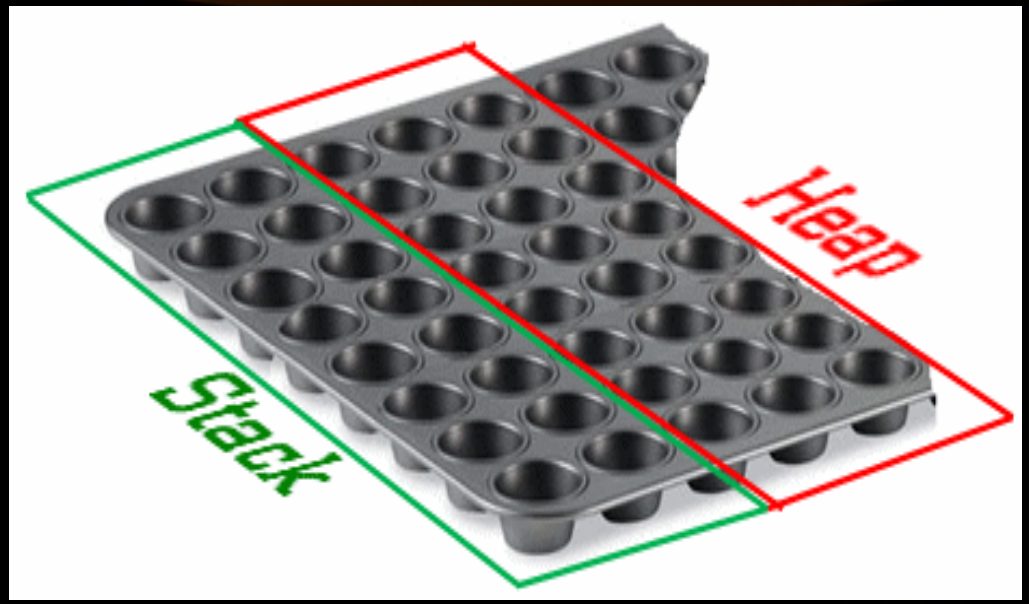
Памет, стек и хийп, разположение на обектите в паметта

