Leeds Beckett University**.
It was built using **C# (.NET)** and demonstrates the use of key **object-oriented design patterns**, including the **Factory** and **Command** patterns.

The project focuses on building a simple yet professional drawing interpreter that processes custom commands to create shapes, with an emphasis on **clean architecture**, **testing**, and **error handling**.

API Documentation

This section contains the generated API reference for the BOOSEapp project.

- BOOSE App
- BOOSE Commands
- BOOSE Unit Tests

Namespace BOOSEapp

Classes

AppCanvas

Implements the BOOSE ICanvas interface for drawing operations Handles all graphic operations like drawing shapes, lines, and managing colours This is the actual canvas that the BOOSE system uses for rendering

CommandException

Custom exceprion for command related errors in the custom BOOSE interpreter Used when commands have problems with parameters or execution

Form1

Main application window for the BOOSE drawing interpreter Handles UI, user input, and coordinates between parser and canvas

MoveToCommand

Handles the moveto command, it moves the cursor wuthout drawing anything Usage: moveto 200,100

MyCommandFactory

Creates command objects based on command names This is the "command creator" is knows how to make each type of command

MyParser

MyStoredProgram

Custom program storage that holds and executes drawing commands This is where all the commands live before they get executed

Interfaces

ICommandHandler

Interface all command classes must implement Defines the basic structre for every drawing command

Namespace BOOSEapp.commands

Classes

CircleCommand

Handles the circle command - It draws a circle at current position Usage: circle 50

DrawToCommand

Handles the drawto command - draws a line from current position to a new position Usage: drawto 110,150

PenColourCommand

Handles the pen command - changes the pen colour Usage: pen 255,0,0 (for Red colour)

RectCommand

Handles the rect command - draws a rectangle at current position Usage: rect 50,100

Namespace BOOSEUnitTests

Classes

CommandFactoryTests

Unit test for MyCommandFactory to verify it creates command objects based on command types correctly

CommandsTests

Unit tests for individual command classes to verify successful execution and parameters are handled properly

IntegrationTests

Integration tests to verify multiple components work together correctly in complete system workflows

ParserTests

Unit tests for MyParser to verify it correctly pares the program text into executable commands