# Exercise – Static Members

This document defines the **exercise assignments** for the ["C# OOP Basics" course @ Software University](https://softuni.bg/csharp-basics-oop). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/#!/List/ByCategory/43/CSharp-OOP-Basics-Exercises).

## Problem 1. Students

Define class **Student**. Add **string field** for a student’s **name** that you are going to receive as a console input. Then add a **static Integer field** to **keep track of how many students’ instances are created**. Initialize the static field with **0 (zero)** and **increment in the constructor**. When you receive command **“End”** stop reading more students names and print their total count on the console.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| Atanas  Atanas  End | 2 |  | Minka  End | 1 |

## Problem 2. Unique Student Names

Define class **Student** containing a single **field – name**. Now Define class **StudentGroup** with **HashSet<Student>** field that will keep all unique students. You are going to receive user input containing student’s names as single parameter on the line until you receive command **“End”**. Create new instances of Students class and **keep track of all unique names** using static counter within the **StudentGroup** class. Then print the **count of unique names**.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| Atanas  Atanas  End | 1 |  | Minka  End | 1 |  | Minka  Minka  Atanas  Nasko  End | 3 |

## Problem 3. Temperature Converter

Create a program that **converts temperature** **from Celsius to Fahrenheit and vice versa**. Use **static methods**. The input data will be in format: {temperature} {unit}. Temperatures will be in **integer** number and units will be one of these two values: Celsius / Fahrenheit***.*** Output value must be **double value** following of empty space and **the converted unit**. You are going to receive input, until you receive command **“End”.** The output must be formatted **2 digits after floating point.**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 24 Celsius  101 Fahrenheit  End | 75.20 Fahrenheit  38.33 Celsius |

## Problem 4. Beer Counter

Define class **BeerCounter** holding static field **beerInStock** that shows how many beers you bought and static field **beersDrankCount** that shows how many beers you have drunk. Manipulate the static fields through static methods **BuyBeer(int bottlesCount)** and **DrinkBeer(int bottlesCount)**. On every input line you will get pair of beers you **bought** and beers you **drank**, until you receive command **“End”.**

* **BuyBeer** – add beers to the beers in stock
* **DrinkBeer** – add beers to the drunk beers counter and subtract beers in stock

After that print **beersInStock** and **beersDrankCount** on the same line separated by 1 space.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 50 49  9 10  End | 0 59 |

## Problem 5. Animal Clinic

Define two classes: **Animal** (**name**, **breed**) and **AnimalClinic** (static field **patientId**, static field **healedAnimalsCount** and static field **rehabilitedAnimalsCount**). You will be given animal data (name and breed) and information whether the animal should be healed or rehabilitated. **Keep track** on the **rehabilitated animals,** on the **healed animals** and **overall patients**. If the animal has been **healed**, you need to **print on the console** the following message:

Patient {patientID} [{name} ({breed})] has been healed!

Otherwise print:

Patient {patientID} [{name} ({breed})] has been rehabilitated!

You will receive information about animals until you receive command **“End”.**

After you receive command **“End”** print total healed animals and total rehabilitated animals in format:

Total healed animals: {count}

Total rehabilitated animals: {count}

After that you will receive one of the following commands heal or rehabilitate and you must **print all** the names and breed of the **healed** or **rehabilitated** animals in format {name} {breed} each animal on new line.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Toshko Terrier heal  End  heal | Patient 1: [Toshko(Terrier)] has been healed!  Total healed animals: 1  Total rehabilitated animals: 0  Toshko Terrier |
| **Input** | **Output** |
| Toshko Terrier rehabilitate  Toshko Terrier rehabilitate  End  rehabilitate | Patient 1: [Toshko(Terrier)] has been rehabilitated!  Patient 2: [Toshko(Terrier)] has been rehabilitated!  Total healed animals: 0  Total rehabilitated animals: 2  Toshko Terrier  Toshko Terrier |
| **Input** | **Output** |
| Toshko Terrier heal  Goshko Bulldog rehabilitate  End  rehabilitate | Patient 1: [Toshko (Terrier)] has been healed!  Patient 2: [Goshko (Bulldog)] has been rehabilitated!  Total healed animals: 1  Total rehabilitated animals: 1  Goshko Bulldog |

## Problem 6. Planck Constant

Create class **Calculation**. Define static constant with value **6.62606896e-34** (Planck constant) and **3.14159** (Pi). Add **static method** that returns reduced Planck constant by the formula:

{Planck constant} / (2 \* {Pi constant})

Print the result of the method on a **single line on the console**. **Do not format** in any way the **result.**

## Problem 7. Basic Math

Define **MathUtil** class that supports **basic** mathematical operations:

* **Sum <first number> <second number>**
* **Subtract <first number> <second number>**
* **Multiply <first number> <second number>**
* **Divide <dividend> <divisor>**
* **Percentage <total number> <percent of that number>**

Use **static methods** and make sure that the application will work with **floating point numbers**.

Read from the console until you receive command **“End”.** Results must be formatted with **2 digits after the floating point.**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Sum 5 5  Multiply 5.5 11  Percentage 1500 99  Divide 12.24 2  Subtract 10.6 0.6  End | 10.00  60.50  1485.00  6.12  10.00 |

## Problem 8. Shapes Volume

Define class **TriangularPrism** that has **base side**, **height** **from base side** and **length**. Define class **Cube** that has **side** **length** and class **Cylinder** that has **radius** and **height**. Define class **VolumeCalculator** that holds **static methods** for calculating the volume of these three figures. The input will be read from the console until command **“End”** is received and will be in some of these formats:

* **TriangularPrism <base side> <height> <length>**
* **Cube <side length>**
* **Cylinder <radius> <height>**

The volume in the **output** must be **rounded 3 digits after** the floating point.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Cube 5  Cylinder 5 11.4  TrianglePrism 1 2 3  End | 125.000  895.354  3.000 |