Учителски екип

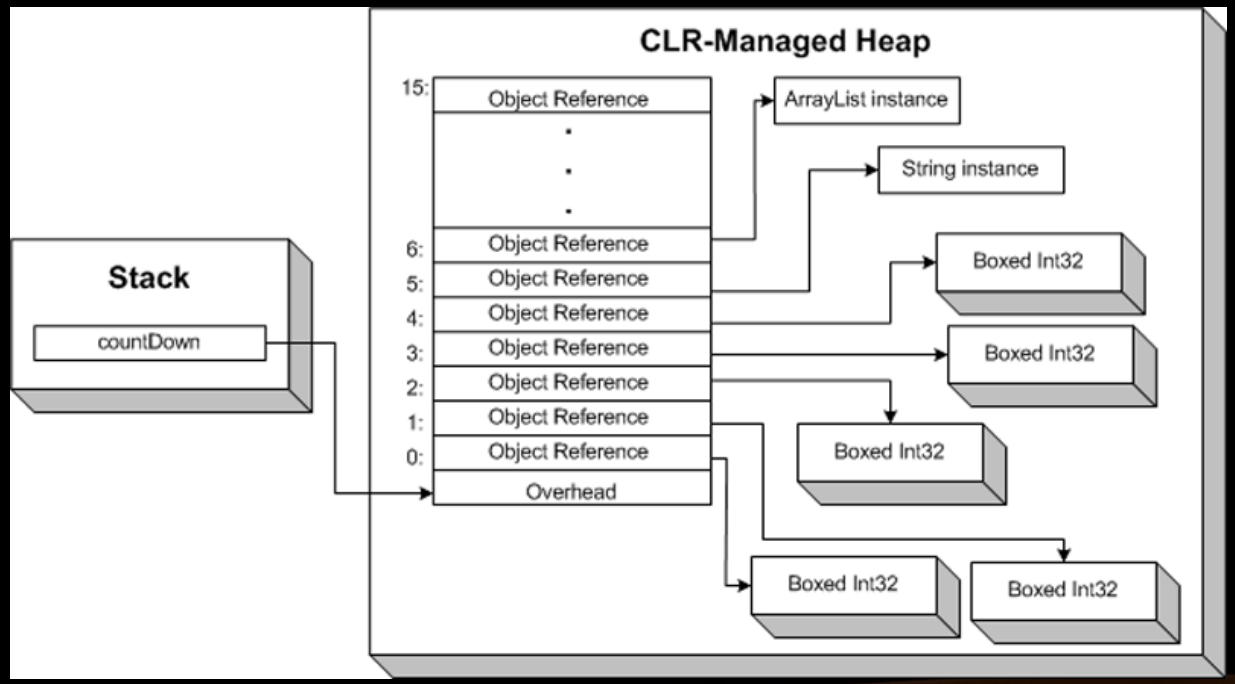
Обучение за ИТ кариера

https://it-kariera.mon.bg/e-learning/



Съдържание

1. Памет
2. Стек
3. Хийп
4. Разположение на обектите в паметта



Управление на паметта в .NET Framework

- Управлението на паметта в .NET е **автоматично**
- Вече **не е необходимо** да се пише специален код, който да освобождава заетата от обектите памет
- При създаване на нов обект се заделя памет в регион, наречен **managed heap**
- Когато обектът стане ненужен, той просто се **“изоставя”,** и в по-късен етап се почиства автоматично от т.нар. **garbage collector** – системата за почистване на паметта.

Управление на паметта в .NET Framework

- За някои обекти не е достатъчно само да се освободи паметта
- Garbage collector-а се грижи **само за паметта** и не знае какви други системни ресурси използва обектът
- **Финализатори** (**finalizers**) – специални методи, които се изпълняват преди обектът да се унищожи

Предимства на автоматичното управление на паметта

- **Освободени сме от грижата ръчно** да почистваме ненужните обекти
- Предотвратяването на т.нар. **“memory leaks”** или **изтичане на памет**.
- **Броене на референциите към обектите**, както и частния случай с **циклични референции не съществува**

Недостатъци на автоматичното управление на паметта

- Почистването й е **тежка** и **времеотнемаща операция**
- **Няма гаранция кога** се изпълнява garbage collector и колко време отнема!

Как се заделя памет в .NET? Стек и Хийп

- Когато CLR се инициализира, той заделя регион от последователни адреси в паметта. Това е т.нар. **динамична памет** или **managed heap**.
- За разлика от стойностните типове, чиято памет се заделя в **стека** и се освобождава веднага, след като променливата излезе от обхват, паметта, нужна **за референтните типове,** винаги се заделя в managed heap.

Как се заделя памет в .NET?

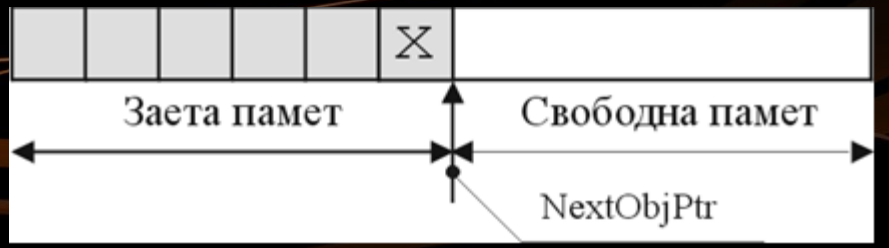
- За да създадем обект в хийп, използваме код, подобен на този:

```
SomeObject x = new SomeObject();
```

- C# компилаторът превежда кода в IL **newobj** инструкция:

```
newobj instance void
MyNamespace.SomeObject::.ctor()
```

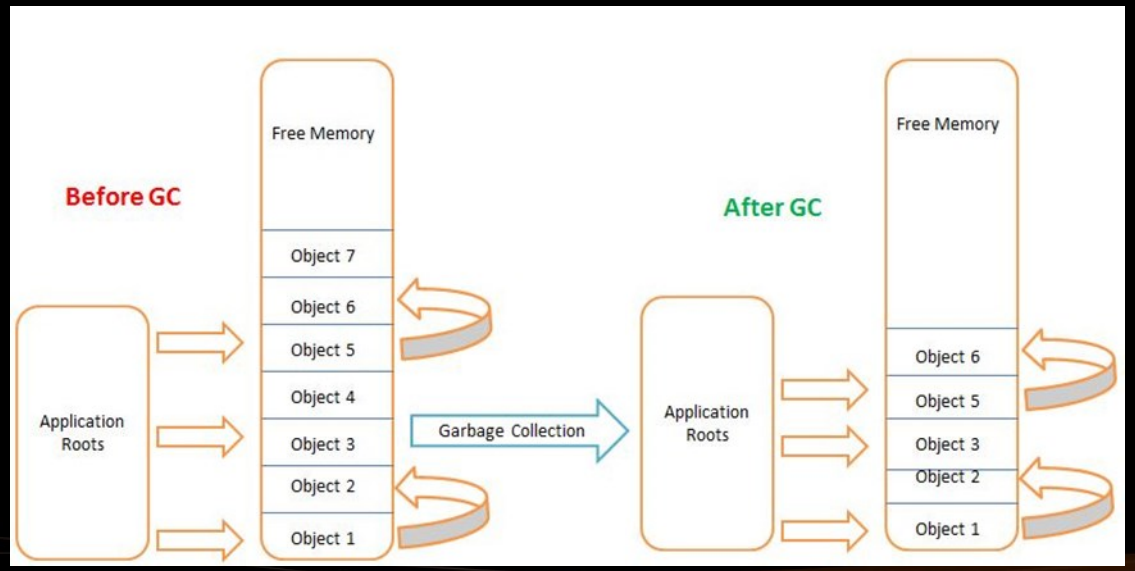
Как се заделя памет в .NET?(2)



