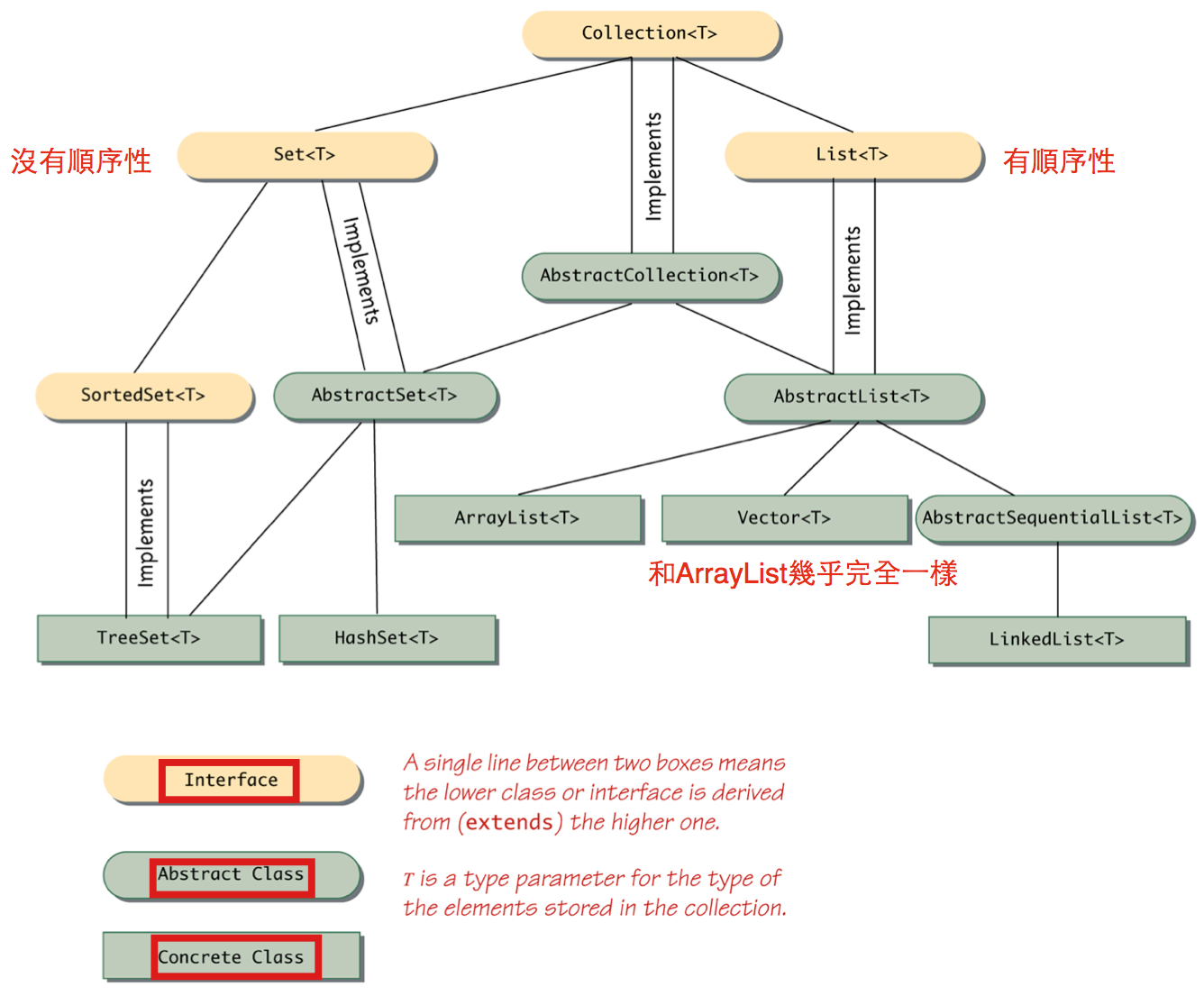
Chapter 16 Collections, Maps and Iterators

1. Java Collection
   1. any class that holds objects and implements the Collection interface
   2. ArrayList<T> implements all the methods in the Collection interface
   3. Collection is the highest level of Java’s framework for collection classes



