**HOMEWORK: 12/01/2022**

Table of Contents

[**Exercise 1: Rhombus of Stars** 2](#_Toc124463969)

[**1.** **Source code:** 2](#_Toc124463970)

[**2.** **Explain:** 2](#_Toc124463971)

[**3.** **Result:** 2](#_Toc124463972)

[**Exercise 2: Point in Rectangle** 2](#_Toc124463973)

[**1.** **Source code:** 2](#_Toc124463974)

[**2.** **Explain:** 2](#_Toc124463975)

[**3.** **Result:** 4](#_Toc124463976)

[**Exercise 3: Student System** 4](#_Toc124463977)

[**1.** **Source code:** 4](#_Toc124463978)

[**2.** **Explain:** 4](#_Toc124463979)

[**3.** **Result:** 5](#_Toc124463980)

[**Exercise 4: Hotel Reservation** 6](#_Toc124463981)

[**1.** **Source code:** 6](#_Toc124463982)

[**2.** **Explain:** 6](#_Toc124463983)

[**3.** **Result:** 7](#_Toc124463984)

# **Exercise 1: Rhombus of Stars**

## **Source code:**

[1651---Advanced-Programming/Classwork1 at main · KieTuaNguyen/1651---Advanced-Programming (github.com)](https://github.com/KieTuaNguyen/1651---Advanced-Programming/tree/main/Classwork1)

## **Explain:**

This code defines a C# class called "StarRhombus" which has a single property "Size" and a method "PrintRow" which takes two integer parameters, "size" and "starCount". The main program creates an instance of the StarRhombus class, prompts the user to enter the size of the rhombus, and assigns the value to the Size property. Then, it uses two for loops to call the PrintRow method twice, first with increasing values of "i" and then with decreasing values of "i". The PrintRow method uses two nested for loops to print a row of stars in the shape of a rhombus. The first for loop prints a number of spaces equal to the size of the rhombus minus the number of stars. The second for loop prints the number of stars passed to the method. Each row is printed on a new line.

## **Result:**

Text

Description automatically generated

# **Exercise 2: Point in Rectangle**

## **Source code:**

[1651---Advanced-Programming/Classwork2 at main · KieTuaNguyen/1651---Advanced-Programming (github.com)](https://github.com/KieTuaNguyen/1651---Advanced-Programming/tree/main/Classwork2)

## **Explain:**

**Class Point:**

This code defines a C# class called "Point" which inherits from the Object class. The class has two private fields "x" and "y", and two public properties "X" and "Y" with getters and setters respectively. The class also has two constructors. The first constructor is the default constructor which does not take any parameters. The second constructor takes two integer parameters "x" and "y" and assigns them to the class fields. The class also has a method "ToString" which override the ToString method of the object class and returns a string in the format "Point (x,y)" when the object is printed.

**Class Rectangle:**

This code defines a C# class called "Rectangle" which inherits from the Object class. The class has two public properties "TopLeft" and "BottomRight" of type Point which are set by the user. The class also has two constructors. The first constructor is the default constructor which does not take any parameters. The second constructor takes two Point parameters "topLeft" and "bottomRight" and assigns them to the class properties. The class also has a method "ToString" which override the ToString method of the object class and returns a string in the format "(TopLeft,BottomRight)" when the object is printed.

It also has a method "IsContainsPoint" which takes a point object as a parameter, and check if the point is inside the rectangle or not. It checks the x and y coordinates of the point, if x is between the x coordinates of the top left and bottom right point of the rectangle and y is between the y coordinates of the top left and bottom right point of the rectangle then it returns true otherwise false.

**Class Program:**

This code defines a C# class called "Program" which contains the Main method. The Main method creates an instance of the "Point" class, prompts the user to enter the x and y coordinates of the point, and assigns the values to the x and y properties of the point object. Then it prints the point coordinates.

It also creates an instance of the "Rectangle" class and assigns the top-left and bottom-right point of the rectangle. Then it prints the coordinates of the rectangle.

