Упражнение: Допълнителни задачи

1. Първи и резервен Отбор

Създайте клас Team. Добавете към него всички обекти person, които въвеждате. Обектите person, по-млади от 40 отиват в първи отбор, другите - в резервен. На края изведете броя на участниците в първия и резервния отбор.

Класът трябва да има private полета за:

- name: string
- firstTeam: List<Person>
- reserveTeam: List<Person>

Класът трябва да има constructors:

- Team(string name)

Трябва да има също public методи за:

- AddPlayer(Person person): void
- FirstTeam: IReadOnlyCollection
- ReserveTeam: IReadOnlyCollection

Трябва да можете да използвате класа:

```
private string name;
private List<Person> firstTeam;
private List<Person> reserveTeam;

public Team(string name)
{
    this.name = name;
    this.firstTeam = new List<Person>();
    this.reserveTeam = new List<Person>();
}
```

Не бива да използвате клас като този:

```
Team team = new Team("Gorno Nanadolnishte");
foreach (var player in persons)
{
    if (player.Age < 40)
        team.FirstTeam.Add(player);
    else
        team.ReserveTeam.Add(player);
}
```

Примери

Вход	Изход
5 Asen Ivanov 20 2200 Boiko Borisov 57 3333 Ventsislav Ivanov 27 600 Grigor Dimitrov 25 666.66 Boiko Angelov 35 555	First team have 4 players Reserve team have 1 players

Решение

Добавете нов клас Team. Неговите полета и конструктор да бъдат например такива:

```
private string name;
private List<Person> firstTeam;
private List<Person> reserveTeam;

public Team(string name)
{
    this.name = name;
}
```

Свойствата за FirstTeam and ReserveTeam имат само getters:

```
public IReadOnlyCollection<Person> FirstTeam
{
    get { return this.firstTeam.AsReadOnly(); }
}

public IReadOnlyCollection<Person> ReserveTeam
{
    get { return this.reserveTeam.AsReadOnly(); }
}
```

Ще има само един метод, който добавя играчи към отбора

```
public void AddPlayer(Person player)
{
    if (player.Age < 40)
    {
        firstTeam.Add(player);
    }
    else
    {
        reserveTeam.Add(player);
    }
}
```

2. **Създаване на футболен отбор

Футболен отбор има променлив брой играчи, име и рейтинг. Един играч има име и статистика, които са в основата на неговото ниво на умения. Един играч има статистика са издръжливост, Спринт, дрибъл, подавания и стрелба. Всяка статистика може да бъде в диапазона [0..100]. Общото ниво на умение на играч се изчислява като средна стойност на статистиките си. Само името на играча и неговата статистика трябва да бъдат видими за всички от външния свят. Всичко останали данни трябва да бъдат скрити. Отборът трябва да показва име, рейтинг (изчислена от нивата на средните умения на всички играчи в отбора и закръглена до цяло число) и методи за добавяне и премахване на играчи. Вашата задача е да моделирате екипа и играчите, чрез правилното използване на принципите на капсулиране. Покажи само свойствата, които трябва да бъдат видими и валидирайте данните по подходящ начин.

Валидация на данните

- Името не трябва да е празно, null, empty или да е само от интервали. Иначе, изведете "A name should not be empty. "
- Stats трябва да е в обхвата 0..100. Иначе изведете "[Име на статистиката] should be between 0 and 100. "
- Ако получите команда да премахнете липсващ играч, изведете "Player [име на играча] is not in [Име на отбора] team. "
- Ако получите команда да добавите играч към липсващ отбор, изведете "Team [име на отбор] does not exists."
- Ако получите команда за показване на stats за липсващ отбор, изведете "Team [име на отбор] does not exists."

Примери

Вход	Изход
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

3. Калории на Pizza

Пицата е изработена от тесто и различни гарнитури. Вие трябва да създадете класа Pizza, който трябва да има име, тесто и гарнитурата като полета. Всеки вид на съставка трябва да има свой собствен клас. Всяка съставка има различни свойства: тестото може да бъде бяло или пълнозърнесто и освен това то може да бъде хрупкави, chewy или домашно приготвени. Гарнитурата може да бъде от тип месо, зеленчуци, сирене или сос. Всяка съставка трябва да има тегло в грамове и метод за изчисляване на калориите му, според типа си. Калории на грам се изчисляват чрез модификатори. Всяка съставка има 2 калории на грам като база и Модификатор, която дава точна калории. Например, бяло тесто има Модификатор на 1.5, chewy тестото има Модификатор на 1.1, което означава, че бял chewy тесто 100 грама ще има 100 * 1,5 * 1.1 = 330,00 общо калории.

Вашата работа е да създадете класовете по такъв начин, че те правилно да са капсуловани и да предоставят публичен метод за всяка пица която изчислява калориите в зависимост от съставките си.

Step 1. Създайте клас Dough

Основната съставка на пицата е тестото. Първо трябва да създадете един клас за него. Той има тип брашно , който може да бъде бял или пълнозърнест. В допълнение има техника на печене, която може да бъде за хрупкави, жилави или домашно приготвени теста. Тестото трябва да има тегло в грамове. Калории на грам от тестото се изчисляват според типа на брашното и техниката на втастване. Всеки вид тестото има 2 калории на грам като база и Модификатор, който дава точните калории. Например, бялото тесто има Модификатор 1.5, жилавото тесто има Модификатор 1.1, което означава, че бяло жилаво тесто с тегло 100 грама ще има (2 * 100) * 1,5 * 1.1 = 330,00 общо калории. По-долу Ви се предоставят модификатори:

- White – 1.5;
- Wholegrain – 1.0;
- Crispy – 0.9;
- Chewy – 1.1;
- Homemade – 1.0;

Всичко, което трябва да е видимо за класа е getter за калории на грам. Вашата задача е да се създаде клас с подходящ конструктор, полета, getters и setters. Проверете дали използвате правилните модификатори за достъп.

