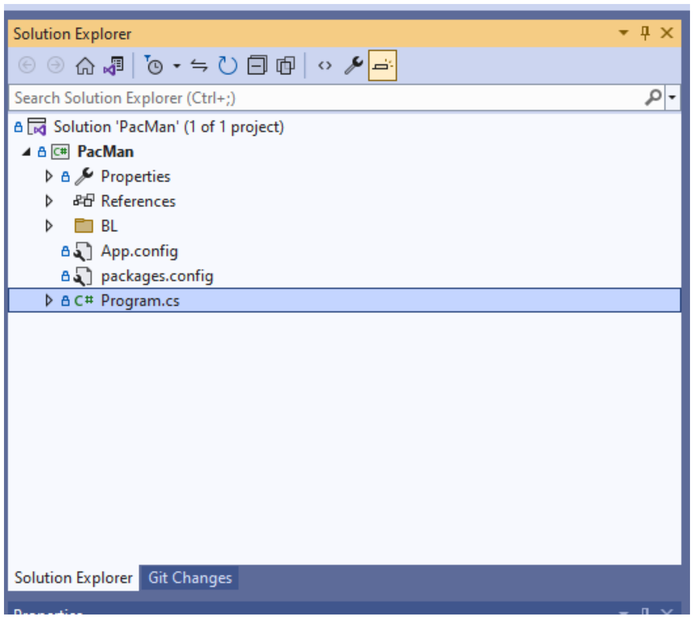
Introduction

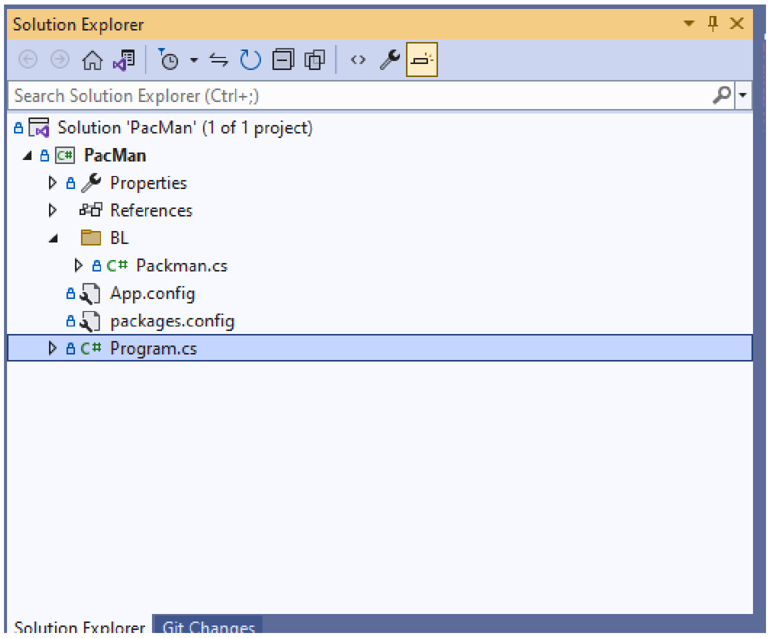
In this guide, we will convert our previous game into an object-oriented program. However, we will only introduce the concept of a class in the game and make all the necessary changes in the code

Here we have some steps to perform

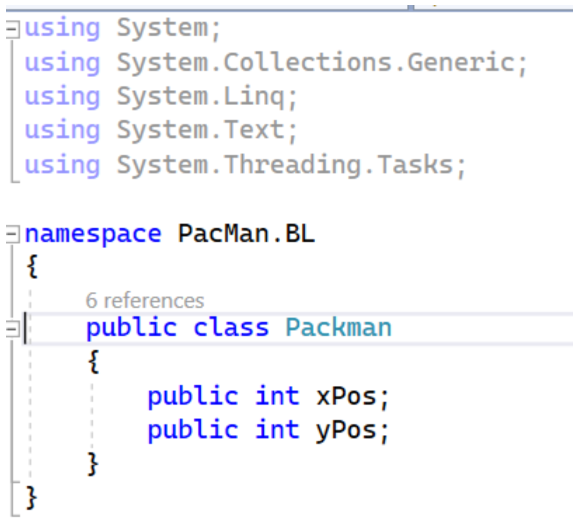
Step1:
Create a Folder in the Solution explore with name BL (Business Layer)



