ASSIGNMENT # 03

COMSATS UNIVERSITY
ATTOCK CAMPUS

NAME: SARA BIBI

REG# (SP21-BCS-033)

SUBMITTED TO:

SIR BILAL HAIDER

DATE: 28,NOV , 2024

DOMAIN SPECIFIC LANGUAGE (DSL):

A **Domain-Specific Language (DSL)** is a programming language or syntax created specifically for a particular application domain. Unlike general-purpose languages (like C#, Python, or Java), a DSL is tailored to address specific tasks or problems within a niche area, making it more expressive and easier to use for that purpose.

What Does "Designing a DSL in C#" Mean?

Designing a DSL in C# involves creating a specialized mini-language or syntax, embedded within or external to the C# code that allows users to define or manipulate elements related to a specific problem domain.

Why Use a DSL?

1. **Improves Productivity:** Allows users to describe complex tasks in a simplified manner.
2. **Increases Readability:** Makes the code or configuration easier to understand for domain experts.
3. **Focus on the Domain:** Reduces the need to deal with general programming constructs when working within a specific domain.

Types of DSLs

1. **Internal DSL:** Embedded within an existing programming language, often leveraging its syntax and libraries.
   Example: LINQ in C#.
2. **External DSL:** Defined independently with its custom syntax and requires a parser to interpret.
   Example: SQL or a configuration file format like JSON/YAML.

Steps to Design a DSL in C#

1. **Define the Domain**: Decide the specific domain and tasks your DSL will address.
   Example: Configuring gameplay elements for a space shooter game.
2. **Design the Syntax**: Define the structure and format of your DSL.
   Example: Use a JSON-like or custom block-style syntax.
3. **Implement Parsing**: Write a parser to interpret the DSL and convert it into objects or commands.
4. **Build an Interpreter/Compiler**: Implement logic to execute or transform the parsed DSL into actions.
5. **Integrate with Applications**: Use the DSL to define configurations or behaviors within your C# application.

CODE.

```
Using System;
using System.Collections.Generic;

namespace GalaxyShooterDSL
{
    // DSL Parser
    public class DSLParser
    {
        public Dictionary<string, Dictionary<string, object>> Parse(string script)
        {
            var objects = new Dictionary<string, Dictionary<string, object>>();
            var lines = script.Split('\n');
            string currentType = null;
            Dictionary<string, object> currentObject = null;

            foreach (var line in lines)
            {
                var trimmed = line.Trim();

                if (trimmed.StartsWith("#") || string.IsNullOrWhiteSpace(trimmed))
                    continue; // Skip comments and empty lines

                if (trimmed.EndsWith("{"))
                {
                    currentType = trimmed.Split(' ')[0];
                    currentObject = new Dictionary<string, object>();
```

```
            objects[currentType] = currentObject;
        }
        else if (trimmed.EndsWith("}"))
        {
            currentType = null;
            currentObject = null;
        }
        else
        {
            var parts = trimmed.Split(':');
            var key = parts[0].Trim();
            var value = parts[1].Trim().TrimEnd(';');

            if (currentObject != null)
            {
                if (value.StartsWith("[") && value.EndsWith("]")) // Array
                {
                    value = value.Trim('[', ']');
                    currentObject[key] = value.Split(',');
                }
                else
                {
                    currentObject[key] = value;
                }
            }
        }
    }
```

```
        return objects;
    }
}

// Game Interpreter
public class GameInterpreter
{
    public void GenerateGameElements(Dictionary<string, Dictionary<string, object>>
parsedObjects)
    {
        foreach (var obj in parsedObjects)
        {
            if (obj.Key == "PlayerShip")
            {
                CreatePlayerShip(obj.Value);
            }
            else if (obj.Key == "EnemyWave")
            {
                CreateEnemyWave(obj.Value);
            }
            else if (obj.Key == "PowerUp")
            {
                CreatePowerUp(obj.Value);
            }
            else if (obj.Key == "Level")
            {
                CreateLevel(obj.Value);
            }
        }
```

```
    }

    private void CreatePlayerShip(Dictionary<string, object> shipData)
    {
        Console.WriteLine($"Player Ship: {shipData["Name"]}, Speed: {shipData["Speed"]}, Health: {shipData["Health"]}");
        Console.WriteLine($"Weapons: {string.Join(", ", (string[])shipData["Weapons"])}");
    }

    private void CreateEnemyWave(Dictionary<string, object> waveData)
    {
        Console.WriteLine($"Enemy Wave: {waveData["Name"]}, Type: {waveData["EnemyType"]}, Count: {waveData["Count"]}, Interval: {waveData["SpawnInterval"]}s");
    }

    private void CreatePowerUp(Dictionary<string, object> powerUpData)
    {
        Console.WriteLine($"Power-Up: {powerUpData["Type"]}, Duration: {powerUpData["Duration"]}s, Location: {string.Join(", ", (string[])powerUpData["SpawnLocation"])}");
    }

    private void CreateLevel(Dictionary<string, object> levelData)
    {
        Console.WriteLine($"Level: {levelData["Name"]}, Background: {levelData["Background"]}");
        Console.WriteLine($"Waves: {string.Join(", ", (string[])levelData["Waves"])}");
        Console.WriteLine($"Power-Ups: {string.Join(", ", (string[])levelData["PowerUps"])}");
```

```
    }
}

// Main Program
public class Program
{
    public static void Main()
    {
        string script = @"
        # Define a player ship
        PlayerShip {
            Name: ""Falcon"";
            Speed: 500;
            Health: 100;
            Weapons: [""Laser"", ""Missile""];
        }

        # Define an enemy wave
        EnemyWave {
            Name: ""Wave1"";
            EnemyType: ""Fighter"";
            Count: 5;
            SpawnInterval: 2;
        }

        # Define a power-up
        PowerUp {
            Type: ""Shield"";
```

