

Data Structures

Collections Framework



Collection Framework

- Framework of built-in **data structures**
- Provides **consistent** interaction with all collections
- Provides **efficient** implementations
- Provides **common algorithms** (e.g. search, sort)
- Size is **flexible**, collections may grow/shrink in size

<http://docs.oracle.com/javase/8/docs/technotes/guides/collections/index.html>



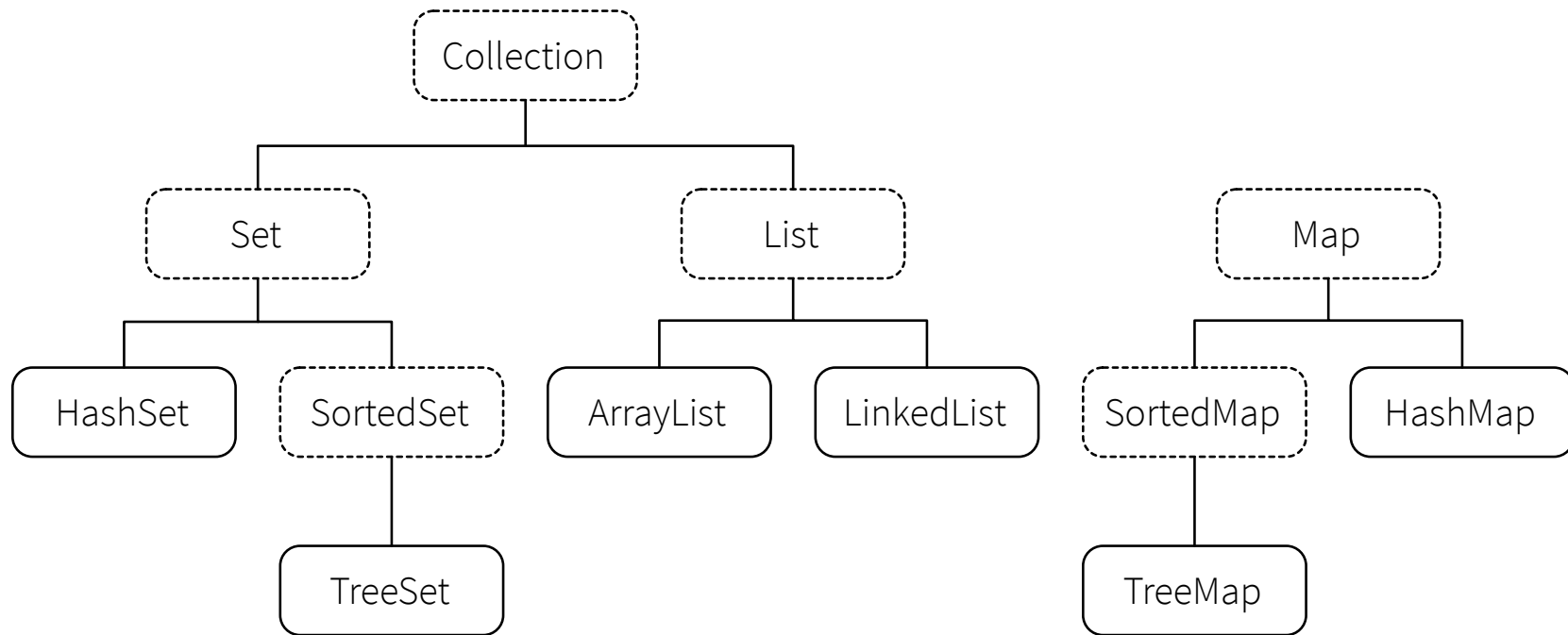
Collection Framework

- A collection must contain elements of the **same type***
- Requires **objects**, does not work with primitive types
 - Use Integer instead of int, etc.
 - Collections are objects, allows **nesting**
- Specify element type using **Generics** syntax
 - e.g. `HashSet<String>`
 - e.g. `HashMap<Integer, String>`

