**37 Модел на явна финализация в .NET среда.Интегриране на Finalize() и Dispose()**

Когато се създава нов обект, CLR проверява дали типът дефинира

**Finalize()** метод и ако това е така, след създаването на обекта в дина-

мичната памет (но преди извикването на неговия конструктор), указател

към обекта се добавя към Finalization list. Така Finalization list съдържа

указатели към всички обекти в хийпа, които трябва да бъдат финализи-

рани (имат **Finalize()** методи), но все още се използват от приложението

(или вече не се използват, но още не са проверени от garbage collector).

**Създаването на обект, поддържащ финализация изисква**

**една допълнителна операция от страна на CLR – поста-**

**вянето на указател във Finalization list и следователно**

**отнема и малко повече време.**

Взаимодействието на garbage collector с обектите, нуждаещи се от фина-

лизация, е твърде интересно. Нека разгледаме следния пример.

Фигурата по-долу показва опростена схема на състоянието на динамич-

ната памет точно преди да започне почистване на паметта. Виждаме че

хийпът съдържа три обекта – A, B и C. Нека всички те са от Поколение 0.

Обект A все още се използва от приложението, така че той ще оцелее при

преминаването на garbage collector. Обекти B и C, обаче, са недостъпни от

корените и се определят от garbage collector-a като отпадъци.

И така, garbage collector първо определя обект B като недостъпен и

следователно – подлежащ на почистване. След това указателят към обект

B се изтрива от Finalization list и се добавя към опашката Freachable. В

този момент обектът се **съживява**, т.е. той се добавя към графа на

достъпните обекти и вече не се счита за отпадък. Garbage collector

пренарежда динамичната памет. При това обект B се третира както всеки

друг достъпен от приложението обект, в нашия пример – обект A.

След това CLR стартира специална нишка с висок приоритет, която за

всеки запис във Freachable queue изпълнява **Finalize()** метода на съот-

ветния обект и след това **изтрива записа от опашката**.

При следващото почистване на Поколение 1 от garbage collector, обект B

ще бъде третиран като недостъпен (защото записът вече е изтрит от Freachable queue и никой от корените на приложението не сочи към

обекта) и паметта, заемана от него ще бъде освободена. Забележете, че

тъй като обектът вече е в по-високо поколение, преди това да се случи е

възможно да минат още няколко преминавания на garbage collector,

Интерфейсът **IDisposable** се препоръчва от Microsoft в тези случаи, в

които искате да гарантирате моментално освобождаване на ресурсите

(вече знаете, че използването на **Finalize()** не го гарантира).

Използването на **IDisposable** се състои в имплементирането на интер-

фейса от класа, който обвива някакъв неуправляван ресурс и освобожда-

ването на ресурса при извикване на метода **Dispose()**.n