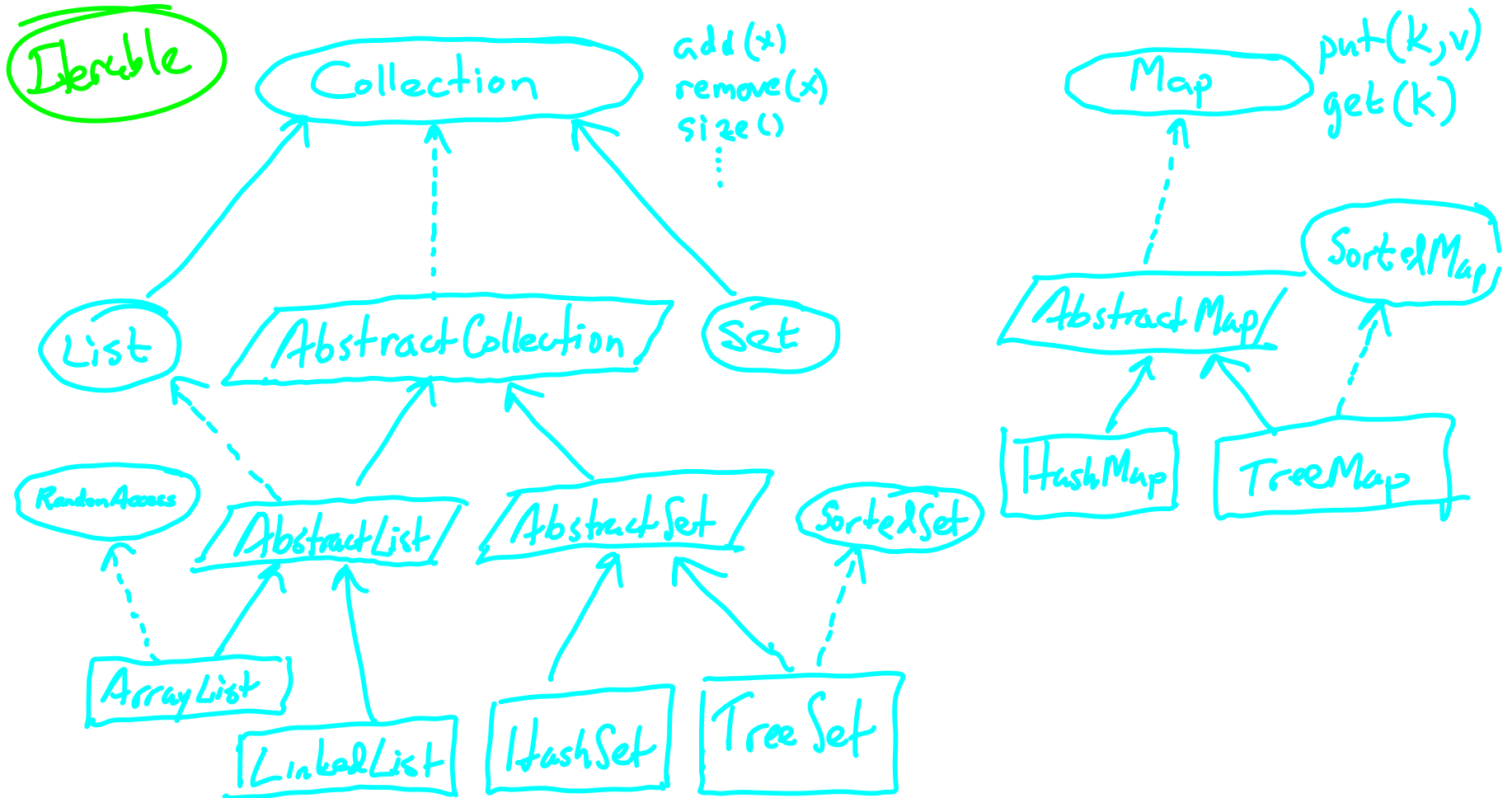


Java Collections as a Case Study in Design

Note Title

1/25/2007



- ① Abstract class — • useful type (polymorphically)
• common implementation (incomplete — leaves things open)
- ② Interfaces — useful type!

Cross cutting — across hierarchy
indicates capability or property

Iterable
Comparable
Cloneable
Runnable

Sorted Set
Sorted Map

- ③ Plug-in Functionality — ex. Comparator

`new TreeSet<T> (Comparator<? super T>)`
`t = new TreeSet<T> (comp);`
parameter is a function

Utilities:

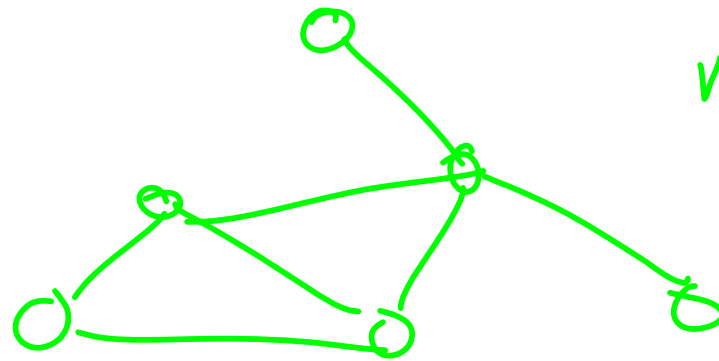
`Collections.sort (List<T> l)`
`.sort (List<T> l, Comparator<? super T> c)`
`.shuffle (List<T> l)`

simplifies API to separate out utilities
(static methods)

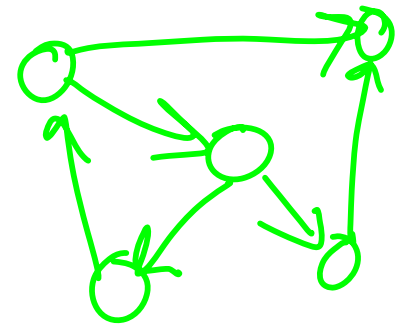
Graph ADT

$$G = (V, E)$$

↑ vertices
edges



Undirected
cities / travel
flow - pipeline
network



Directed
graph

cycle?
shortest path