CSD-4464 Java EE

Class 5: Collections

Lambton College

Collections

- High performance Interface for handling data structures
- Includes various types e.g Maps, Lists, Sets, Tree
- Built in high performance algorithms to allow quicker development
- The Collection interface includes many great methods for manipulating the collection, such as Stream(), forEach(), contains(), parallelStream()

Lists

- This implements **Collection** and an instance of List stores an ordered collection of elements.
- Elements can be inserted or accessed by their position in the list, using a zero-based index.
- A list may contain duplicate elements.
- Think of Lists as a more featureful version of arrays.
- Creation Syntax

```
List<Type> myList = new ArrayList<>();
```

Lists sort

- void sort(Comparator<? super T> c)
- If the specified comparator is null then all elements in this list must implement the Comparable interface and the elements' natural ordering should be used.
- A comparator is a function that compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.
- Implements a variation of Mergesort

Maps

- The Map interface maps unique keys to values. A key is an object that you use to retrieve a value at a later date.
- Extremely fast lookups by Key
- Creation syntax

```
Map<KeyType, ValueType> myMap = new HashMap<>();
```

Sets

- A Set is an array style Collection that cannot contain duplicate elements
- The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited
- Order has no guarantees*
- Creation Syntax

```
Set<Type> mySet = new HashSet<>(); //creates a HashSet which implements Set
```

*unless using a SortedSet