Chapter 19 Java Data Structures

1. The Java Collections Framework defines the Java API for handling common data structures tasks in Java. It defines classes and interfaces for storing and manipulating data in sets, lists, and maps.

2. By defining these two methods in the Collection interface, the instances of Collection can use these two methods. For instance, you may have a method with a parameter of the Collection type. This parameter can use the hashCode method and the equals method since the methods are in the Collection interface.

3. The default implementation of the equals method is to compare the references of this object with the other. The default implementation of the hashCode method returns the object's memory address of the object.

4. The Set is an interface. To create an instance of Set, you need to use HashSet or TreeSet. To insert an element to a set, use the add method. To remove an element from a set, use the remove method. To find the size of a set, use the size() method. To traverse a set, use the iterator method to obtain an iterator. You can then traverse the set through the iterator.

5. HashSet is unsorted, but TreeSet is sorted. HashSet is more efficient than TreeSet if you don't want the elements in a set to be sorted. If you create a TreeSet using its default constructor, the compareTo method is used to compare the elements in the set, assuming that the class of the elements implements the Comparable interface. To use a comparator, you have to use the constructor TreeSet(Comparator comparator) to create a sorted set that uses the compare method in the comparator to order the elements in the set.

6. Use the add or remove method to add or remove elements from a list. Use the listIterator() to obtain an iterator. This iterator allows you to traverse the list bi-directional.

7. ArrayList and LinkedList can be operated similarly. The critical differences between them are their internal implementation, which impacts the performance. ArrayList is efficient for retrieving elements, and for adding and removing elements from the end of the list. LinkedList is efficient for adding and removing elements anywhere in the list.

8. Vector is the same as ArrayList except that, except that Vector contains the synchronized methods for accessing and modifying the vector. Since Vector implements List, you can use the methods in List to add, remove elements from a vector, and use the size() method to find the size of a vector. To create a vector, use either its constructors.

9. Stack is a subclass of Vector. The Stack class represents a last-in-first-out stack of objects. The elements are accessed only from the top of the stack. You can retrieve, insert, or remove an element from the top of the stack. To add a new element to a stack, use the push method. To remove an element from the top of the stack, use the method pop. To find a stack size, use the size() method.

10. Map is an interface. To create an instance of Map, you need to use the HashMap class or the TreeMap class. The HashMap and TreeMap have various constructors that you can use to create a HashMap or a TreeMap. You can use the put method to add an entry to a map, and the remove method to remove an entry with the specified key from the map. Use the size method to find the size of a map.

11. 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. For instance, you can use the sort method to sort a list. The Arrays class contains various static methods for sorting and searching arrays, for comparing arrays, and for filling array elements. It also contains a method for converting an array to a list.