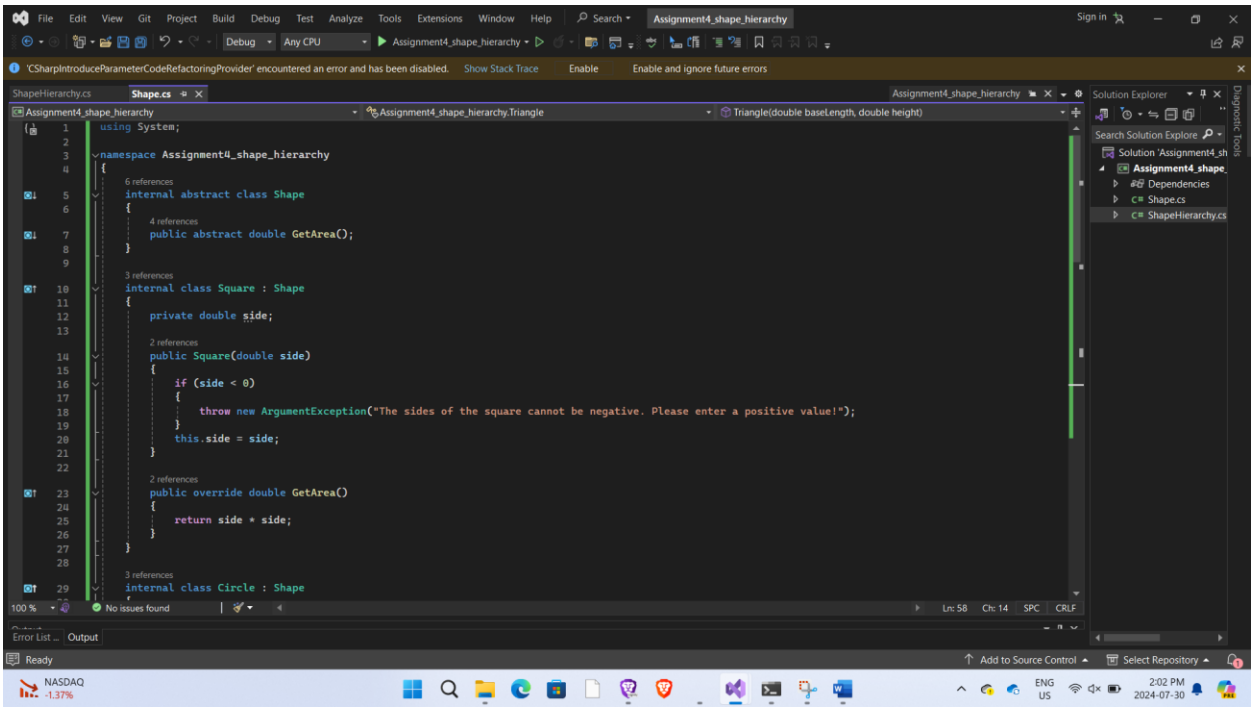
Assignment #4

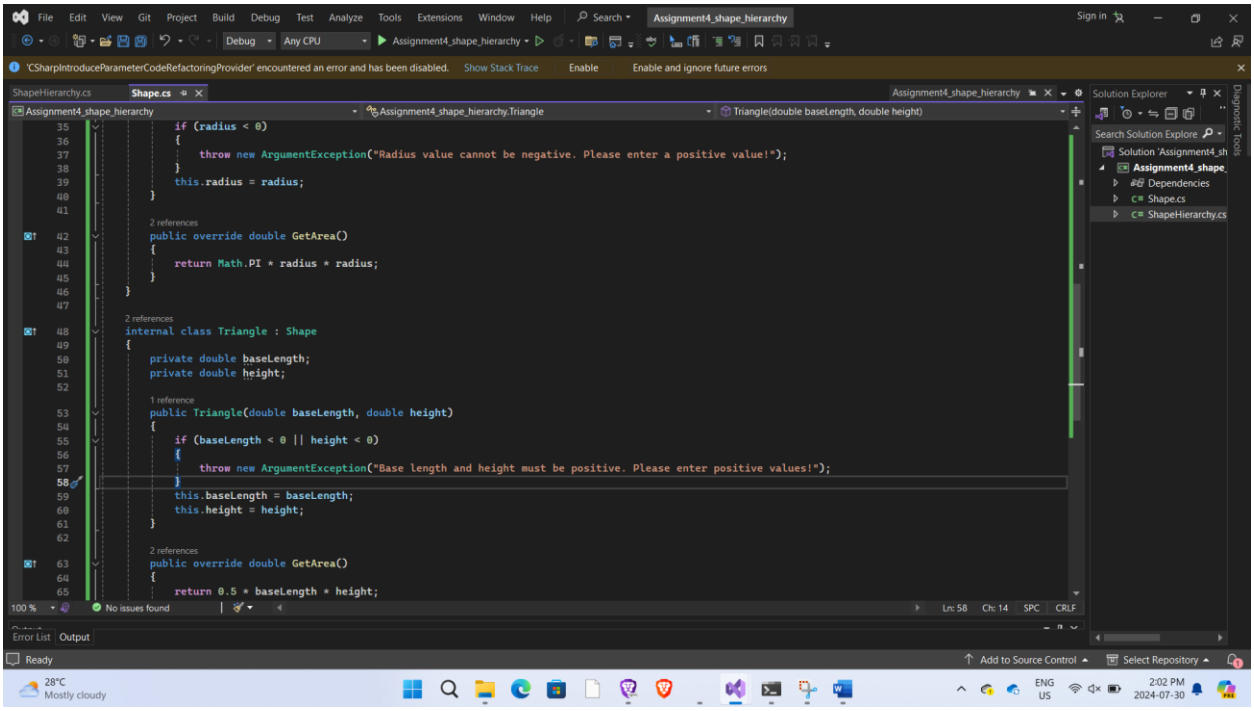
Student Name : Shreejana Shrestha

Student Id : C0930321

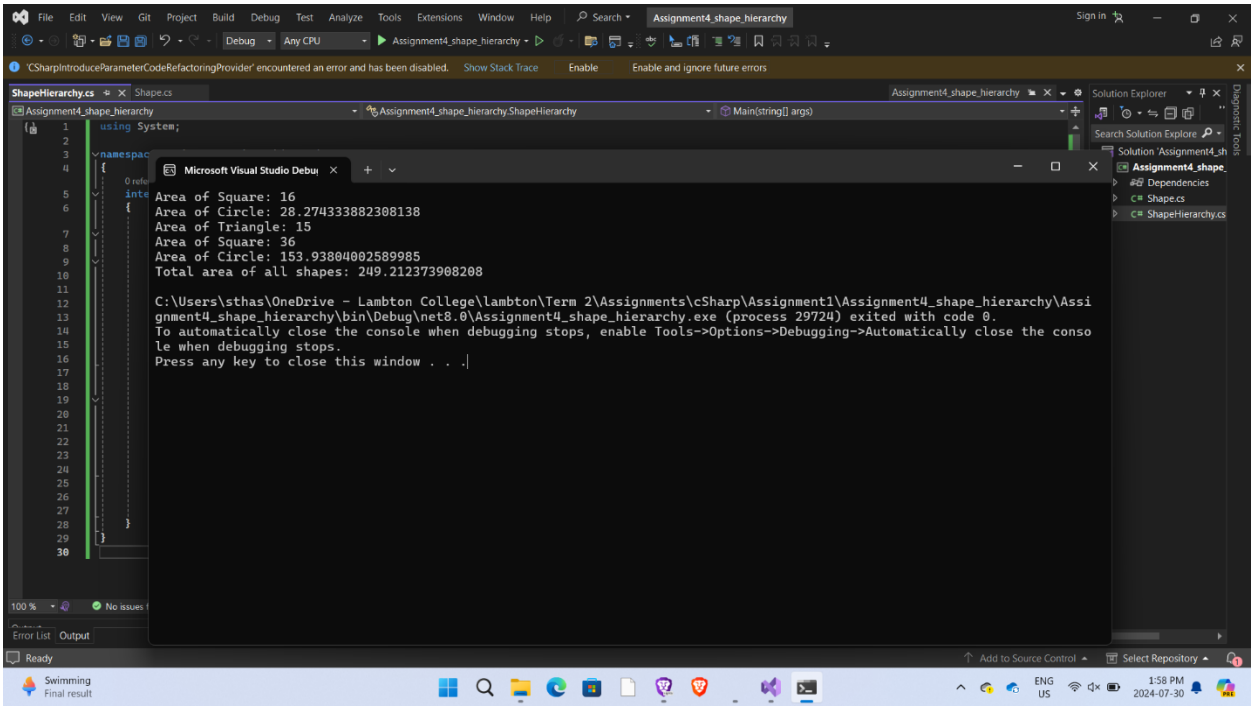
Lab Question: Shape Hierarchy

Code Screenshots



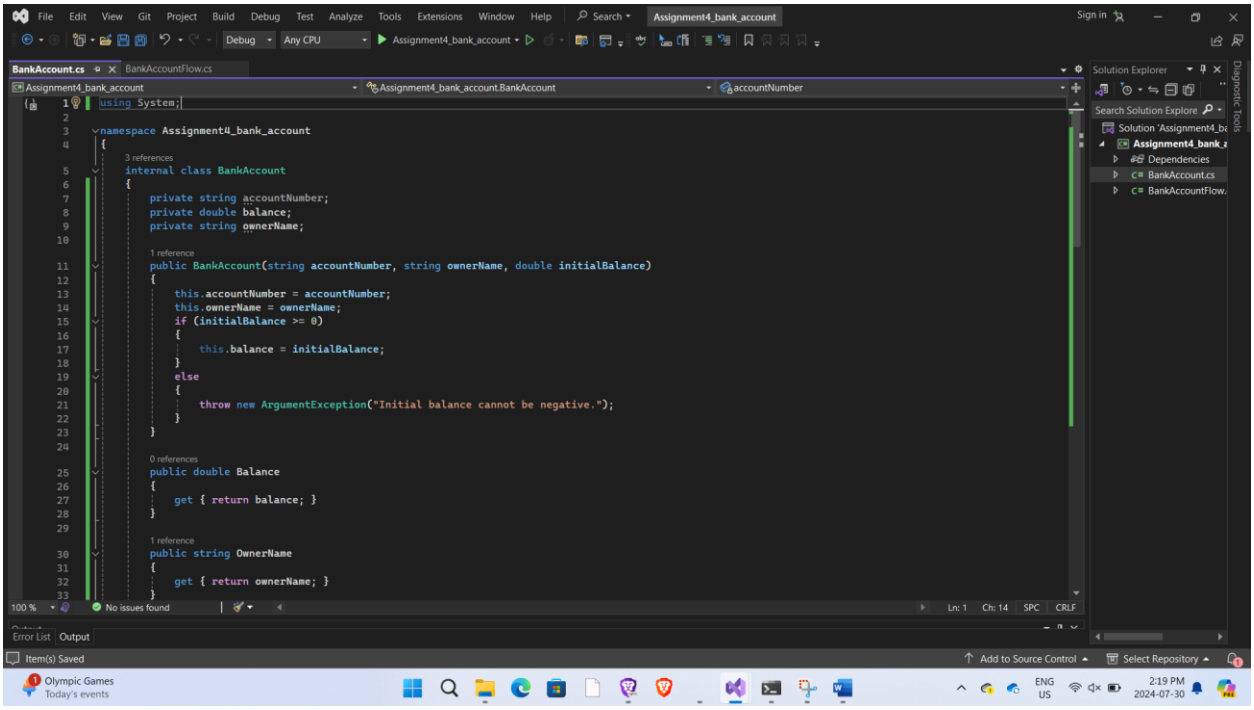

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
Assignment4_shape_hierarchy
Sign in
Debug
Any CPU
Assignment4_shape_hierarchy
'CSharpIntroduceParameterCodeRefactoringProvider' encountered an error and has been disabled.
Show Stack Trace
Enable
Enable and ignore future errors
ShapeHierarchy.cs
Shape.cs
Assignment4_shape_hierarchy
Assignment4_shape_hierarchy.Triangle
Triangle(double baseLength, double height)
if (radius < 0)
{
throw new ArgumentException("Radius value cannot be negative. Please enter a positive value!");
}
this.radius = radius;
}
2 references
public override double GetArea()
{
return Math.PI * radius * radius;
}
}
2 references
internal class Triangle : Shape
{
private double baseLength;
private double height;
1 reference
public Triangle(double baseLength, double height)
{
if (baseLength < 0 || height < 0)
{
throw new ArgumentException("Base length and height must be positive. Please enter positive values!");
}
this.baseLength = baseLength;
this.height = height;
}
2 references
public override double GetArea()
{
return 0.5 * baseLength * height;
100 %
No issues found
Ln: 58
Ch: 14
SPC
CRLF
Error List
Output
Solution Explorer
Search Solution Explore
Solution 'Assignment4_sh
Assignment4_shape_
Dependencies
Shape.cs
ShapeHierarchy.cs
Diagnostic Tools
Ready
Add to Source Control
Select Repository
28°C
Mostly cloudy
ENG
US
2:02 PM
2024-07-30