<http://docs.oracle.com/javase/tutorial/extra/generics/>



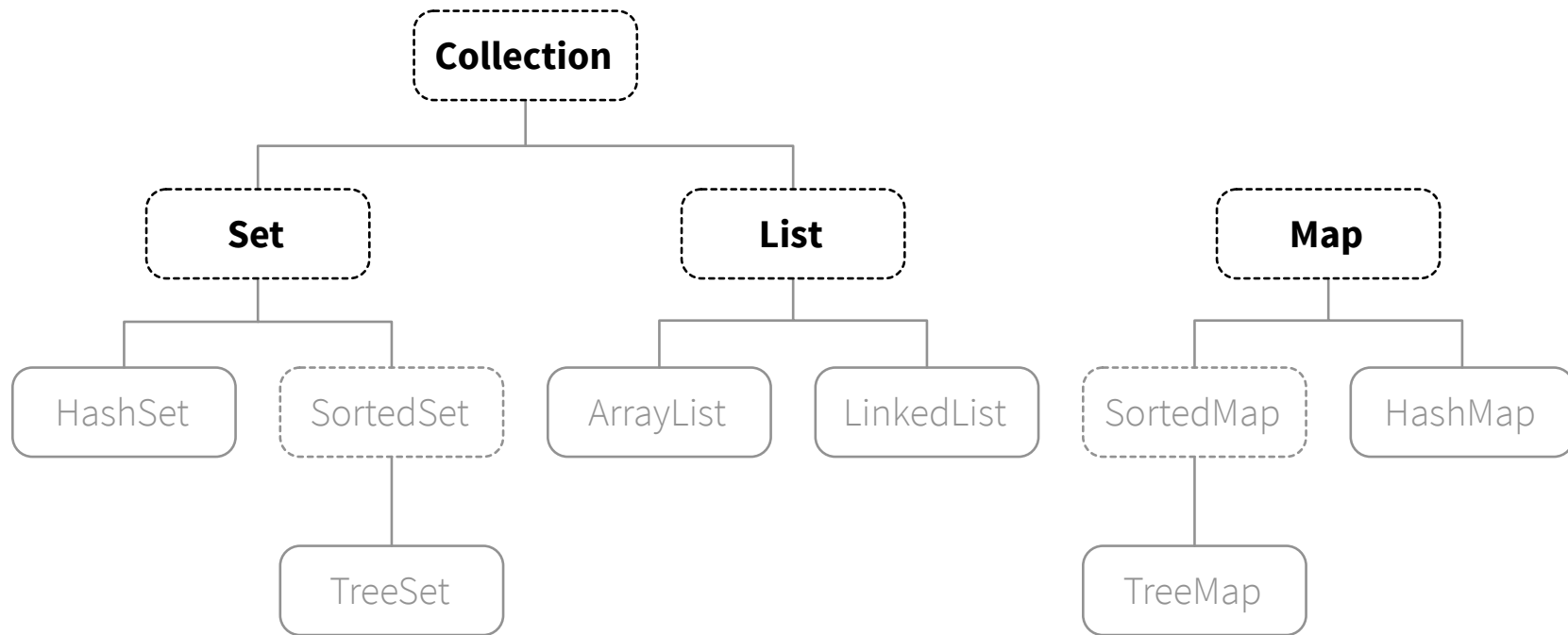
Collection Framework*



*Abbreviated Framework



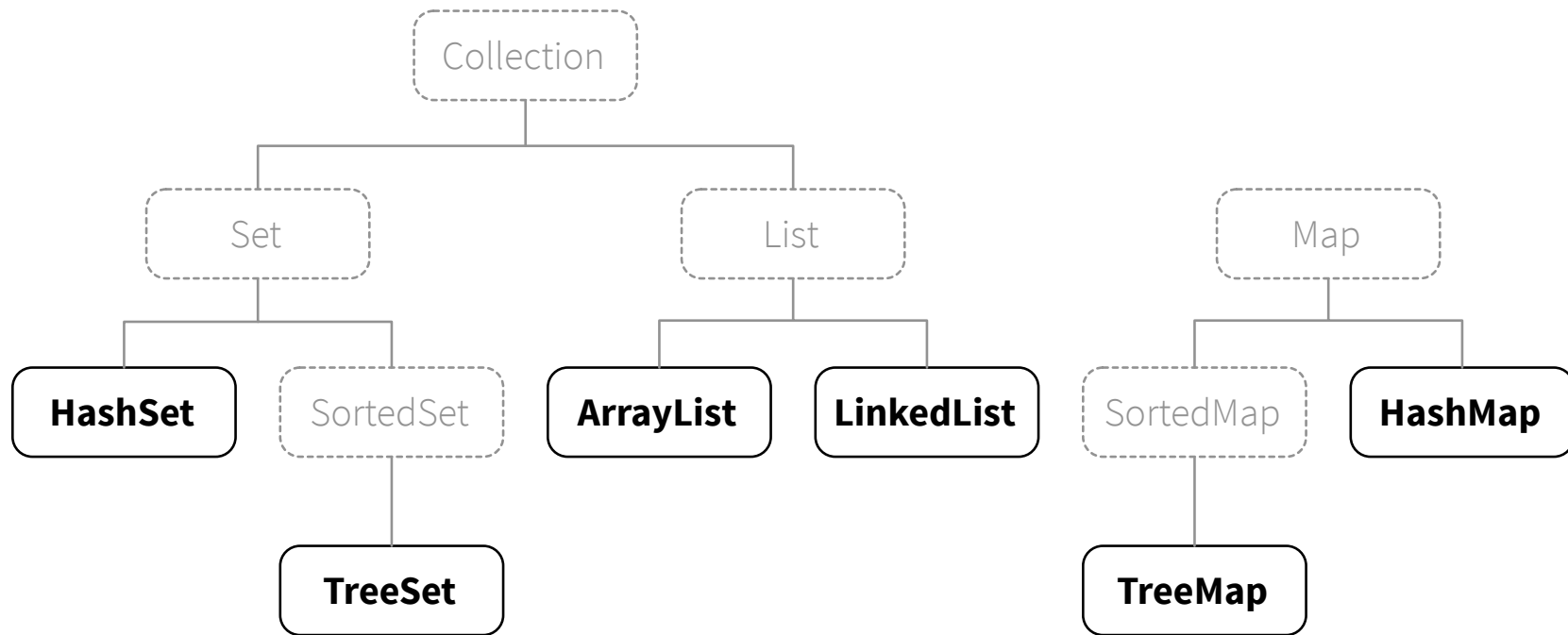
Collection Framework*



*Core Interfaces



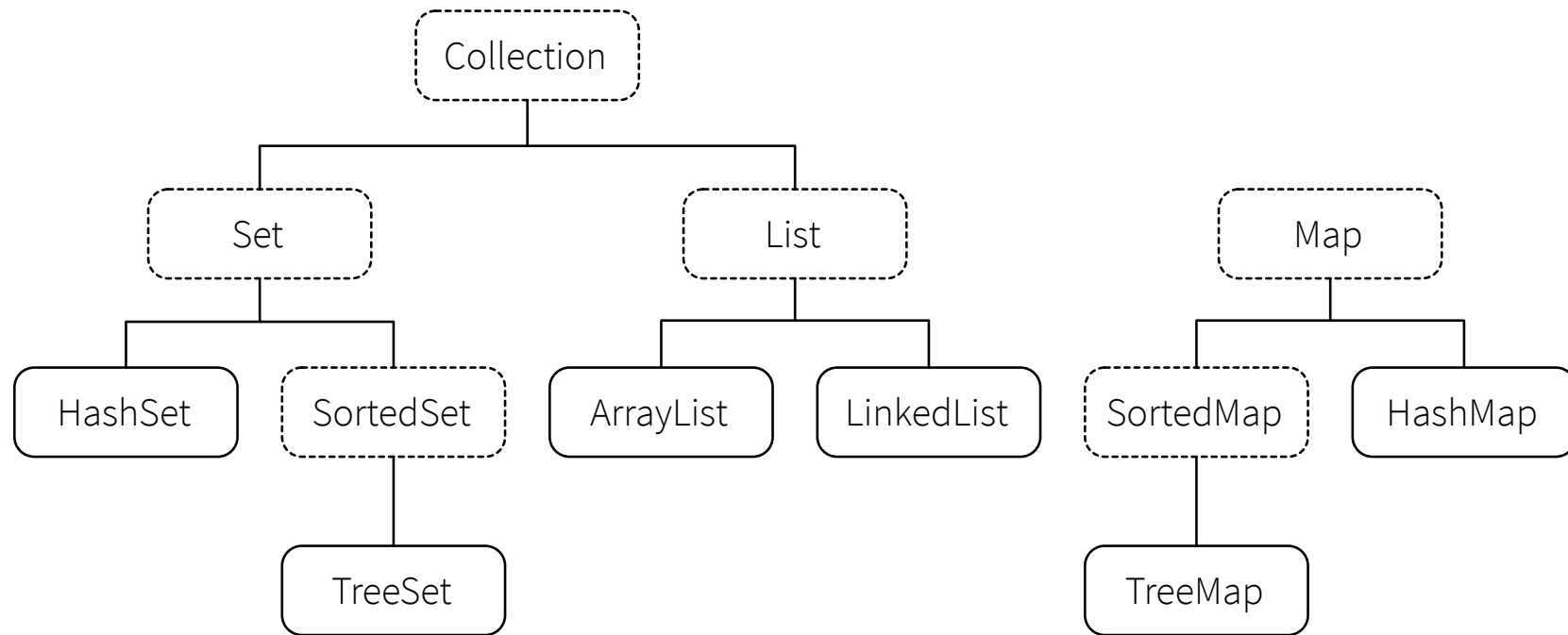
Collection Framework*



*Core Implementations



Collection Framework*



*Abbreviated Framework



Collection » List » ArrayList

- Iteration is in **insertion** order
- Operations `add(E e)`, `get()` and `set()` are **constant time*** (i.e. fast)
- Operations `add(int i, E e)`, `remove()`, and `contains()` are **linear time** (i.e. slow)
- Good **default** implementation

<http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>



Collection » List » LinkedList

- Iteration is in **insertion** order
- Double-linked list, so operations adding/removing to front or back is **constant time** (i.e. fast)
- Operations that require an index (like getting or removing at an index) are **linear time** (i.e. slow)
- Choose if need to **insert/remove** elements at front

