Java Collections as a Case Study in Design
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>)

\[ t = \text{new TreeSet<T> (comp)} \]

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 graph
- Cities/travel
- Flow - pipeline
- Network

- Directed graph
- Cycle?
- Shortest path