Output Screenshots

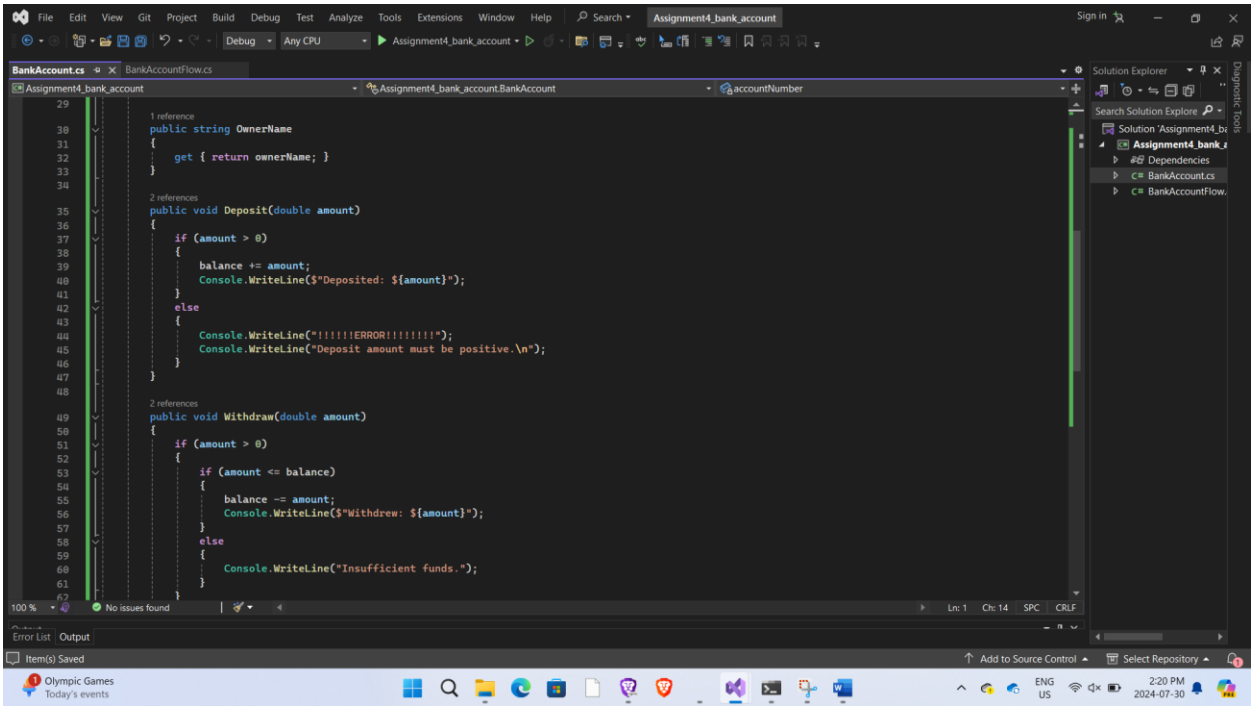


Lab Question: Bank Account

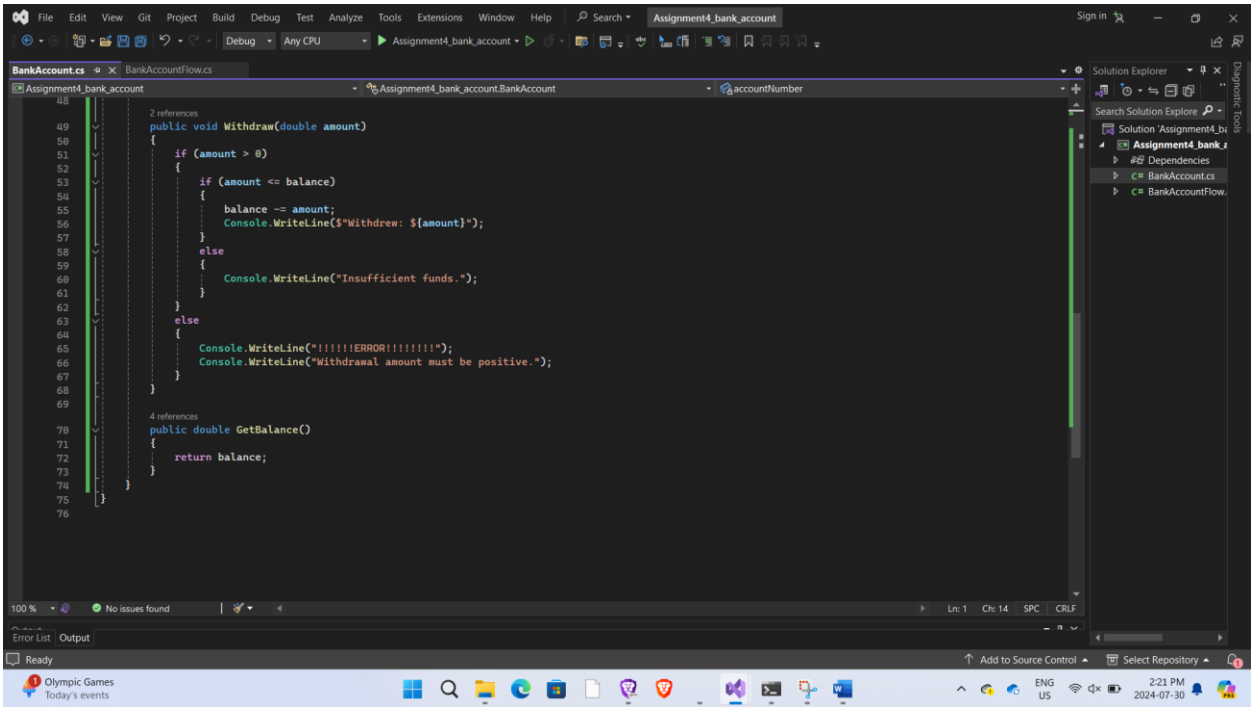
Code screenshots



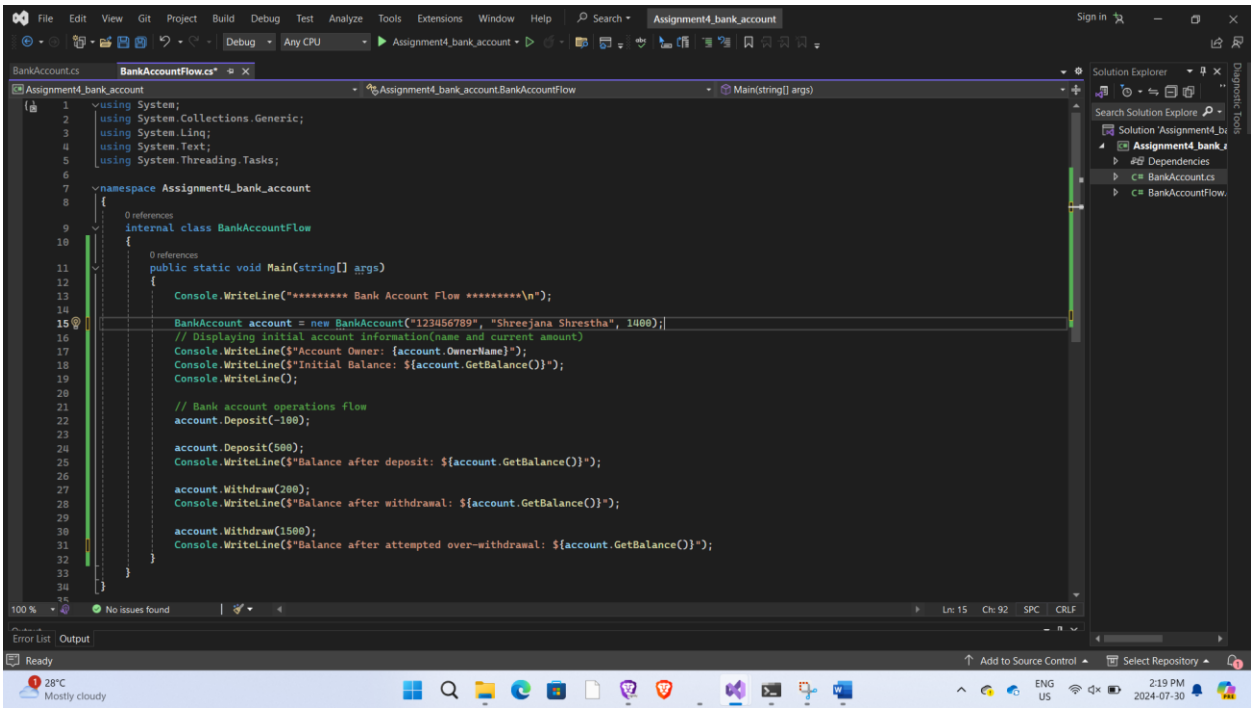
```
using System;

namespace Assignment4_bank_account
{
    internal class BankAccount
    {
        private string accountNumber;
        private double balance;
        private string ownerName;

        public BankAccount(string accountNumber, string ownerName, double initialBalance)
        {
            this.accountNumber = accountNumber;
            this.ownerName = ownerName;
            if (initialBalance >= 0)
            {
                this.balance = initialBalance;
            }
            else
            {
                throw new ArgumentException("Initial balance cannot be negative.");
            }
        }

        public double Balance
        {
            get { return balance; }
        }

        public string OwnerName
        {
            get { return ownerName; }
        }
```



```
        public string OwnerName
        {
            get { return ownerName; }
        }

        public void Deposit(double amount)
        {
            if (amount > 0)
