1. Questions?

   (a) Many languages provide support for associative arrays. `Map<K,V>` implementations are associative arrays for Java.
   (b) A key is used to find (or map to) a value.
   (c) Supports `V get(K)`, `V put(K,V)`, `V remove(K)`, and `size` and friends.
   (d) Also: `containsKey(K)` and `containsValue(V)`.
   (e) Also: `keySet`, `values`, and `entrySet`.

3. MapList
   (a) Implement the mappings in a `List`.
   (b) Each element of the list is an `Association<K,V>`.
   (c) Worries?
      i. Slow. Every operation is linear.
      ii. The `List` doesn’t facilitate keeping mappings with unique keys.

4. Tables implement an `OrderedMap`.
   (a) Keys must be comparable.
   (b) Values stored in an `OrderedStructure` (a `SplayTree`) of `ComparableAssociations`.
   (c) Expected logarithmic performance.

5. Friday: Hashtables.
   (a) An unordered collection of key-value pairs (`Association<K,V>`).
   (b) Goal of the structure is to attempt constant time access. Possible, but hard.
   (c) Notion of hashing into a vector or array.
      i. Every `Object` has a method `o.hashCode()` that generates a reproducible integer that is not obviously correlated to the object’s value.
      ii. The `hashCode` of two `equals` values are (ideally!) equal.
   (d) Notion of collision.
   (e) Two techniques: open addressing and external chaining.
   (f) Open addressing resolves collisions by rehashing, but suffers from occasional expansion of the table.
      i. Clustering.
      A. Primary clustering: when items seek the same first slot, and then continue to conflict during resolution.
      B. Secondary clustering: when items that wouldn’t originally collide end up colliding during resolution.
      C. Both can be avoided, but it takes effort on a case-by-case basis.
      ii. Linear probing, rehashing, and double hashing.
      A. Linear probing: find next free slot. (Best supported by Java.)
      B. Rehashing: use a second hash function.
      C. Double hashing: use a second hash function to generate linear probing offset.
      iii. Reserved entries: values interact in the table....
   (g) External chaining resolves collisions by keeping values in an external list (or other container). See `ChainedHashtable`.