C#   .NET

Generational Garbage collector:

The CLR's Garbage collector (GC) is a generational garbage collector, also known as ephermal garbage collector.

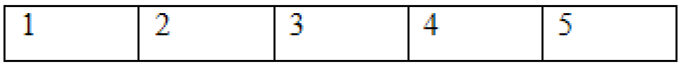
It has three generations:

Generation 0 :

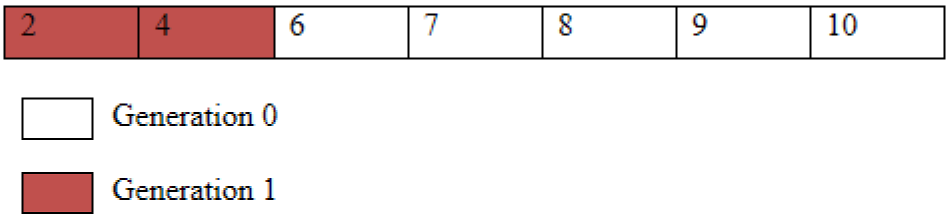
It contains all newly constructed object which are never examined by GC.

Generation 1:

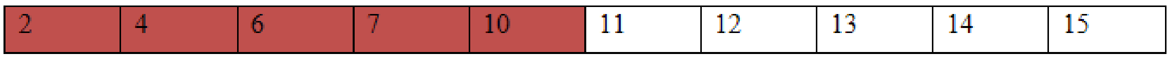
The CLR, when initializes, selects a budget size in kb for generation 0 . If the creation of an object causes the generation 0 to surpass its budget, garbage collection is started. The objects which are not collected in Generation 0 are moved to Generation 1 and Generation 0 is emptied. Let's say the budget of Generation 0 is equal to size of 5 objects. So generation 0 would look like below before creation of object 6:



After creation of object 6, garbage allocation gets started which deallocates the garbage objects 1, 3 and 5 and moves 2 and 4 adjacent to each other in Generation 1.



The budget size of generation 1 is also selected by CLR upon initialization. Creation of object 11 causes the GC to start again which may move some more objects to generation 1.

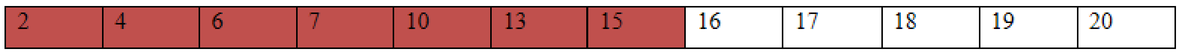


Generation 1 is ignored for Garbage Collection until it reaches its budget size for Garbage collection, which improves the performance of GC.

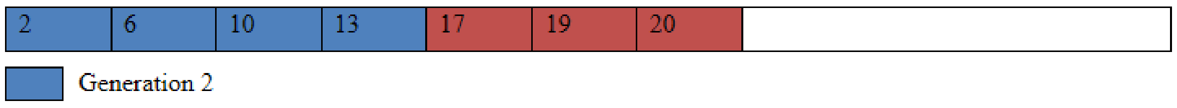
Generation 2:

Over the several generation 0 collection, generation 1 may surpass it's budget limit which cause GC to collect the Garbage from both generations. In this case, generation 1 survivors are promoted to generation 2, generation 0 survivors are promoted to generation 1, and generation 0 is empty.

Let's say allocation object 21 cause Garbage collection and generation 1 budget have been
reached.



So heap would look like below with the object that survived in Generation 1 promoted to generation 2.



So basically Generation GC assumes that newer objects have more probability to collected.

We know that CLR selects budgets for all three generations but it can modify them as GC is a self-tuning collector. If GC sees that there are very few surviving objects after collecting generation 0, it might decide to reduce the budget of generation 0, so that lesser work is done. On the other hand, if GC collects generation 0 and sees that there are a lot of surviving objects, not a lot of memory was reclaimed in the garbage collection. In this case, the garbage collector will grow generation 0's budget. The GC also modifies the budgets of generation 1 and generation 2 accordingly.