            {
                balance += amount;
                Console.WriteLine($"Deposited: ${amount}");
            }
            else
            {
                Console.WriteLine("!!!!!!ERROR!!!!!!!");
                Console.WriteLine("Deposit amount must be positive.\n");
            }
        }

        public void Withdraw(double amount)
        {
            if (amount > 0)
            {
                if (amount <= balance)
                {
                    balance -= amount;
                    Console.WriteLine($"Withdrew: ${amount}");
                }
                else
                {
                    Console.WriteLine("Insufficient funds.");
                }
```



```
2 references
public void Withdraw(double amount)
{
    if (amount > 0)
    {
        if (amount <= balance)
        {
            balance -= amount;
            Console.WriteLine($"Withdrew: ${amount}");
        }
        else
        {
            Console.WriteLine("Insufficient funds.");
        }
    }
    else
    {
        Console.WriteLine("!!!!!!ERROR!!!!!!!");
        Console.WriteLine("Withdrawal amount must be positive.");
    }
}

4 references
public double GetBalance()
{
    return balance;
}
}
}
```



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

namespace Assignment4_bank_account
{
    0 references
    internal class BankAccountFlow
    {
        0 references
        public static void Main(string[] args)
        {
            Console.WriteLine("********* Bank Account Flow *********\n");

            BankAccount account = new BankAccount("123456789", "Shreejana Shrestha", 1400);
            // Displaying initial account information(name and current amount)
            Console.WriteLine($"Account Owner: {account.OwnerName}");
            Console.WriteLine($"Initial Balance: ${account.GetBalance()}");
            Console.WriteLine();

            // Bank account operations flow
            account.Deposit(-100);

            account.Deposit(500);
            Console.WriteLine($"Balance after deposit: ${account.GetBalance()}");

            account.Withdraw(200);
            Console.WriteLine($"Balance after withdrawal: ${account.GetBalance()}");

            account.Withdraw(1500);
            Console.WriteLine($"Balance after attempted over-withdrawal: ${account.GetBalance()}");
        }
    }
}
```

Output Screenshots