1. Wildcards (【電腦】萬用字元，通配符)
   1. no specified type parameter, use “?” to be a wide range of argument types
   2. syntax:

public void method(String arg1, ArrayList<?> arg2)

* 1. bounded wildcards: specifying the type to be an ancestor type or descendent type of some class or interface

<? extends String> //descendent class of String

<? super String> //ancestor class of String

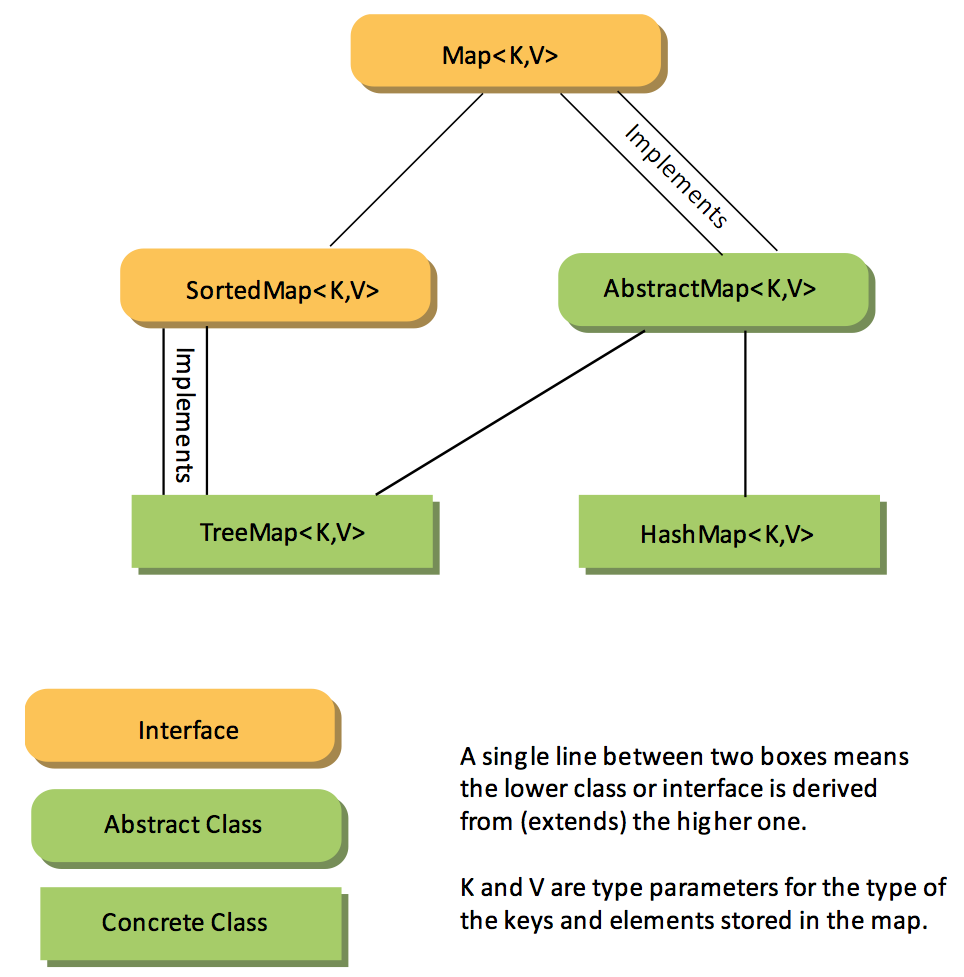
* 1. <個人補充>wildcards vs type parameter

<N extends Number> Collection<N> getThatCollection(Class<N> type)

