

CISC 3115 TY3

C25a: Sorting and Searching

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Concept of data structure
- Use data structures
 - List
 - Stack
 - Queue and priority queue
 - Set and map

Outline of This Lecture

- Sorting and searching with List, ArrayList, and LinkedList
 - The Comparator and Comparable interfaces
 - Collections and Arrays classes
 - Sorting
 - Searching

Review: The Comparator Interface

- Defined as,

```
package java.util.Comparator;
```

```
public interface Comparator<T> {
```

```
    public int compare(T lhs, T rhs);
```

```
}
```

- The compare method: returns a negative value if lhs is less than rhs; a positive value if lhs is greater than rhs; and zero if they are equal.

Review: The Collections Class

- Recall previous lecture and examples
 - Comparable and Comparators ([Lecture C20a](#))
 - Example programs: [Comparator 1](#), [Comparator 2](#), [Comparator 3](#), and [Sorting using Collections and Arrays](#)
- The Collections class contains various static methods
 - for operating on collections and maps,
 - for creating synchronized collection classes,
 - and for creating read-only collection classes

The Collections Class: Some Additional Methods

java.util.Collections

List

```
+sort(list: List): void  
+sort(list: List, c: Comparator): void  
+binarySearch(list: List, key: Object): int  
+binarySearch(list: List, key: Object, c: Comparator): int  
+reverse(list: List): void  
+reverseOrder(): Comparator  
+shuffle(list: List): void  
+shuffle(list: List, rnd: Random): void  
+copy(des: List, src: List): void  
+nCopies(n: int, o: Object): List  
+fill(list: List, o: Object): void
```

Collection

```
+max(c: Collection): Object  
+max(c: Collection, c: Comparator): Object  
+min(c: Collection): Object  
+min(c: Collection, c: Comparator): Object  
+disjoint(c1: Collection, c2: Collection): boolean  
+frequency(c: Collection, o: Object): int
```

Sorts the specified list.

Sorts the specified list with the comparator.

Searches the key in the sorted list using binary search.

Searches the key in the sorted list using binary search with the comparator.

Reverses the specified list.

Returns a comparator with the reverse ordering.

Shuffles the specified list randomly.

Shuffles the specified list with a random object.

Copies from the source list to the destination list.

Returns a list consisting of n copies of the object.

Fills the list with the object.

Returns the max object in the collection.

Returns the max object using the comparator.

Returns the min object in the collection.

Returns the min object using the comparator.

Returns true if $c1$ and $c2$ have no elements in common.

Returns the number of occurrences of the specified element in the collection.

Sorting and Searching ArrayList and LinkedList

- Examples
 - Use Comparator and Collections

Sorting

- static `<T extends Comparable<? super T>>`
void sort(List<T> list)
 - Sorts the specified list into ascending order (i.e., standard order), according to the natural ordering of its elements.
- static `<T> void sort(List<T> list, Comparator<? super T> c)`
 - Sorts the specified list according to the order induced by the specified comparator.

Sorting: Algorithms

- Question:
 - In your best case, what algorithms are not used? Selection sort, Bubble sort, or neither?
 - Actual questions being asked: "what is the most efficient way to sort 1 million 32-bit integers?"

Searching

- Sequential search
 - Iterate through the list, and find the key
 - On average, how many elements do we need to visit?
- Binary search
 - We had an [attempt](#).
 - Requirement: a list must be sorted
 - On average, how many elements do we need to visit?

Sorting and Searching: Examples

- Sorting lists using `Collections` and `Arrays` classes
 - `Comparable` interface
 - `Comparator` interface
- Searching
 - Binary search on sorted lists
 - Compare performance between binary search on "random access" lists and that on non-"random access" lists

Questions?

- Collections and Arrays
- Comparator
- Comparable
- Use sorting and searching
- Binary search on lists (random access or non-random access)?