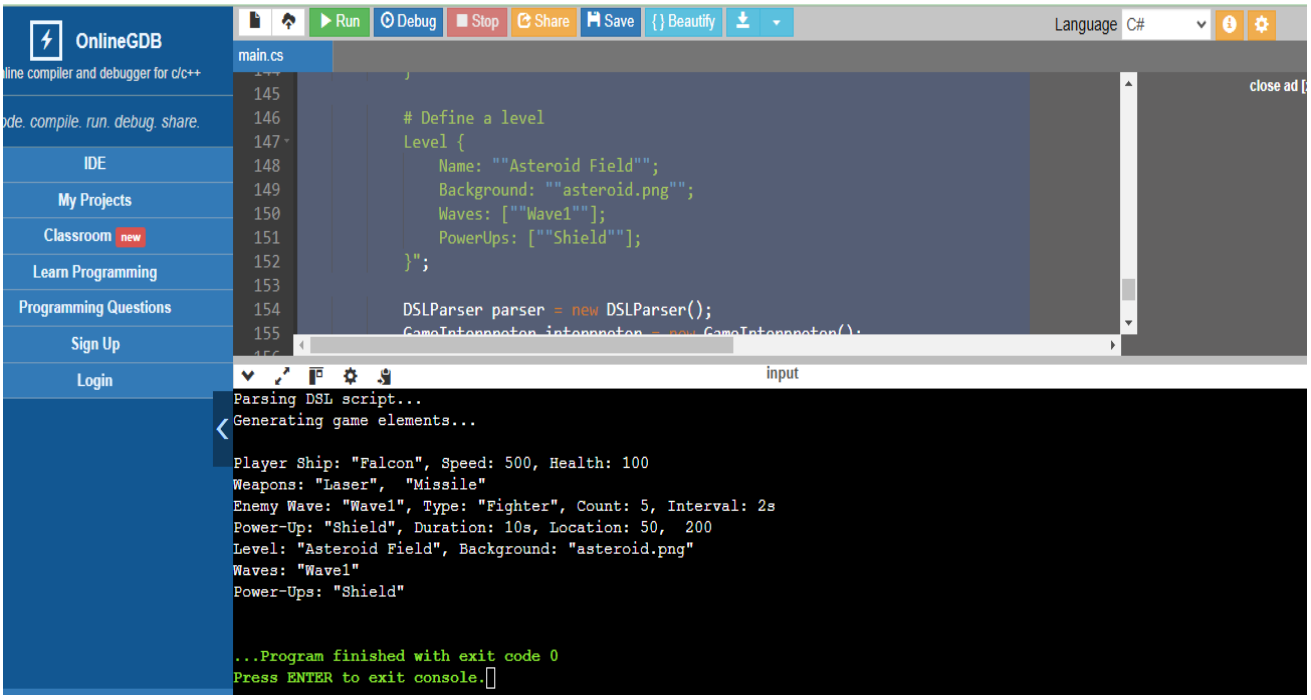
```
            Duration: 10;
            SpawnLocation: [50, 200];
        }

        # Define a level
        Level {
            Name: ""Asteroid Field"";
            Background: ""asteroid.png"";
            Waves: [""Wave1""];
            PowerUps: [""Shield""];
        }";

        DSLParser parser = new DSLParser();
        GameInterpreter interpreter = new GameInterpreter();

        Console.WriteLine("Parsing DSL script...");
        var parsedObjects = parser.Parse(script);
        Console.WriteLine("Generating game elements...\n");
        interpreter.GenerateGameElements(parsedObjects);
    }
  }
}
```

OUTPUT:


OnlineGDB
line compiler and debugger for c/c++
ode. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
Run
Debug
Stop
Share
Save
{} Beautify
Language C#
main.cs
close ad
# Define a level
Level {
Name: ""Asteroid Field"";
Background: ""asteroid.png"";
Waves: [""Wave1""];
PowerUps: [""Shield""];
}";
DSLParser parser = new DSLParser();
input
Parsing DSL script...
Generating game elements...
Player Ship: "Falcon", Speed: 500, Health: 100
Weapons: "Laser", "Missile"
Enemy Wave: "Wave1", Type: "Fighter", Count: 5, Interval: 2s
Power-Up: "Shield", Duration: 10s, Location: 50, 200
Level: "Asteroid Field", Background: "asteroid.png"
Waves: "Wave1"
Power-Ups: "Shield"
...Program finished with exit code 0
Press ENTER to exit console.