**Exercise: Encapsulation**

Problems for exercise and homework for the ["C# OOP" course @ SoftUni"](https://softuni.bg/trainings/3585/csharp-oop-february-2022).

You can check your solutions here: <https://judge.softuni.org/Contests/1498/Encapsulation-Exercise>

* **Pizza Calories**

A pizza is made of dough and different toppings. You should model a **class Pizza,** which should have a **name**, **dough,** and **toppings** as fields. Every type of **ingredient** should have its **class**. Every ingredient has different properties: the **dough** can be white or wholegrain and in addition, it can be crispy, chewy, or homemade. The **topping** can be of type meat, veggies, cheese, or sauce. **Every ingredient** should weigh grams and a method for **calculating** its calories according to its type. Calories per gram are calculated through **modifiers**. Every ingredient has 2 calories per gram as a **base** and a **modifier** that **gives** the **exact** calories. For example, a white dough has a modifier of 1.5, a chewy dough has a modifier of 1.1, which means that a **white** **chewy** dough, weighing **100** **grams** will have 2 \* 100 \* 1.5 \* 1.1 = 330.00 **total** **calories**.

**Your job** is to model the classes in such a way that they are **properly encapsulated** and to provide a **public** method for every pizza that **calculates its calories according to the ingredients it has**.

**Step 1. Create a Dough Class**

The base ingredient of a **Pizza** is the dough. First, you need to create a **class** for it. It has a **flour type,** which can be **white** or **wholegrain**. In addition, it has a **baking technique,** which can be **crispy**, **chewy,** or **homemade**. The dough should weigh grams. The calories per gram of dough are calculated **depending** on the **flour type** and the **baking technique**. Every **dough** has **2 calories** **per** **gram** as a base and a **modifier** that gives the exact calories. For example, a white dough has a modifier of 1.5, a chewy dough has a modifier of 1.1, which means that a **white** **chewy** **dough**, weighing **100** **grams** will have (2 \* 100) \* 1.5 \* 1.1 = 330.00 **total** **calories**. You are given the **modifiers** below:

**Modifiers:**

* **White - 1.5**
* **Wholegrain - 1.0**
* **Crispy - 0.9**
* **Chewy - 1.1**
* **Homemade - 1.0**

Everything that the class should expose is a **getter** for the **calories per gram**. Your task is to create the class with a proper **constructor**, **fields**, **getters,** and **setters**. Make sure you use the **proper access modifiers**.

**Step 2. Validate Data for the Dough Class**

Change the internal logic of the **Dough** class by adding a **data validation** in the **setters**.

Make sure that if an **invalid flour type** or an **invalid baking technique** is given a proper **Exception** is thrown with the message **"Invalid type of dough."**.

The allowed weight of dough is in the **range** [1..200] grams. If it is **outside** of this **range** throw an **Exception** with the message **"Dough weight should be in the range [1..200]."**.

**Exception Messages**

* **"Invalid type of dough."**
* **"Dough weight should be in the range [1..200]."**

Make a test in your main method that reads Doughs and prints their calories until an "**END**" command is given.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Dough White Chewy 100  END | 330.00 |
| Dough Tip500 Chewy 100  END | Invalid type of dough. |
| Dough White Chewy 240  END | Dough weight should be in the range [1..200]. |

**Step 3. Create a Topping Class**

Next, you need to create a **Topping class**. It can be of four different types - **meat**, **veggies**, **cheese**, or a **sauce**. A **Тopping** weights grams. The **calories per gram** of topping are **calculated depending on its type**. The **base calories** **per gram** are **2**. Every different type of topping has a **modifier**. For example, **meat** has a **modifier of 1.2**, so a **meat** topping will have **1.2 calories per gram** (1 \* 1.2). Everything that the class should expose is a **getter** for **calories per gram**. You are given the **modifiers** below:

Modifiers:

* **Meat - 1.2;**
* **Veggies - 0.8;**
* **Cheese - 1.1;**
* **Sauce - 0.9;**

Your task is to create the class with a **proper** **constructor**, **fields**, **getters,** and **setters**. Make sure you use the **proper** **access modifiers**.

**Step 4. Validate Data for the Topping Class**

Change the internal logic of the **Topping** class by adding **data validation** in the **setter**.

Make sure the **Тopping** is one of the provided types, otherwise throw a proper **Exception** with the message **"Cannot place [name of invalid argument] on top of your pizza."**.

The allowed weight of a **Тopping** is in the range [1..50] grams. If it is **outside of this range** throw an **Exception** with the message **"[Topping type name] weight should be in the range [1..50]."**.