Step2 :
Create a class with Pacman in the BL folder

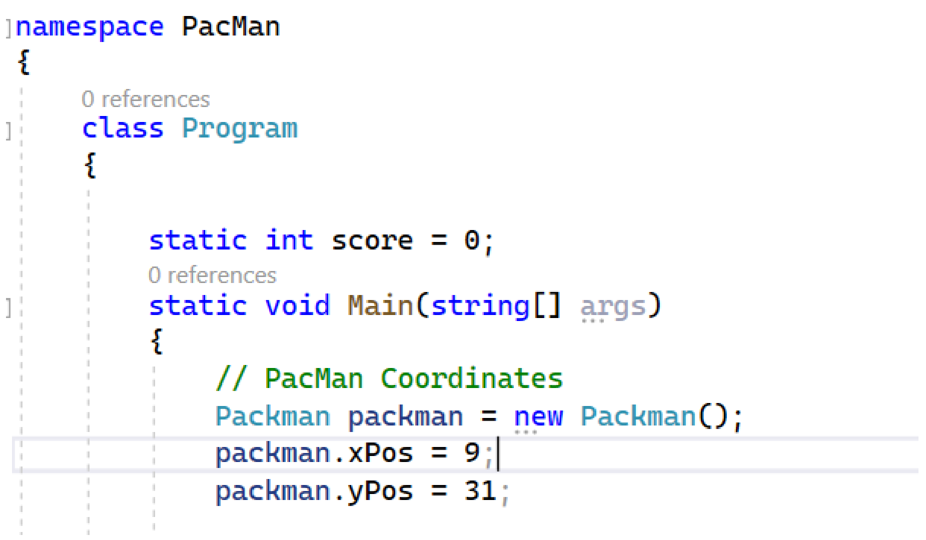


Step 3:
Create the required parameters in the class. Since we only need the x and y position of the Pacman, we will create two variables for this purpose.



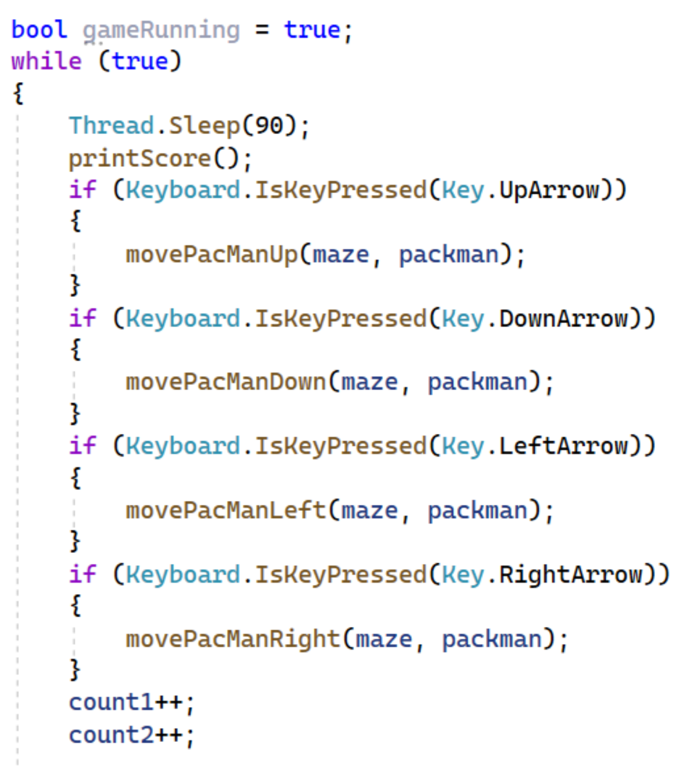
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PacMan.BL
{
    6 references
    public class Packman
    {
        public int xPos;
        public int yPos;
    }
}
```

Step 4:
Our main function is in the Program.cs file, which contains the driver code to manage the flow of the entire game. Once we create an object, we need to initialize its variables



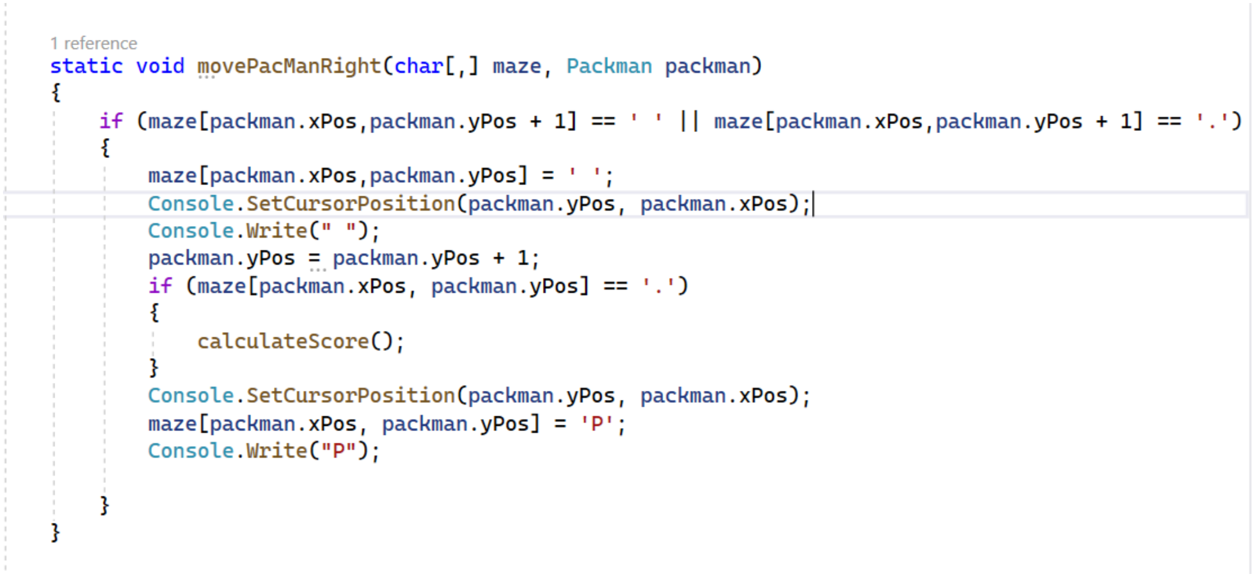
```
namespace PacMan
{
        0 references
        class Program
        {