Step 2. Валидирайте данните за класа Dough

Променете вътрешната логика на класа Dough class чрез добавяне на валидация във setters.

Подсигурете при невалиден вход на типа брашно flour type или невалидна техника на печене подходящо изключение да се връща със съобщение "Invalid type of dough.".

Допустимото тегло на тестото е в диапазона [1..200] грама. Ако е извън диапазона да се върне изключение със съобщение "Dough weight should be in the range [1..200].".

Съобщения на изключенията

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

Направете тест на метода main, който въвежда различни видове теста и извежда техните калории, докато се въведе команда "Край".

Примери

Вход	Изход
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. Създайте клас Topping

След това трябва да създаде класа Topping. Той може да бъде четири различни вида – месо, зеленчуци, сирене или сос. Гарнитурата има тегло в грамове. Калориите на грам гарнитура се изчисляват в зависимост от типа ѝ. Базовите калории на грам са 2. Всеки различен вид гарнитура има модификатор. Например месото има Модификатор 1.5, така че месната гарнитура ще има 1.5 калории на грам (1 * 1.5). Всичко, което трябва да изложи класа е getter за калории на грам. По-долу ви се предоставени модификаторите:

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

Вашата задача е да се създаде клас с подходящ конструктор, полета, getters и setters. Проверете дали използвате правилни модификатори за достъп.

Step 4. Валидиране на данните за класа Topping

Сменете вътрешната логика на класа Topping, като добавите валидация на данните в setter-а.

Уверете се, че гарнитурата е измежду предоставените типове, в противен случай изведете подходящо изключение със съобщение "Cannot place [name of invalid argument] on top of your pizza".

Теглото на гарнитурите е в диапазона [1..50] грама. Ако е извън този диапазон да се върне изключение със съобщението "[Topping type name] weight should be in the range [1..50].".

Съобщения на изключенията

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

Направете тест на метода main, който въвежда количество тесто и гарнитури и след това извежда техните калории.

Примери

Вход	Изход
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. Създайте клас Pizza !.

Пицата трябва да има име, няколко гарнитурата и тесто. Използвайте двата класа, които сте направили по-рано. Пицата трябва да имат публични getters за нейното име, брой гарнитури и общото количество калории. Общото количество калории се изчисляват чрез сумиране на калориите на всички съставки, които пицата има. Създайте клас, използвайки подходящ конструктор, направете метод за добавяне на гарнитура, публични getters за тестото и за общото количество калории. Входът за пицата се състои от няколко реда. На първия ред е името на пица и броя на гарнитурите, които има. На втория ред да се въвежда тестото. На следващите редове ще получите всяка гарнитура на пицата. Броят на редовете за гарнитурите се въвежда на първия ред. Ако създаването на пица е успешно да се изведе на един ред името на пицата и общото количество калории в нея.

Step 6. Валидиране на данни за клас Pizza

Името на pizza не трябва да е празен низ. Също не трябва да е повече от 15 символа. Ако не отговаря на това условие, се връща изключение със съобщение "Pizza name should be between 1 and 15 symbols."

Броят на гарнитурите трябва да е в диапазона [0...10]. иначе се връща изключение със съобщение "Number of toppings should be in range [0..10]."

Вашата задача е да изведете името на пицата и общото количество калории в нея, според примера по-долу:

Примери

Вход	Изход
Pizza Meatless 2 Dough Wholegrain Crispy 100 Topping Veggies 50 Topping Cheese 50 END	Meatless – 370.00 Calories.
Pizza Meatfull 5 Dough White cheWy 200 Topping Meat 50 Topping Cheese 50 Topping meat 20 Topping sauce 10	Meatfull – 1028.00 Calories.

`Topping Meat 30` `END`	

`Pizza Bulgarian 20`
`Dough Tip500 Balgarsko 100`
`Topping Sirene 50`
`Topping Cheese 50`
`Topping Krenvirsh 20`
`Topping Meat 10`
`END`
	`Number of toppings should be in range [0..10].`

`Pizza Bulgarian 2`
`Dough Tip500 Balgarsko 100`
`Topping Sirene 50`
`Topping Cheese 50`
`Topping Krenvirsh 20`
`Topping Meat 10`
`END`
	`Invalid type of dough.`

`Pizza Bulgarian 2`
`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.`

4. * Ферма за животни допълнение

Към разглежданата задача направете валидацията, която беше пропусната във второто упражнение и допълнете проекта

Step 3. Подходящо валидиране на данните

Валидирайте името на обектите от клас Chiken (не може да е null, празно или да съдържа само интервали). В случай на невалидно име, да се върне изключение със следното съобщение "Name cannot be empty."

Валидирайте свойството age, с минималната и максималната допустими стойности. В случай на невалидна възраст, върнете изключение със съобщение "Age should be between 0 and 15."

Не забравяйте да обработвате правилно вероятно получените изключения.