# Упражнения: Шаблонните класове, интерфейси и методи

## Кутия за всичко

Създайте клас **Box<>**, който може да съхранява всичко.

Той трябва да има два публични метода:

* void Add(element item)
* element Remove()

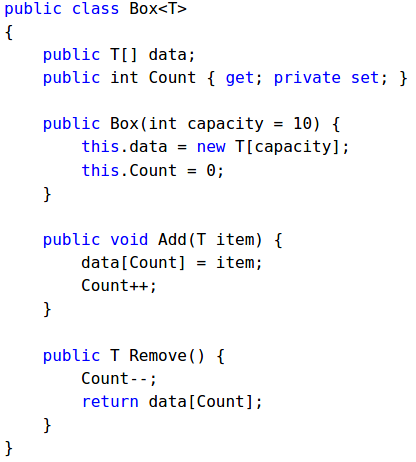
Добавянето трябва да добавя новото най-отгоре. Премахването да взима най-горния.

### Примери

|  |
| --- |
| public static void Main(string[] args)  {  Box<int> box = new Box<int>();  box.Add(1);  box.Add(2);  box.Add(3);  Console.WriteLine(box.Remove());  box.Add(4);  box.Add(5);  Console.WriteLine(box.Remove());  } |

### Подсказки

Използвайте формата Box<T>, за да създадете шаблонен клас:



## Буркан за всичко

Създайте шаблонен клас Jar, който може да бъде инициализиран с **произволен** тип и да **съхранява** стойността. **Предефинирайте** метода **ToString()** да отпечатва типа и стойността на съхраняваните данни във формат **{class full name: value}.**

### Бележка

Класът се използва в следващите задачи. За да вземете пълното име на класа, използвайте свойството [.GetType().FullName](https://msdn.microsoft.com/en-us/library/system.type.fullname(v=vs.110).aspx).

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 123123 | System.Int32: 123123 |
| life in a box | System.String: life in a box |

## Буркан за низове

Използвайте класа, създаден в предната задача и го тествайте с класа **System.String.** На първия ред ще получите **n** - броят на низовете, които да прочетете от конзолата. На следващите **n** реда ще са самите низове. За всеки от тях създайте буркан и извикайте неговия метод **ToString()**, за да отпечатате съдържанието му на конзолата.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 2  life in a box  box in a life | System.String: life in a box  System.String: box in a life |

## Буркан за цели числа

Като предната задача, но този път тествайте вашия универсален буркан със съдържание цели числа.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  7  123  42 | System.Int32: 7  System.Int32: 123  System.Int32: 42 |

## Създател на масиви

Създайте клас ArrayCreator с метод и едно-единствено предефиниране:

* static T[] Create(int length, T item);

Методът трябва да връща масив с указаната дължина, в който на всеки елемент е присвоена подадената стойност.

### Примери

|  |
| --- |
| static void Main(string[] args)  {  string[] strings = ArrayCreator.Create(5, "Pesho");  int[] integers = ArrayCreator.Create(10, 33);  } |

## Шаблонен метод за размяна на буркани

Създайте шаблонен метод, който получава масив с буркани от произволен тип и разменя местата на елементите на две указани позиции.

Както в предните примери, прочетете **n** на брой буркана от тип String и ги добавете в масива. Тук обаче на следващия ред ще получите команда за размяна, състояща се от два индекса. Използвайте метода, който създадохте, за да размените бурканите с позиция, съответстваща на подадените индекси и накрая отпечатайте всички буркани в масива.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  Pesho  Gosho  Swap me with Pesho  0 2 | System.String: Swap me with Pesho  System.String: Gosho  System.String: Pesho |

## Шаблонен метод за размяна на цели числа

Като предната задача, но този път тествайте вашия масив с универсални буркани с цели числа.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  7  123  42  0 2 | System.Int32: 42  System.Int32: 123  System.Int32: 7 |

## Шаблонен метод за броене на низове

Създайте **метод** който получава като параметър **масив от кой да е от типовете данни, които могат да бъдат сравнявани** и **един елемент от същия тип**. Методът трябва да **връща броя на елементите, които са по-големи по стойност от подадения елемент**. **Променете вашия клас Jar** така, че да поддържа **сравняване на стойностите** на съхранените данни.

На първия ред ще получите **n** - броят на елементите, които да добавите в масива. На следващите **n** реда ще получите самите елементи. На последния ред ще е стойността на елемента, спрямо който ще сравнявате всеки един елемент от масива.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  aa  aaa  bb  aa | 2 |

## Шаблонен метод за броене на дробни числа

Като предната задача, но този път тествайте вашия списък с универсални кутии с числа от тип **double**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  7.13  123.22  42.78  7.55 | 2 |

## Универсална везна

Създайте клас **Scale<T>**, Съдържа два елемента: left и right. Получава ги чрез своя единствен конструктор:

* Scale(T left, T right)

Везната трябва да има един-единствен метод:

* T getHeavier()

По-големият от двата елемента е по-тежък. Методът трябва да връща default(T), ако елементите са еднакви.

