A black background with a black square

Description automatically generated with medium confidence **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ

КОМПЮТЪРНО И СОФТУЕРНО ИНЖЕНЕРСТВО

**Програмни Среди:**

Курсова работа:

„Дигитализация на структура на въпрос“

**Реализация**

|  |  |  |
| --- | --- | --- |
| Съставил: | Факултетен номер: | Група: |
| Ангел Любомиров Стойнов | 121222150 | 40 |

Съдържание

[Задача 44-04: 3](#_Toc195392820)

[Задача 44-02: 5](#_Toc195392821)

[Комбинация на двете предложени структури в една обща: 8](#_Toc195392822)

# Задача 44-04:

**Интерфейсни типове в програмирането:**

**Дефиниция и предназначение:** Интерфейсите в програмирането са абстрактни типове, които дефинират набор от методи, свойства, индексатори и събития, които трябва да бъдат имплементирани от класове или структури.

**Значение и ползи от интерфейсите:**

* Поддържат абстракция и полиморфизъм, позволявайки по-лесно поддържане и адаптиране на кода.
* Улесняват създаването на модулни и гъвкави приложения.
* Подобряват преизползваемостта и поддържат по-добра организация на кода.

**Характеристики:**

* Интерфейсите не съдържат реализация на дефинираните членове.
* Реализацията се осигурява от конкретните типове (класове или структури).
* Един тип може да имплементира множество интерфейси, което позволява голяма гъвкавост.

**Интерфейс в .NET**

**Общи характеристики в .NET:**

* Интерфейсите могат да бъдат имплементирани както от референтни (класове), така и от стойностни типове (структури).
* Дефинират начина, който трябва да се спази от имплементиращите го типове.

**Използване на интерфейси със стойностни типове (структури)**

1. **Предимства**

* Позволяват използване на полиморфизъм със структури.
* Увеличават възможността за преизползване на кода (reusability), тъй като методите могат да работят с различни структури, имплементиращи общ интерфейс.
* По-малко натоварване на паметта при стандартно използване, тъй като не изискват garbage collection и се освобождават веднага след излизане извън обсег.

1. **Предизвикателства и недостатъци**

**А) Boxing и Unboxing**

* При използване на структури като интерфейси, стойностните типове трябва да бъдат „опаковани“ (boxing) в референтни типове, което води до допълнителна консумация на памет и процесорно време.
* Това се случва, защото структурите, които обикновено са на стека, се преместват в heap, когато са използвани чрез интерфейс.

**Б) Производителност**

* Преместването между стека и heap води до допълнителни разходи.
* Необходимостта от garbage collection за опакованите обекти може да понижи производителността, особено в системи, изискващи висока ефективност.

**В) Мутационни ефекти**

* Модификации върху опакован стойностен тип чрез интерфейс не засягат оригиналния обект, а само неговото копие.
* Това може да доведе до бъгове и неконсистентност, ако не е правилно разбрано и предвидено от разработчиците.

1. **Добавени от мен подточки за интерфейсни типове в .NET**

* Интерфейсите поддържат множествено наследяване на типове (за разлика от класовете в .NET).
* От C# 8.0 интерфейсите могат да имат default методи с реализации, което подобрява възможностите за еволюция на интерфейсите без нарушаване на съществуващи имплементации.
* Интерфейсите са важен инструмент в Dependency Injection (DI), което подобрява тестируемостта и поддръжката на кода.

# Задача 44-02:

**Интерфейсни типове в програмирането:**

1. **Какво е интерфейсен тип?**

* Интерфейсите са референтни типове данни, които не могат да бъдат инстанцирани директно.
* Те не съдържат реализация на методи, свойства и събития.
* Интерфейсите предоставят средство за постигане на множествено наследяване.

1. **Създаване на интерфейс в C#:**

* Интерфейс се създава чрез ключовата дума *interface*.
* Добра практика е името на интерфейса да започва с "I".
* Членовете на интерфейса са по подразбиране абстрактни и публични.
* Интерфейсите могат да съдържат методи и свойства, но не могат да съдържат полета (променливи), константи, конструктори или деструктори.

1. **Имплементация на интерфейс:**

* В C# интерфейс се имплементира с помощта на символа **:**, последван от името на интерфейса.
* Интерфейсите могат да бъдат имплементирани както от класове, така и от структури.
* Един клас може да имплементира множество интерфейси.
* Не е необходима ключовата дума *override* при внедряване на интерфейс.

1. **Предимства и недостатъци на интерфейсите:**
   1. **Предимства:**

* Позволяват множествено наследяване, което не е възможно при класовете.
* Осигуряват разширяемост на кода, като позволяват добавяне на нова функционалност към съществуващи класове без промяна на оригиналния код.
  1. **Недостатъци:**
* Изискват имплементация на всички членове, което може да доведе до излишен код при класове, имплементиращи множество интерфейси.
* Риск от „Interface Pollution“ – претоварване на интерфейса на класа с несвързани членове от множество интерфейси.