It also calls the IsContainsPoint method of the rectangle class, passing the point object as a parameter, and prints whether or not the point is inside the rectangle. The IsContainsPoint method is not defined in this code snippet, it is supposed that this method is defined in the rectangle class and it checks whether the point is inside the rectangle or not.

## **Result:**

Text

Description automatically generated

# **Exercise 3: Student System**

## **Source code:**

[1651---Advanced-Programming/Classwork3 at main · KieTuaNguyen/1651---Advanced-Programming (github.com)](https://github.com/KieTuaNguyen/1651---Advanced-Programming/tree/main/Classwork3)

## **Explain:**

**Class Student:**

This code defines a C# class called "Student" which has 3 private fields "name", "age", "grade" and 3 public properties "Name", "Age", "Grade" with getters and setters respectively. The class also has a constructor that takes three parameters "name", "age", "grade" and assigns them to the class fields. The class also has a method "ToString" which override the ToString method of the object class and formats the output of the student's name, age, and grade when the object is printed.

**Class Function:**

This code defines a C# class called "Function" which has a List of Student objects and several methods. The "AddAStudentToList" method prompts the user to enter a student's name, age, and grade, creates a new Student object with that information and adds it to the list of students. The "FindByName" method takes a string parameter "studentName" and loops through the list of students, comparing each student's name to the input name. If a student's name matches the input name, the student's information is printed to the console. The "Menu" method displays a menu with options for the user to add a student to the list, find a student by name, or exit.

**Class Program:**

This code defines a C# class called "Program" which contains the Main method. The Main method creates an instance of the "Function" class, displays a menu with options for the user, and prompts the user to enter their choice. The program uses a do-while loop to repeatedly display the menu, prompt the user for their choice, and perform the corresponding action (add a student, find a student by name, or exit the program) based on the user's choice. The menu will be displayed again after each iteration of the loop until the user chooses to exit the program. If the user enters 1, it calls the AddAStudentToList method in the function class, which prompts the user to enter the information of a student and adds the student to the list of students. If the user enters 2, it prompts the user to enter the name of the student, calls the FindByName method in the function class, which finds the student by name and prints the student's information. If the user enters 3, it end the program and print the message "End the program!" If the user enters any other number, it print the message "Invalid choice! Please try again!" and will repeat the process until the user enters 3.

## **Result:**

Text

Description automatically generated

A computer screen capture

Description automatically generated with medium confidence

# **Exercise 4: Hotel Reservation**

## **Source code:**

[1651---Advanced-Programming/Classwork4 at main · KieTuaNguyen/1651---Advanced-Programming (github.com)](https://github.com/KieTuaNguyen/1651---Advanced-Programming/tree/main/Classwork4)

## **Explain:**

This code defines a namespace "Classwork4" which contains an internal class "Program" which in turn contains a nested class "PriceCalculator". The PriceCalculator class has two enumerations "Season" and "Discount" which define the different seasons and discounts available. The "Season" enumeration has four members, Autumn, Spring, Summer and Winter, with values 1, 2, 3 and 4 respectively. The "Discount" enumeration has three members, None, SecondVisit and Vip, with values 0, 10 and 20 respectively. The PriceCalculator class also contains a static method "CalculateTotalPrice" that takes four parameters: pricePerDay, numberOfDays, season and discount. This method calculates the total price based on the price per day, the number of days, the season and the discount using the following steps:

* It calculates the "seasonMultiplier" by casting the enum type season to double.
* It calculates the "totalPrice" by multiplying pricePerDay, numberOfDays and seasonMultiplier.
* It calculates the "discountType" by casting the enum type discount to double and dividing by 100
* It returns the "totalPrice" minus the "totalPrice" multiplied by the "discountType".

In the Main method, it sets the values of pricePerDay, numberOfDays, season and discount, calls the CalculateTotalPrice method and uses Console.WriteLine to print the total price.

## **Result:**

Chart

Description automatically generated with medium confidence