# Working with Abstraction: Lab

Problems for exercises and homework for the ["C# OOP" course @ SoftUni](https://softuni.bg/opencourses/csharp-oop)".

You can test your solutions here: [https://judge.softuni.bg/Contests/Working-with-Abstraction-Lab](https://judge.softuni.bg/Contests/1495/Working-with-Abstraction-Lab)

## Rhombus of Stars

Create a program that reads a **positive** **integer** n as input and prints on the console a **rhombus** with size **n**:

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **input** | **output** |  | **input** | **output** |  | **input** | **output** |  | **input** | **output** |
| 1 | \* | 2 | \*  \* \*  \* | 3 | \*  \* \*  \* \* \*  \* \*  \* | 4 | \*  \* \*  \* \* \*  \* \* \* \*  \* \* \*  \* \*  \* |

#### Hint

Create a PrintRow() method to easily reuse code.

## Point in Rectangle

Create a class **Point** and a class **Rectangle**. The **Point** should hold **coordinates X** and **Y** and the **Rectangle** should hold 2 **Points** – its **top** **left** and **bottom** **right** corners. In the **Rectangle** class, you should implement a Contains(Point point) method that returns **true** or **false**, based on **whether** the **Point** given as **attribute** is **inside** or **outside** of the **Rectangle** object. Points **on** **the** **side** of a Square are considered **inside**.

### Input

* On the first line read the **coordinates** of the **top** **left** and **bottom** **right** corner of the **Rectangle** in the format: **"{topLeftX} {topLeftY} {bottomRightX} {bottomRightY}"**.
* On the second line, read an integer **N** and on the next **N** lines, read the **coordinates** of **points**.

### Output

* For each point, print out the result of the Contains() method.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **input** | **output** |  | **input** | **output** |  | **input** | **output** |
| 0 0 3 3  5  0 0  0 1  4 4  5 3  1 2 | True  True  False  False  True |  | 2 -3 12 3  4  8 -1  11 3  1 1  2 4 | True  True  False  False |  | 5 8 12 15  6  0 0  5 8  12 15  8 15  7 15  8 12 | False  True  True  True  True  True |

## Student System

You are given a **working** **project** for a small **Student** **System**, but the code is very poorly organized. Break up the code **logically** into **smaller** **functional** **units** – **methods** and **classes** and don’t break the functionality.

The program supports the following commands:

* "**Create {studentName} {studentAge} {studentGrade}**"– creates a new student and adds them to the repository.
* "**Show {studentName}**" – prints on the console information about a student in the format:  
  "**{studentName} is {studentAge} years old. {commentary}"**, where the **commentary** is based on the student’s grade.
* "**Exit"** – closes the program.

Following the **next rules** will help you to **easily solve the problem**:

* You should have **only one class** in **only one file**!
* You should **remove any unnecessary data** (**usings**, **fields**, **properties**, **constants**, etc.)!
* You can use **auto-properties** if you don’t have any **validation** or **encapsulation** inside this property!
* **Most collections** used inside the class **should not be exposed to public** because of its **vulnerability**!
* You **should break the code into smaller units** (**methods** with **appropriate return type**)!
* You should be **consistent** with **the naming** and **the ordering** of **the elements** of the class!

**Do not** add any **extra validation** or **functionality** to the app!

### Examples

|  |  |
| --- | --- |
| **input** | **output** |
| Create Pesho 20 5.50  Create Mimi 18 4.50  Create Gosho 25 3  Show Pesho  Show Mimi  Exit | Pesho is 20 years old. Excellent student.  Mimi is 18 years old. Average student. |

## Hotel Reservation

Create a **static** class **PriceCalculator** that calculates the total price of a holiday, given the **price** **per** **day**, **number** **of** **days**, the **season** and a **discount** **type**.The **discount** **type** and **season** should be **enums**.

You can create a **static class** holding **only one static method** inside. In order to get the **necessary data** for the calculations **inside the class**, you can **pass the data** as an **arguments to the static method**. You are **free** to **implement** any calculation **logic** inside the **method** on the condition that **your output is correct**.

Use your **Main()** method to **read the input** and **print on the console**, but use the **static** **GetTotalPrice()** method in our **static** class **PriceCalculator** in order to **calculate the total price** of the holiday.

The price per day will be multiplied depending on the season by:

* **1** during **Autumn**
* **2** during **Spring**
* **3** during **Winter**
* **4** during **Summer**

The discount is applied to the total price and is one of the following:

* 20% for VIP clients
* 10% for clients, visiting for a second time
* 0% if there is no discount

### Input

On a **single** **line** you will receive all the **information** about the **reservation** in the format:

**"{pricePerDay} {numberOfDays} {season} {discountType}"**, where:

* The price per day will be a valid decimal in the range [0.01…1000.00]
* The number of days will be a valid integer in range [1…1000]
* The season will be one of: **Spring**, **Summer**, **Autumn**, **Winter**
* The discount will be one of: **VIP**, **SecondVisit**, **None**, but it **can** also **be** **omitted** from the input

### Output

On a **single** **line**, print the **total** **price** of the **holiday**, rounded to **2** **digits** after the decimal separator.

### Examples

|  |  |
| --- | --- |
| **input** | **output** |
| 50.25 5 Summer VIP | 804.00 |
| 40 10 Autumn SecondVisit | 360.00 |
| 120.20 2 Winter | 721.20 |