I think this is the most extraordinary collection of talent, of human knowledge, that has ever been gathered together at the White House—with the possible exception of when Thomas Jefferson dined alone.
—John F. Kennedy

The shapes a bright container can contain!
—Theodore Roethke

Journey over all the universe in a map.
—Miguel de Cervantes

Not by age but by capacity is wisdom acquired.
—Titus Maccius Plautus

It is a riddle wrapped in a mystery inside an enigma.
—Winston Churchill

20

Generic Collections

Objectives

In this Chapter you'll learn:

- What collections are.
- To use class `Arrays` for array manipulations.
- To form linked data structures using references, self-referential classes and recursion.
- The type-wrapper classes that enable programs to process primitive data values as objects.
- To use the collections framework (prebuilt data structure) implementations.
- To use collections framework methods (such as `search`, `sort` and `fill`) to manipulate collections.
- To use the collections framework interfaces to program with collections polymorphically.
- To use iterators to "walk through" a collection.
- To use persistent hash tables manipulated with objects of class `Properties`.
- To use synchronization and modifiability wrappers.

Self-Review Exercises

20.1 Fill in the blanks in each of the following statements:

a) A(n) _____ is used to iterate through a collection and can remove elements from the collection during the iteration.

ANS: `Iterator`.

b) An element in a `List` can be accessed by using the element's _____.

ANS: index.

c) Assuming that `myArray` contains references to `Double` objects, _____ occurs when the statement "`myArray[ 0 ] = 1.25;`" executes.

ANS: autoboxing.

d) Java classes _____ and _____ provide the capabilities of arraylike data structures that can resize themselves dynamically.

ANS: `ArrayList`, `Vector`.

e) If you do not specify a capacity increment, the system will _____ the size of the `Vector` each time additional capacity is needed.

ANS: double.

f) You can use a(n) _____ to create a collection that offers only read-only access to others while allowing read/write access to yourself.

ANS: unmodifiable wrapper.

g) Assuming that `myArray` contains references to `Double` objects, _____ occurs when the statement "`double number = myArray[ 0 ];`" executes.

ANS: auto-unboxing.

h) Algorithm _____ of `Collections` determines whether two collections have elements in common.

ANS: `disjoint`.

20.2 Determine whether each statement is *true* or *false*. If *false*, explain why.

a) Values of primitive types may be stored directly in a collection.

ANS: False. Autoboxing occurs when adding a primitive type to a collection, which means the primitive type is converted to its corresponding type-wrapper class.

b) A `Set` can contain duplicate values.

ANS: False. A `Set` cannot contain duplicate values.

c) A `Map` can contain duplicate keys.

ANS: False. A `Map` cannot contain duplicate keys.

d) A `LinkedList` can contain duplicate values.

ANS: True.

e) `Collections` is an `interface`.

ANS: False. `Collections` is a `class`; `Collection` is an `interface`.

f) `Iterators` can remove elements.

ANS: True.

g) With hashing, as the load factor increases, the chance of collisions decreases.

ANS: False. As the load factor increases, fewer slots are available relative to the total number of slots, so the chance of a collision increases.

h) A `PriorityQueue` permits `null` elements.

ANS: False. Attempting to insert a `null` element causes a `NullPointerException`.

Execises

NOTE: Solutions to the programming exercises are located in the **`ch20solutions`** *folder. Each exercise has its own folder named* **`ex20_##`** *where ## is a two-digit number representing the exercise number. For example, exercise 20.17's solution is located in the folder* **`ex20_17`**.

20.3 Define each of the following terms:

a) `Collection`

ANS: Interface `Collection` is the root interface in the collection hierarchy from which interfaces `Set`, `Queue` and `List` are derived.

b) `Collections`

ANS: Class `Collections` provides `static` methods that manipulate collections polymorphically. These methods implement algorithms for searching, sorting, and so on.

c) `Comparator`

ANS: An interface that specifies how collections objects are ordered.

d) `List`

ANS: An interface implemented by ordered collections that allow duplicate elements (sequences), such as `ArrayList`, `LinkedList` and `Vector`.

e) load factor

ANS: The ratio of the number of occupied cells in a hash table to the size of the hash table.

f) collision

ANS: A situation where two different keys hash into the same cell.

g) space/time trade-off in hashing

ANS: When the load factor is increased, the result is better memory utilization. However, the program runs slower due to increased hashing collisions.

h) `HashMap`

ANS: Java utilities package class that enables programmers to use a hash table to store key/value pairs. Unlike `Hashtable`, it is not synchronized and permits `null` keys and values.