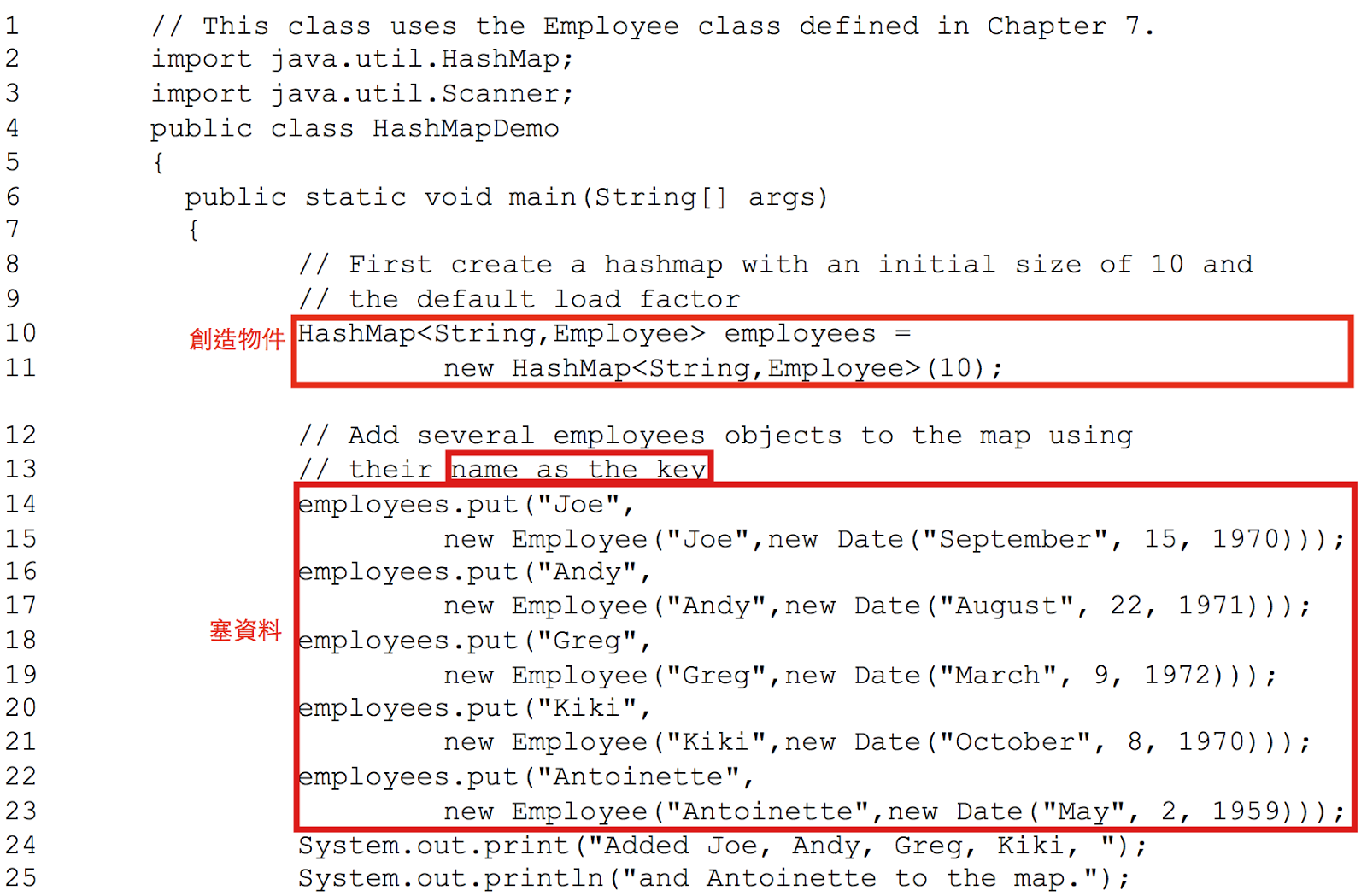
Collection<? extends Number> getThatCollection(Class<? extends Number>)

