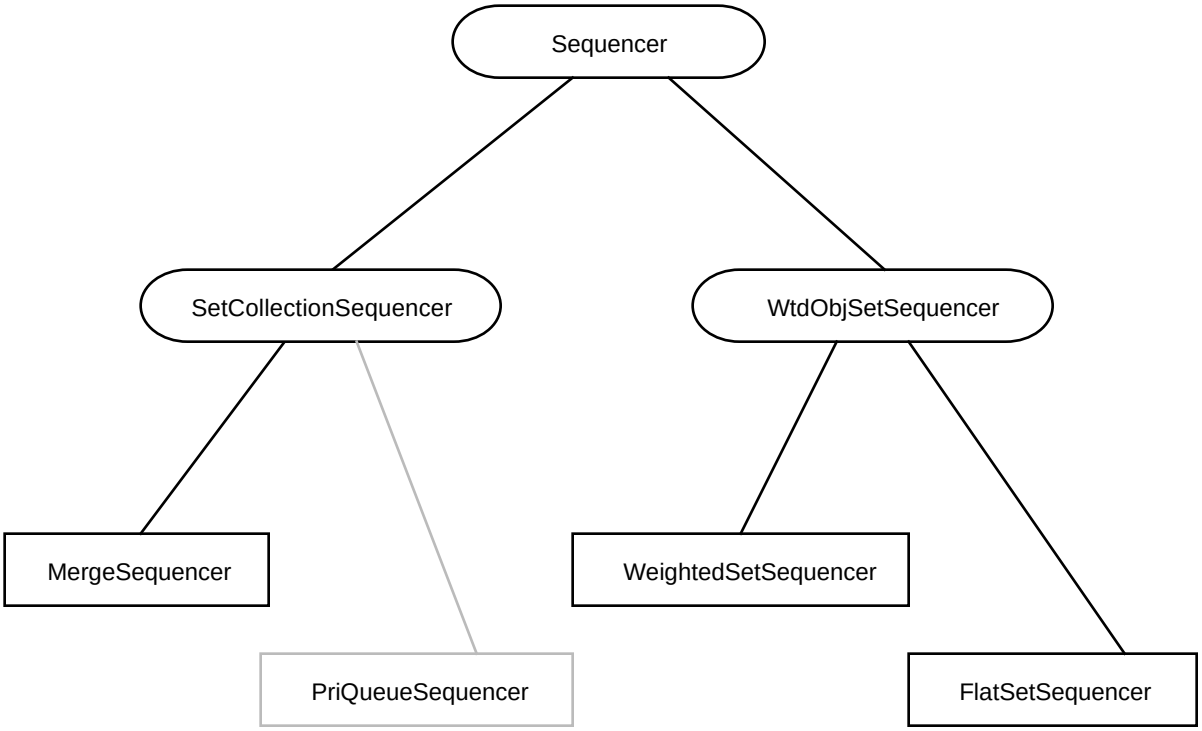
Sequencers

These are the Sequencers defined in MARIAN v.2.1. Remember that ovals denote Java interfaces and rectangles Java classes.



A total of six sequencers are possible in the Academy, following a 2x3 decision table. The first (binary) decision is whether component sets are added to the Sequencer individually, following the addSet() method defined in the SetCollectionSequencer interface, or whether sets are added in the form of a WtdObjSet of key values together with a mapping function, following addSets() in the WtdObjSetSequencer interface.

The ternary decision governs the data structures and algorithms used within the Sequencer. The motivating question is how many and what sort of sets are to be merged. One possibility is that all the sets to be merged are *flat:* that is, all the weights within each component set are the same. In that case the component sets can be ordered once based on their overall weights and explored in that order.

On the other hand, if the component sets are proper weighted object sets (that is, if the weights of their component objects vary), then they must be constantly shuffled as they are merged. How to do this efficiently depends on whether there are few component sets or many.

If there are only a few sets, we are best off maintaining them in some relatively unsophisticated data structure and using a linear search to determine the next element in the sequence. The cost of finding the next elements among k component sets is then $k \bullet c_k$ where c_k is relatively small. As k grows larger, it becomes more efficient to use more aggressive data structures, where the cost of finding the next element in the sequence is **O**(log(k)), or precisely $\log(k) \bullet c_{\log(k)}$. Since $c_{\log(k)}$ is larger than c_k, this complexity is only required when k is greater than some threshold value. As a rule of thumb, this turning point is usually around 20; our searchers usually either deal with values much less than this or much greater.

The decision table thus looks like this:

	Sets added individually	**Sets added as WtdObjSet + Mapping**
Flat sets (any number)	*	FlatSetSequencer (used in UnwtdLinkClassManager)
Few proper sets $k \bullet c_k < \log(k) \bullet c_{\log(k)}$	MergeSequencer (used in SuperclassManagers, TextManager and StructuredDocumentManager)	?
Many proper sets $\log(k) \bullet c_{\log(k)} < k \bullet c_k$	*	WeightedSetSequencer (used in WtdLinkClassManager)

The colored cells are the Sequencers being implementing in v. 2.1 MARIAN. The cells labeled "*" are not required except as conveniences, since we can always take a collection of individual sets and create a "key" WtdObjSet to and a trivial mapping to recover them. However, if the code for them can be produced easily while building the corresponding WtdObjSetSequencers, they should be. In particular, the Sequencer in the lower left corner has already been defined as the "empty" PriQueueSequencer class.

The Sequencer labeled "?" can also be passed over in v.2.1, but it should be designed for. More sophisticated versions of the TextManager, projected for v. 2.2, will be clearer and simpler to code if we have it available at that time.