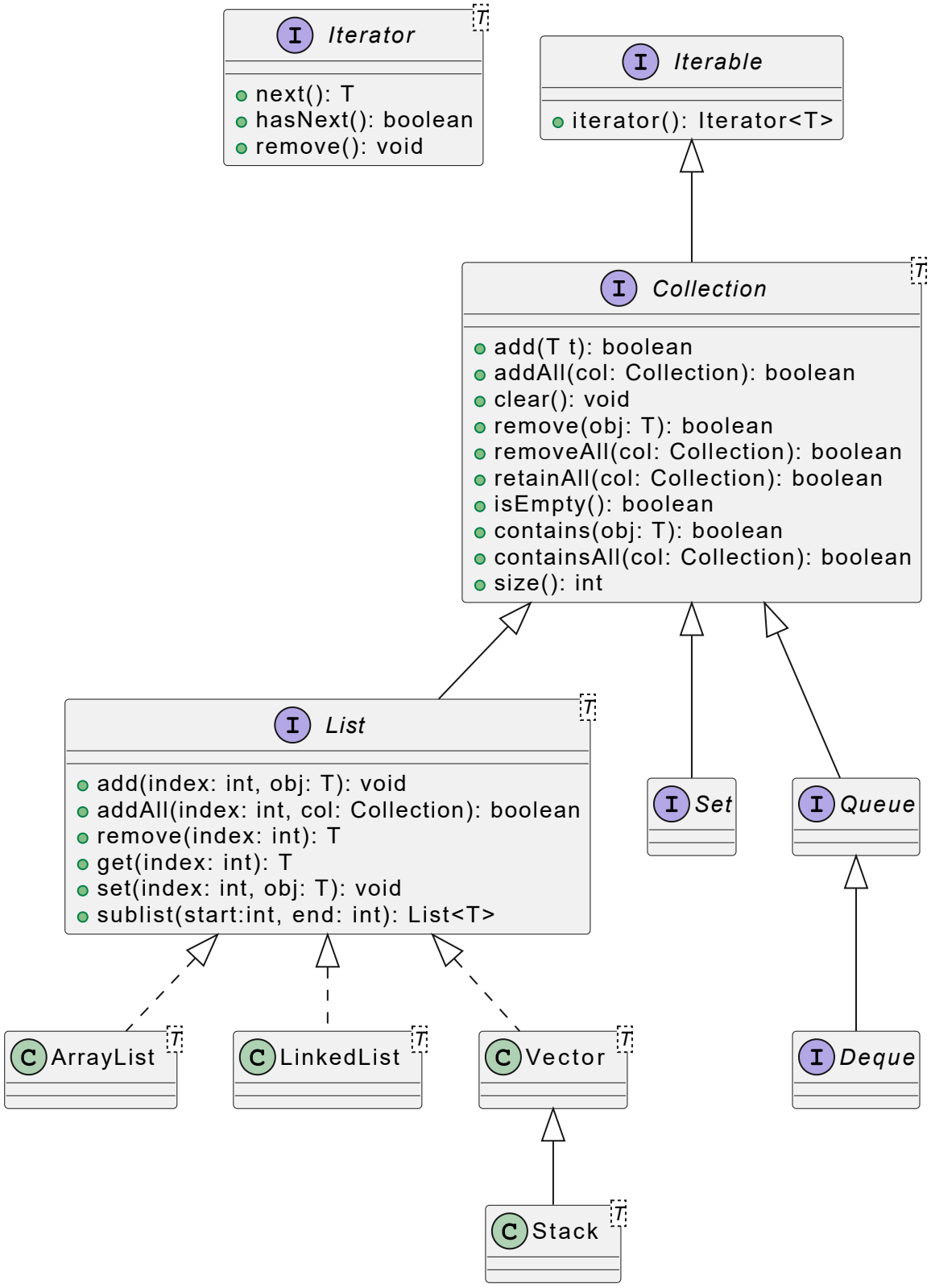
Collection framework in Java

Limitations of using an array and the need of a new type called Collection

- Arrays are created in the heap, the size of which can be determined during the runtime (unlike C language)
- Once array is created of a particular size, you cannot increase or decrease the size
- An array occupies bytes continuously in the RAM, and sometimes, if the array size is huge, we may not get enough memory
- insertion and deletion operations on an array element/s is time consuming (arrays do not provide any built in mechanisms)

Collections in Java

- Introduced in JDK 1.0
- Lots of things changed in version 1.2
- Java 5 brought in the feature of generics (homogeneous collection)
- Java 8 brought in the feature of streams (a stream of data)
- Java collection framework is a bunch of interfaces and many implementing classes
    - choose an implementation based on your requirement and optimization



List implementations

1. ArrayList
   - introduced in version 1.2
   - internal mechanism is an array
   - default size of the internal array is 10
   - every time we add an element, the capacity is ensured
   - insertion and deletion operations are time consuming
   - since arrays are faster (random access), this should be the default choice for `List` type
2. LinkedList
   - uses a linked list to store data

- elements are not contiguous
- each element has a reference to the next and previous element
- insertion and deletion operations are faster
- if there is a lot of performance loss due to heavy insertion and deletion operations, then choose this class for an object of `List` type

3. Vector
    - internal mechanism is an array, just like an ArrayList
    - was introduced in version 1.0 (hence a.k.a. legacy collection)
    - went through lot changes in version 1.2 (and in version 1.5) and was aligned with the new collection framework
    - most of the methods are synchronized (data safety is ensured in a multithreaded application)
        - additional overhead involved
    - use this class only if your data in the collection is shared to multiple threads
4. Stack