<http://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html>



Collection » Set » HashSet

- Iteration is in **unsorted** order
 - Iteration order is not guaranteed
 - Iteration order may change over time
- Operations `add()`, `remove()`, and `contains()` are **constant time** (i.e. fast)
- Good **default** implementation

<http://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html>



Set » SortedSet » TreeSet

- Iteration is in **sorted** order
 - Iteration order may change over time
 - Can quickly navigate forward and backward
- Operations `add()`, `remove()`, and `contains()` are **$\log(n)$ time** (i.e. decent)
- Only choose if need to **maintain sorted** order

<http://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html>



Map

- Must specify **key type** and **value type**
 - e.g. `HashMap<Integer, String>`
- Keys must be **unique** and **immutable**
 - `String` may be a key
 - `ArrayList` may *not* be a key
- Values may have **duplicates** and **may change**
 - `String` and `ArrayList` may be values



Map » HashMap

- Iteration of keys is in **unsorted** order
 - Iteration order is not guaranteed
 - Iteration order may change over time
- Operations `get()` and `put()` are **constant time** (i.e. fast)
- Good **default** implementation

<http://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>



Map » SortedMap » TreeMap

- Iteration of keys is in **sorted** order
 - Iteration order may change over time
 - Can quickly navigate forward and backward
- Operations `get()` and `put()` are **log(n) time** (i.e. decent)
- Only choose if need to **maintain sorted** order

<http://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html>



Collections Class

- Not to be confused with the **Collection** interface
- Utility class of **static** methods
 - **Helper methods** like `addAll()` and `copy()`
 - **Common operations** like `binarySearch()`, `min()`, `max()`, `frequency()`, `reverse()`, `sort()`, `shuffle()`, `swap()`

<http://docs.oracle.com/javase/8/docs/api/java/util/Collections.html>





CHANGE THE WORLD FROM HERE