Garbage collection

* Basic of working of garbage collection
  + How the Memory is managed by garbage collector
  + Problems faced earlier
    - Forgotten to free memory not needed
    - Attempted to free memory which is already freed
* Managed heap
  + Reserved address space
  + nextObjPtr
    - Calculation of how memory sufficiency is checked
  + Procedure of allocation of resources from managed heap
  + Comparison of C heap with Managed heap
    - We never delete memory from managed heap like in C heap
    - Memory allocation in contiguous which gives locality of reference
* Garbage collection algorithm
  + Optimizing engine determines the apt time to do GC
  + Determines which objects are no longer used
  + GC has access to list of roots maintained by JIT and runtime
  + Unreachable roots are freed
* Finalize method
  + Finalize method call causes
    - Generation 0 is full
    - Code explicitly calls System.GC’s static Collect method
    - Windows is reporting low memory conditions
    - The CLR is unloading an AppDomain
    - The CLR is shutting down
  + How it works
  + Internals
    - Finalization list
    - Freachable queue
  + Dispose pattern
    - Forcing an object to clean up
  + Ressurection
    - GC places a reference to the object on the freachable queue, the object is now reachable from a root and has come back to life
* Monitoring and controlling of objects manually
  + GC Handle table
  + GC modes and Flags used in that
* Generations
  + Garbage collection id called when Generation 0 gets full
  + Utmost there can be 3 generations
  + Starts form Generation 0
  + The newer an object is, the shorter its lifetime will be .
  + The older an object is, the longer its lifetime will be .
  + Collecting a portion of the heap is faster than collecting the whole heap .
* Thread hijacking
  + Multiple threads
  + Safe point
    - Thread can be suspended until GC completes.