Particles, vertices and trees

- Particle base class (Part)
  - Should contain only intuitive information for a particle:
    Momentum, particle type, point on the track (or x,y at fixed z)
    Charge? (could be different type for Sim and Rec)
- Derived classes
  - GenPart, SimPart, RecPart
  - Contain additional information, specific to this class
    error matrix on position and momentum @ the point , key to ProtoPart…
  - Disadvantages:
    - users have to deal with 4 classes
    - need for methods to create Part containers from XxxPart if needed
  - Advantages:
    - one always knows which type an object is
    - no dynamic casting (ugly!)
- Vertex base class (Vertex)
  - Also only minimum information for a vertex
    Position, Type of vertex (decay, interaction, kink…)
- Derived classes
  - GenVrtx, SimVrtx, RecVrtx (if the 3 are needed!)
  - Additional information:|
    Error covariance matrix, Chi2, ndf, Proper time of decay?
- Containers
  - XxxPart should always be in a container (or in two?) such that it can be easily deleted
  - 2 types of containers:
    - Simple list of XxxPart (e.g. ListOfPions)
    - Tree (from simple as Ks-> pi pi to more complex as
      Bs-> KDs->Kpi or B-> J/ΨΦ ->μμ KK
  - Advantages of a tree over contained relationship:
    - When passing a particle, one is sure the method uses only the particle class, not any further information which could be derived from it
    - A particle exists without a vertex, and most tools do not need to know there is one
    - Specific methods for the tree: navigator, consistency checker (e.g. check there is no double ref to the same ProtoPart)
  - Always deal with *copies* of particles (at least conceptually, possibly use SmartPtr in cases where no real copy is needed)
- Use case for building a decay tree (J/ΨΦ), starting from ProtoPart
  1. Build a list of μ's
  2. Build a list of K's
  3. Build a list of J/Ψ's and a tree for each of these (the μ's are copied). The tree includes the μμ vertex as fitted from these two particles
     This is a use case for an object to belong to 2 containers…
  4. id for Φ
  5. Possibly refit the μ's and K's with mass constraint (no pb since they are copies)

6. Look for B’s from the lists of J/Ψ’s and of F□□□□□□□□□□□□□□□□□□□□□□□
7. Build the B particle(s) and its tree, fit vertex with μ’s and K’s. Question: should one put in the tree a reference to that vertex for the J/Ψ and the Φ?
8. Check for consistency of the solution(s)