            static int score = 0;
            0 references
            static void Main(string[] args)
            {
                // PacMan Coordinates
                Packman packman = new Packman();
                packman.xPos = 9;
                packman.yPos = 31;
```

Make all the necessary changes in the code, such as modifying functions to accept the Pacman object instead of passing the x and y positions separately. In the function, we will use the parameters of the Pacman object to perform the required actions.



```
bool gameRunning = true;
while (true)
{
    Thread.Sleep(90);
    printScore();
    if (Keyboard.IsKeyPressed(Key.UpArrow))
    {
        movePacManUp(maze, packman);
    }
    if (Keyboard.IsKeyPressed(Key.DownArrow))
    {
        movePacManDown(maze, packman);
    }
    if (Keyboard.IsKeyPressed(Key.LeftArrow))
    {
        movePacManLeft(maze, packman);
    }
    if (Keyboard.IsKeyPressed(Key.RightArrow))
    {
        movePacManRight(maze, packman);
    }
    count1++;
    count2++;
```

In the function implementation, we are using the data members of the Pacman object instead of passing the parameters separately to the functions.


```
1 reference
static void movePacManRight(char[,] maze, Packman packman)
{
    if (maze[packman.xPos,packman.yPos + 1] == ' ' || maze[packman.xPos,packman.yPos + 1] == '.')
    {
        maze[packman.xPos,packman.yPos] = ' ';
        Console.SetCursorPosition(packman.yPos, packman.xPos);
        Console.Write(" ");
        packman.yPos = packman.yPos + 1;
        if (maze[packman.xPos, packman.yPos] == '.')
        {
            calculateScore();
        }
        Console.SetCursorPosition(packman.yPos, packman.xPos);
        maze[packman.xPos, packman.yPos] = 'P';
        Console.Write("P");

    }
}
```

Task 1:
Implement only one User of your business application, which you developed in last Programming Day.
Make a Class for your User. Define its Data Members.
Then implement the Create, Read, Update, and Delete features for that user using the objects of the Class. Use List instead of Arrays to store and process the data in the program. Use file handling to manage the records.

Takeaway Task

Implement only following feature from your previous game, such as one player, one enemy, and all collision mechanics (e.g., enemy fire, player fire), score management, and health management which you developed in last Programming Day.
Make a relevant Class for your Player and define its data members.
Hint: Use 2D array to store the map of the game and manage everything. Without 2D arrays you cannot do that.