- CLR изчислява необходимата памет, прибавя размера на **MethodTablePointer**, **SyncBlockIndex** и **NextObjPtr**.
  1. Ако в хийпа има памет - се заделя, извиква се конструктора и адресът на обекта се връща от оператора **new**.
  2. Ако няма памет се стартира garbage collector и после пак се опитва да създаде обекта
  3. Ако и тогава няма достатъчно памет, хийпът се увеличава
  4. Ако това е невъзможно, **new** операторът предизвиква **OutOfMemoryException**.

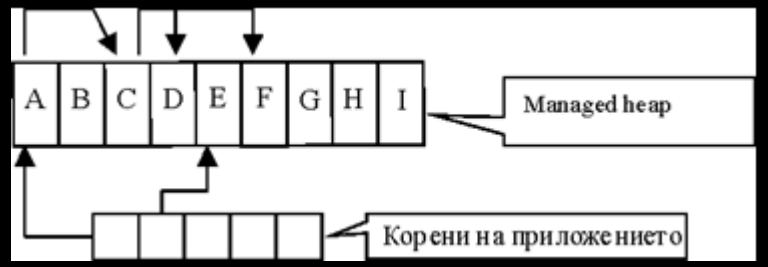
Как работи Garbage Collector?

- При достатъчно свободна памет това става с преместването на един указател. Ако в хийпа няма достатъчно място:
- Нишките трябва да се **приспят**
- **Освобождават се** неизползваните обекти



Алгоритъмът за почистване на паметта

- **Корените на приложението** са точката, от която системата за почистване на паметта започва своята работа
- Ако глобална променлива сочи към обект A от хийпа, то A се добавя към графа. Ако A съдържа указател към C, а той от своя страна към обектите D и F, всички те също стават част от графа. Така garbage collector обхожда рекурсивно в дълбочина всички обекти, достъпни от глобалната променлива A:
- Обектите, нуждаещи се от финализация **не се унищожават веднага**. Те остават и указатели към тях се добавят в опашката **Freachable**

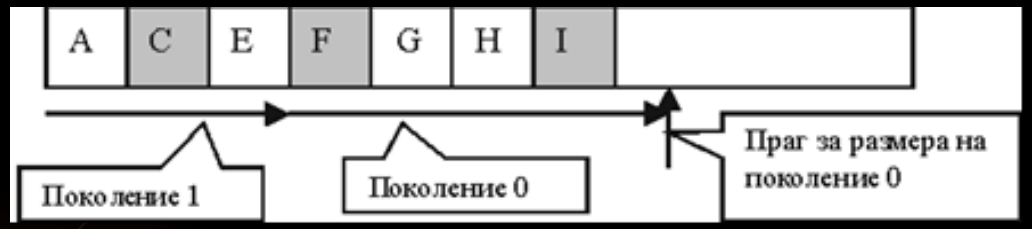


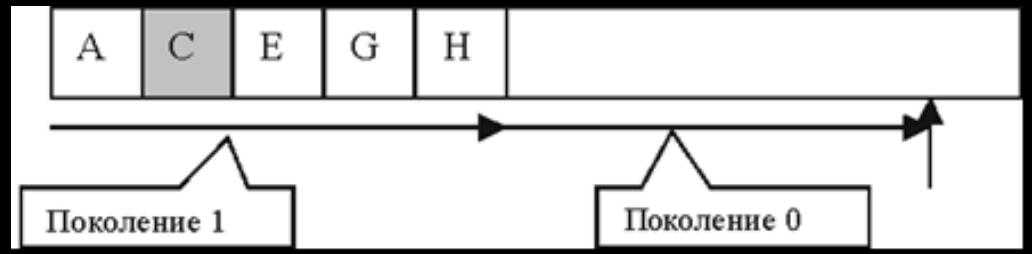
Поколения памет

- Почистването на част от динамичната памет винаги е по-бързо от почистването на цялата памет.
- Garbage collector ги разделя на **поколения** - колкото по-нов е един обект, толкова по-вероятно е животът му да е кратък. колкото по-стар е обектът, толкова по-големи са очакванията той да живее дълго.
- Обектите, създадени по едно и също време обикновено имат връзка помежду си и имат приблизително еднаква продължителност на живота.