## Подобрен списък

Създайте универсална структура данни, която може да съхранява **произволен** тип данни, който **може** да бъде сравняван. Реализирайте функциите:

* **void Add(T element)**
* **T Remove(int index)**
* **bool Contains(T element)**
* **void Swap(int index1, int index2)**
* **int CountGreaterThan(T element)**
* **T Max()**
* **T Min()**

Създайте команден интерпретатор, който чете команди и променя подобрения списък, който сте създали. Инициализирайте списъка да съхранява низове. Реализирайте командите:

* **Add <element>** - добавя даден елемент в края на списъка
* **Remove <index>** - премахва елемента, намиращ се на указаната позиция
* **Contains <element>** - отпечатва дали списъкът съдържа даден елемент **(True или False)**
* **Swap <index> <index>** - разменя местата на елементите с указаните индекси
* **Greater <element>** - преброява елементите, които са по-големи от подадения елемент и отпечатва техния брой
* **Max** - отпечатва максималния елемент от списъка
* **Min** - отпечатва минималния елемент от списъка
* **Print** - отпечатва всички елементи в списъка, всеки на отделен ред
* **END** - приключва с четенето на командите

**Няма** да има **никакви** невалидни команди във входните данни.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Add aa  Add bb  Add cc  Max  Min  Greater aa  Swap 0 2  Contains aa  Print  END | cc  aa  2  True  cc  bb  aa |

## Сортиране на подобрения списък

Разширете решението на предната задача чрез създаване на допълнителен **клас** **Sorter**. Той трябва да има един-единствен статичен **метод** **Sort()**, който може да сортира обекти от тип **CustomList**, съдържащи данни от произволен тип, който подлежи на сравняване. **Разширете списъка с команди**, така че да поддържа една допълнителна команда Sort:

* **Sort** - сортира елементите в списъка в нарастващ ред.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Add cc  Add bb  Add aa  Sort  Print  END | aa  bb  cc |

## \*Обхождане на подобрения списък

За всяка от командите за отпечатване вероятно сте използвали цикъл **for**. Разширете вашия подобрен списък, така че да реализира интерфейса **IEnumerable<T>.** Това би позволило да обхождата вашия списък с командата foreach.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Add aa  Add bb  Add cc  Max  Min  Greater aa  Swap 0 2  Print  END | cc  aa  2  cc  bb  aa |

## Tuple

Има нещо много странно в C#. Нарича се [**Tuple**](https://msdn.microsoft.com/en-us/library/system.tuple(v=vs.110).aspx). Това е клас, който може да съхранява няколко обекта, но нека се фокусираме върху тип Tuple, който съхранява два обекта. Първият е “**item1**”, а вторият - “**item2**”. Това е нещо подобно на **KeyValuePair** с изключение на това, че **просто съхранява елементи,** които не са **нито ключове, нито стойности**. Странността идва от факта, че нямате никаква идея какво съдържат тези елементи. Името на класа нищо не ви подсказва, методите които има - също. И така, нека си представим, че по някаква причина бихме искали да се опитаме сами да направим такъв клас, ей така - просто за да упражним шаблоните.

Задачата: създайте клас “**Tuple**”, съдържащ два обекта. Както споменахме, първият ще е “**item1**”, а вторият - “**item2**”. Тънкостта тук идва от това, да накараме класа да поддържа шаблони. Това ще рече когато създаваме нов обект от клас “**Tuple**”, трябва да начин изрично да укажем типа и на двата елемента поотделно.

### Вход

Входните данни включват три реда:

* Първият съдържа името на човек и адресът му. Те са отделени с интервал(и). Вашата задача е да ги прочетете в tuple-а и да ги отпечатате на конзолата. Форматът на входните данни е:

**<<first name> <last name>>** **<address>**

* Вторият ред съдържа **име** на човек и **количеството бира** (int), което той може да изпие. Формат:

**<name> <liters of beer>**

* Последният ред съдържа **Integer** и **Double**. Формат:

**<Integer> <Double>**

### Изход

* Отпечатайте елементите на tuple-а във формат: {**item1**} -> {**item2**}

### Ограничения

Използвайте добрите практики, които сме учили. Създайте клас и му добавете getters и setters за клас-променливите му. Входните данни ще са валидни, няма нужда изрично да ги проверявате!

### Пример

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Sofka Tripova Stolipinovo  Az 2  23 21.23212321 | Sofka Tripova -> Stolipinovo  Az -> 2  23 -> 21.23212321 |

## Threeuple

Създайте клас **Threeuple**. Както показва и името му, той ще съдържа нещо повече от двойка обекти. Задачата е също лесна, нашият **Threeuple** трябва да съдържа **три обекта**. Направете си getter-и и setter-и. Може даже да наследите предния клас.

### Вход

Входните данни се състоят от три реда:

* Първият ред съдържа име, адрес и град, във формат:

**<<first name> <last name>> <address> <town>**

* Вторият ред съдържа име, литри бира и булева променлива със стойност **drunk** или **not**. Форматът е:

**<name> <liters of beer> <drunk or not>**

* Третият ред съдържа име, наличност по банковата сметка (double) и име на банката. Форматът е:

**<name> <account balance> <bank name>**

### Изход

* Отпечатайте Threeuple обектите във формат: {**firstElement**} -> {**secondElement**} -> {**thirdElement**}

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Sofka Tripova Stolipinovo Plovdiv  MitkoShtaigata 18 drunk  SashoKompota 0.10 NkqfaBanka | Sofka Tripova -> Stolipinovo -> Plovdiv  MitkoShtaigata -> 18 -> True  SashoKompota -> 0.1 -> NkqfaBanka |
| Ivan Ivanov Tepeto Plovdiv  Mitko 18 not  Sasho 0.10 NGB | Ivan Ivanov -> Tepeto -> Plovdiv  Mitko -> 18 -> False  Sasho -> 0.1 -> NGB |

### Бележки

Може да използвате и надградите решението на предната задача.