**Основни интерфейсни типове в .NET**

1. **IComparable**

* Интерфейс за сравнение на обекти, дефиниращ метода CompareTo(object).
* Използва се при сортиране на масиви, колекции и операции, изискващи сравнение.
* Имплементиран от типове като System.Char, System.Int32, System.Double, System.String, и System.Enum.

1. **IEnumerable**

* Интерфейс, който дефинира метод GetEnumerator(), връщащ итератор за обхождане.
* Позволява използването на конструкцията foreach.
* Имплементиран от типове като System.Array, System.String, ArrayList, Hashtable, Stack, Queue, и SortedList.

1. **IEnumerator**

* **Интерфейс, който дефинира итератор за обхождане на колекции с:**
  + Свойство Current – връща текущия елемент.
  + Метод MoveNext() – преминава към следващия елемент.
  + Метод Reset() – връща итератора в начално състояние.

**Използване на интерфейси със стойностни типове (структури)**

1. **Предимства:**

* Позволяват полиморфизъм и гъвкавост при структури.
* Намалено натоварване на паметта, защото не изискват garbage collection.

1. **Предизвикателства и недостатъци:**

* **Boxing и Unboxing:** Преобразуването на стойностен тип към интерфейс (референтен тип) изисква "опаковане", водещо до повишени разходи за памет и производителност.
* **Производителност:** Допълнителни разходи при преместване между стека и купчината.
* **Мутационни ефекти:** Промени върху опаковани стойностни типове чрез интерфейс не влияят върху оригиналните обекти, което може да предизвика неочаквано поведение и бъгове.

# Комбинация на двете предложени структури в една обща:

**Интерфейсни типове в програмирането**

1. **Дефиниция и предназначение:**

* Интерфейсите са референтни, абстрактни типове, които дефинират набор от методи, свойства, индексатори и събития.
* Интерфейсите не могат да бъдат инстанцирани директно и не съдържат реализация.
* Реализацията се осигурява от конкретните типове (класове или структури).

1. **Значение и ползи от интерфейсите:**

* Поддържат абстракция и полиморфизъм, позволявайки лесна поддръжка и адаптация на кода.
* Улесняват създаването на модулни и гъвкави приложения.
* Подобряват преизползваемостта и организацията на кода.
* Позволяват множествено наследяване на типове, което е невъзможно при класовете.
* Позволяват разширяемост на кода без промяна на съществуващия изходен код.

1. **Характеристики:**

* Интерфейсите се дефинират чрез ключовата дума interface в C#.
* Добра практика е името на интерфейса да започва с буквата "I".
* Членовете на интерфейса по подразбиране са публични и абстрактни.
* Интерфейсите могат да съдържат методи и свойства, но не и полета, константи, конструктори или деструктори.
* От C# 8.0 интерфейсите могат да имат default методи с реализации, което помага за тяхната еволюция.
* Един тип може да имплементира множество интерфейси, което осигурява гъвкавост.

**Имплементация на интерфейс в C#**

Интерфейс се имплементира чрез символа :, последван от името на интерфейса. Интерфейсите могат да бъдат имплементирани както от класове, така и от структури. Не е необходима ключовата дума override при имплементация на интерфейс.

1. **Предимства**

* Поддържат множествено наследяване.
* Осигуряват гъвкавост и възможност за разширяване без промяна на оригиналния код.
* Подобряват тестируемостта и поддръжката на кода (Dependency Injection).

1. **Недостатъци**

* Необходимост от имплементация на всички членове, което може да доведе до излишен код.
* Риск от „Interface Pollution“ – претоварване на интерфейса на класа с несвързани членове.

**Основни интерфейсни типове в .NET**

1. **IComparable**

* Дефинира метода CompareTo(object) за сравнение.
* Използва се за сортиране и операции, изискващи сравнение.
* Имплементиран от типове като System.Char, System.Int32, System.Double, System.String, и System.Enum.

1. **IEnumerable**

* **Дефинира метода GetEnumerator(), връщащ итератор за обхождане на колекции.**
* Поддържа конструкцията foreach.
* Имплементиран от System.Array, System.String, ArrayList, Hashtable, Stack, Queue и SortedList.

1. **IEnumerator**

* **Дефинира итератор за обхождане с методи и свойства:**
  + Current – текущ елемент.
  + MoveNext() – преминава към следващия елемент.
  + Reset() – връща итератора в начално състояние.

**Използване на интерфейси със стойностни типове (структури):**

1. **Предимства:**

* Позволяват полиморфизъм със структури.
* Увеличават повторната употреба на кода.
* Намалено натоварване на паметта поради липса на garbage collection.

1. **Предизвикателства и недостатъци:**

**А. Boxing и Unboxing:**

* Преобразуването на структури към интерфейсен тип изисква „опаковане“, увеличаващо разходите за памет и производителност.

**Б. Производителност:**

* Преместване на данни между стека и heap намалява ефективността.

**В. Мутационни ефекти:**

* Модификации върху опакованите стойностни типове не засягат оригиналния обект, което може да доведе до неконсистентност и бъгове.