Почистване на Поколение 1 и Поколение 2

- След първото обхождане свободни са **C,F** и **I**. Но само F **и** I са от поколение 0 и се почистват.
- C ще се почисти, когато е свободна и няма обекти от по-ниско поколение



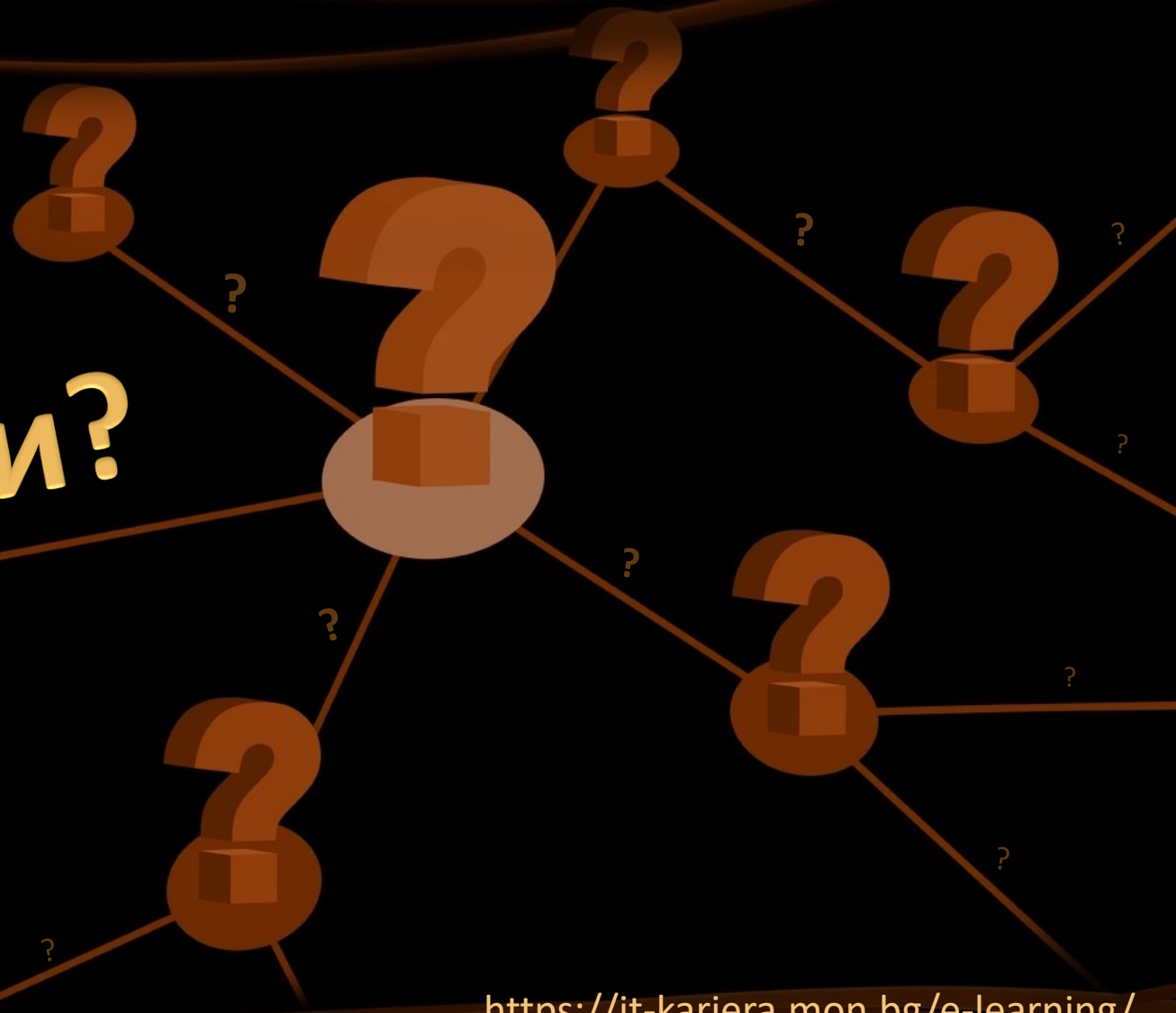


Блок памет за големи обекти

- всички големи обекти (с размер над 20 000 байта) се разполагат в отделен хийп. Разликата между него и стандартния managed heap е това, че хийпът за големи обекти **не се дефрагментира** и пести много процесорно време.
- Всичко това става прозрачно за разработчиците, сякаш има един единствен хийп.
- Големите обекти винаги се считат за част от Поколение 2. Трябва да създаваме по-малко на брой големи обекти и да ги използваме в случаите, когато те ще живеят дълго време.

Памет, стек и хийп, разположение на обектите в паметта

Въпроси?



https://it-kariera.mon.bg/e-learning/

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**