**Exception Messages**

* **"Cannot place [name of invalid argument] on top of your pizza."**
* **"[Topping type name] weight should be in the range [1..50]."**

Make a test in your main method that reads a single dough and a topping after that and prints their calories.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Dough White Chewy 100  Topping meat 30  END | 330.00  72.00 |
| Dough White chewy 100  Topping Krenvirshi 500  END | 330.00  Cannot place Krenvirshi on top of your pizza. |
| Dough White Chewy 100  Topping Meat 500  END | 330.00  Meat weight should be in the range [1..50]. |

**Step 5. Create a Pizza Class!**

A **Pizza** should have a **name**, some **toppings**, and **dough**. Make use of the **two classes you made earlier**. In addition, a **Pizza** should have **public getters** for its **name**, the **number of toppings**, and the **total calories**. The **total calories** are **calculated by summing the calories of all the ingredients a Pizza has**. Create the class using a **proper constructor**, expose a **method** for **adding a topping**, a **public** **setter** for the dough, and a **getter**for the **total calories**.

The input for a **Pizza** consists of **several** **lines**. On the first line is the **Pizza name** and on the second line, you will get input for the **dough**. On the next lines, you will receive every topping the **Pizza** has.

If the creation of the **Pizza** was **successful,** print on a single line the name of the **Pizza** and the **total calories** it has.

**Step 6. Validate Data for the Pizza Class**

The **name** of the **Pizza** should **not** be an **empty string**. In addition, it should **not be longer than 15 symbols**. If it does not fit, throw an **Exception** with the message **"Pizza name should be between 1 and 15 symbols."**.

The **number of toppings** should be in the range [0..10]. If not, throw an **Exception** with the message **"Number of toppings should be in range [0..10]."**.

Your task is to print the **name** of the **Pizza** and the **total** **calories** it has according to the examples below.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Pizza Meatless  Dough Wholegrain Crispy 100  Topping Veggies 50  Topping Cheese 50  END | Meatless - 370.00 Calories. |
| Pizza Burgas  Dough White Homemade 200  Topping Meat 123  END | Meat weight should be in the range [1..50]. |
| Pizza Bulgarian  Dough White Chewy 100  Topping Sauce 20  Topping Cheese 50  Topping Cheese 40  Topping Meat 10  Topping Sauce 10  Topping Cheese 30  Topping Cheese 40  Topping Meat 20  Topping Sauce 30  Topping Cheese 25  Topping Cheese 40  Topping Meat 40  END | Number of toppings should be in range [0..10]. |
| Pizza Bulgarian  Dough White Chewy 100  Topping Sirene 50  Topping Cheese 50  Topping Krenvirsh 20  Topping Meat 10  END | Cannot place Sirene on top of your pizza. |

* **\*\*Football Team Generator**

A football **Team** has a variable **number of players**, a **name,** and a **rating**. A **Player** has a **name** and **stats,** which are the basis for his skill level. The stats a player has are **endurance**, **sprint**, **dribble**, **passing,** and **shooting**. Each stat can be an **integer** in the range [0..100]. The overall **skill** **level** of a **player** is calculated as the **average** of his **stats**. Only the **name** of a player and his **stats** should be visible to the entire outside world. **Everything** **else** should be **hidden**.

A **Team** should expose a **name**, a **rating** (calculated by the average skill level of all players in the team and **rounded** to the **integer** part only), and **methods** for **adding** and **removing** **players**.

Your task is to **model** the **Team** and the **Player** classes following the proper principles of **Encapsulation**. Expose **only** the properties that need to be visible and **validate** **data** appropriately.

**Input**

Your application will receive commands until the "**END**" command is given. The command can be one of the following:

* **"Team;{TeamName}"** - add a new **Team**;
* **"Add;{TeamName};{PlayerName};{Endurance};{Sprint};{Dribble};{Passing};{Shooting}"** - add a new **Player** to the **Team**;
* **"Remove;{TeamName};{PlayerName}"** -remove the **Player** from the **Team**;
* **"Rating;{TeamName}"** - print the **Team** rating, rounded to an integer.

**Data Validation**

* A name cannot be null, empty, or white space. If not, print **"A name should not be empty."**
* Stats should be in the range 0..100. If not, print **"[Stat name] should be between 0 and 100."**
* If you receive a command to remove a missing **Player**, print **"Player [Player name] is not in [Team name] team."**
* If you receive a command to add a **Player** to a missing **Team**, print **"Team [team name] does not exist."**
* If you receive a command to show stats for a missing **Team**, print **"Team [team name] does not exist."**

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Team;Arsenal  Add;Arsenal;Kieran\_Gibbs;75;85;84;92;67  Add;Arsenal;Aaron\_Ramsey;95;82;82;89;68  Remove;Arsenal;Aaron\_Ramsey  Rating;Arsenal  END | Arsenal - 81 |
| Team;Arsenal  Add;Arsenal;Kieran\_Gibbs;75;85;84;92;67  Add;Arsenal;Aaron\_Ramsey;195;82;82;89;68  Remove;Arsenal;Aaron\_Ramsey  Rating;Arsenal  END | Endurance should be between 0 and 100.  Player Aaron\_Ramsey is not in Arsenal team.  Arsenal - 81 |
| Team;Arsenal  Rating;Arsenal  END | Arsenal - 0 |