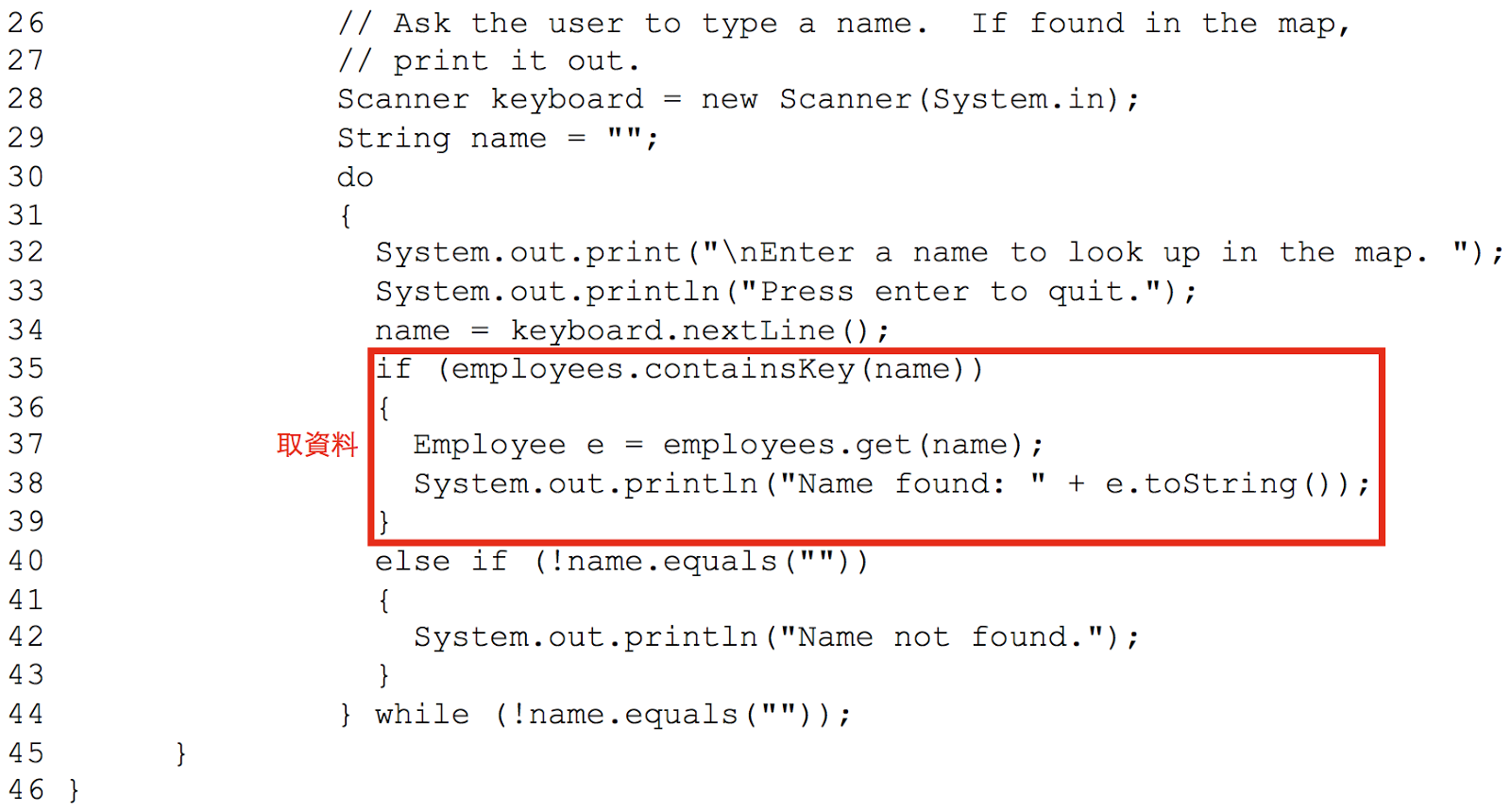
* + 1. first declaration: passed argument and return type are the same type
    2. second declaration: passed argument and return type doesn’t have to be the same type

1. Collection Framework
   1. Collection<T> interface describes the basic operations that all collection classes should implement
   2. method headings example
      1. boolean isEmpty()
      2. public boolean contains(Object target)
      3. public boolean containsAll(Collection<?> collectionOfTargets)
      4. public boolean equals(Object other)
      5. public int size()
      6. Iterator<T> iterator()
      7. public Object[] toArray()
      8. public <E> E[] toArray(E[] a)
      9. public int hashCode()
      10. public boolean add(T element) (Optional)
      11. public boolean addAll(Collection<? extends T> collectionToAdd()) (Optional)
      12. public boolean remove(Object element) (Optional)
      13. public void clear() (Optional)
      14. public boolean retainAll(Collection<?> saveElements) (Optional)
   3. “Optional” operation means the method does not completely implement its intended semantics and still has to be implemented!!
   4. If a trivial implementation is given, the method body should throw an UnsupportedOperationException
2. Concrete Collections Classes
   1. Set<T> interface has concrete classes such as…
      1. HashSet<T>
      2. TreeSet<T>
   2. List<T> interface has concrete classes such as…
      1. ArrayList<T>
      2. Vector<T>
      3. LinkedList<T>
3. Map Framework
   1. deals with collections of ordered pairs(key and associated value)