20.4 Explain briefly the operation of each of the following methods of class `Vector`:

a) `add`

ANS: Appends the specified element to the end of this `Vector`.

b) `set`

ANS: Replaces the element at the specified position.

c) `remove`

ANS: Removes the first occurrence of an element from the `Vector`.

d) `removeAllElements`

ANS: Removes all `Vector` elements and sets its size to zero.

e) `removeElementAt`

ANS: Removes the element at the specified position.

f) `firstElement`

ANS: Returns a reference to the first element in the `Vector`.

g) `lastElement`

ANS: Returns a reference to the last element in the `Vector`.

h) `contains`

ANS: Determines whether a `Vector` contains a specified object.

i) `indexOf`

ANS: Returns the position of the first occurrence of a specified object, or `-1` if the object does not occur in the `Vector`.

j) `size`

ANS: The current number of elements in the `Vector`.

k) `capacity`

ANS: The number of elements available for storage (used and unused).

20.5 Explain why inserting additional elements into a `Vector` object whose current size is less than its capacity is a relatively fast operation and why inserting additional elements into a `Vector` object whose current size is at capacity is a relatively slow operation.

ANS: A `Vector` whose current size is less than its capacity has memory available. Insertions are fast because new memory does not need to be allocated. A `Vector` that is at its capacity must have its memory reallocated and the existing values copied into it.

20.6 By extending class `Vector`, Java's designers were able to create class `Stack` quickly. What are the negative aspects of this use of inheritance, particularly for class `Stack`?

ANS: Operations can be performed on `Stack` objects that are not normally allowed, which can lead to corruption of the stack. For example, `Vector` method `insertElementAt` should not be performed on `Stack`.

20.7 Briefly answer the following questions:

a) What is the primary difference between a `Set` and a `Map`?

ANS: `Maps` contain both the key and the value, and `Sets` contain only values. `Sets` contain unique values; `Maps` contain unique keys but may contain duplicated values.

b) What happens when you add a primitive type (e.g., `double`) value to a collection?

ANS: Autoboxing occurs when a primitive type value is added to a collection. For example, a `double` primitive type value is converted to an object of its wrapper class `Double`.

c) Can you print all the elements in a collection without using an `Iterator`? If yes, how?

ANS: Yes, the elements in a collection can be printed by iterating through the collection with the enhanced `for` statement. Alternatively, you can use the collection's `toString` method.

20.8 Explain briefly the operation of each of the following `Iterator`-related methods:

a) `iterator`

ANS: Returns an `Iterator` for a collection.

b) `hasNext`

ANS: Determines whether a collection has another element.

c) `next`

ANS: Returns the next element in a collection.

20.9 Explain briefly the operation of each of the following methods of class `HashMap`:

a) `put`

ANS: Adds a key/value pair into the `HashMap`.

b) `get`

ANS: Locates the value associated with the specified key.

c) `isEmpty`

ANS: Returns a `boolean` value indicating whether or not the `HashMap` is empty.

d) `containsKey`

ANS: Determines whether specified key is in the `HashMap`.

e) `keySet`

ANS: Returns a `Set` of the keys in the `HashMap`.

20.10 Determine whether each of the following statements is *true* or *false*. If *false*, explain why.

a) Elements in a `Collection` must be sorted in ascending order before a `binarySearch` may be performed.

ANS: True.

b) Method `first` gets the first element in a `TreeSet`.

ANS: True.

c) A `List` created with `Arrays` method `asList` is resizable.

ANS: False. The `List` is fixed length.

20.11 Explain the operation of each of the following methods of the `Properties` class:

a) `load`
ANS: Reads the contents from a specified `InputStream`.
b) `store`
ANS: Writes the contents to a specified `OutputStream`.
c) `getProperty`
ANS: Returns the value associated with a key.
d) `list`
ANS: Displays the contents of a `Properties` object to the specified output stream.