Introduction

Overview

This project implements **BOOSE (Basic Object-Oriented Software Engineering) Interpreter**, a comprehensive medium-scale software engineering solution built with .NET 8. The interpreter provides a complete implementation of the BOOSE language runtime, enabling execution of custom domain-specific programs with support for variables, methods, control flow, and graphics rendering.

Project Purpose

The BOOSE Interpreter serves as an educational and functional implementation of a domain-specific language (DSL) designed for procedural programming with integrated graphics capabilities. It demonstrates core software engineering principles including architecture design, code organization, and systematic testing practices suitable for medium-scale applications.

Key Features

- **BOOSE Language Execution**: Full-featured interpreter that parses and executes BOOSE programs with support for:
    - Variable management (integer, real, and boolean types)
    - Method definitions with parameters and return values
    - Control flow structures (if/else, for loops, while loops)
    - Array operations (peek/poke functionality)
    - Boolean expressions with logical operators
- **Graphics Rendering**: Integrated canvas system enabling:
    - Shape drawing (circles, rectangles, triangles)
    - Color management via RGB values
    - Cursor positioning and line drawing
    - Text rendering capabilities
- **Robust Architecture**:
    - Interface-driven design with `IBooseRunner` and `IVariableManager`
    - Separation of concerns through `AppCanvas` and `VariableManager`
    - Comprehensive error handling and validation
    - Structured method definition storage

Technology Stack

- **Framework**: .NET 8
- **Language**: C#

- **UI Framework**: Windows Forms
- **Testing**: Microsoft Testing Framework (MSTest)

Project Structure

The solution is organized into several key components:

- **UnrestrictedBooseRunner**: Core interpreter engine responsible for parsing and executing BOOSE code
- **VariableManager**: Manages variable storage, scoping, and lifecycle
- **AppCanvas**: Graphics abstraction layer providing drawing primitives
- **Form1**: Windows Forms user interface for interacting with the interpreter
- **CommandFactory**: Factory pattern implementation for command handling
- **Test Suite**: Comprehensive unit tests validating interpreter functionality

Getting Started

To begin using the BOOSE Interpreter, refer to the Getting Started guide which covers installation, configuration, and basic usage examples.

Documentation

This documentation provides detailed information about:

- Setting up and configuring the interpreter
- Writing and executing BOOSE programs
- Architecture and design patterns
- API reference and method documentation
- Testing and quality assurance practices

Namespace Medium_Scale_Software_Engineering_Project

Classes

AppCanvas

Canvas for drawing shapes and lines. Implements the ICanvas interface for BOOSE.

Form1

Main form for the BOOSE interpreter app.

MyCommandFactory

Factory for creating command objects for BOOSE. Handles error reporting for command creation.

UnrestrictedBooseRunner

VariableManager

Manages variables, arrays, and their types for BOOSE programs. Handles declaration, assignment, and value retrieval.

Interfaces

IBooseRunner

Interface for BOOSE program execution. Defines the contract for running BOOSE code with a canvas.

IVariableManager

Interface for managing variables in BOOSE programs. Defines contracts for variable declaration, assignment, and retrieval.