* 1. method headings
     1. boolean isEmpty()
     2. public boolean containsValue(Object value)
     3. public boolean containsKey(Object key)
     4. public boolean equals(Object other)
     5. public int size()
     6. public int hashCode()
     7. public Set<Map.Entry<K,V>> entrySet()
     8. public Collection<V> values()
     9. public V get(Object key)
     10. public V put(K key, V value) (Optional)
     11. public void putAll(Map<? extends K, ? extends V> mapToAdd) (Optional)
     12. public V remove(Object key) (Optional)
  2. unordered map
     1. HashMap<K,V>
        1. load factor(between 0 and 1): If number of elements in hash table exceeds the load factor, then the capacity of the hash table is automatically increased (default is 0.75)
        2. initial capacity is 16





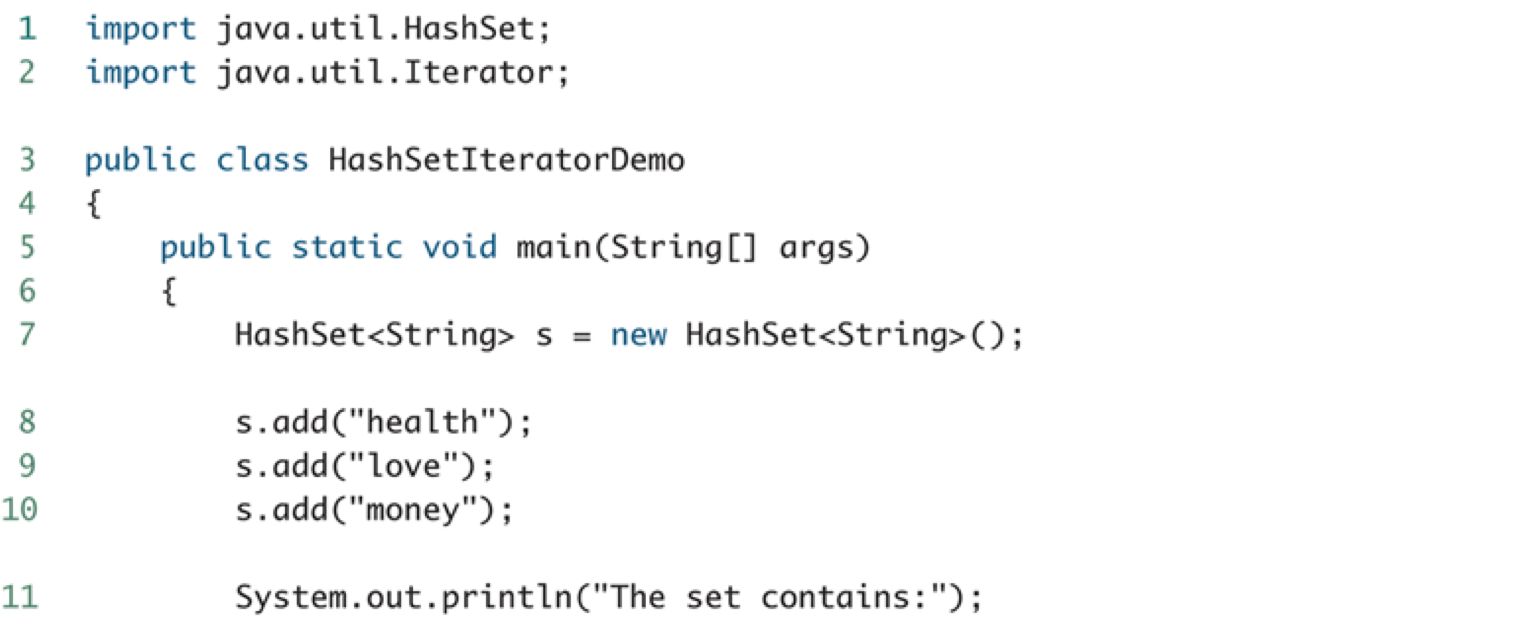
* + - 1. if want to use custom class as parameterized type K, then custom class need to override…

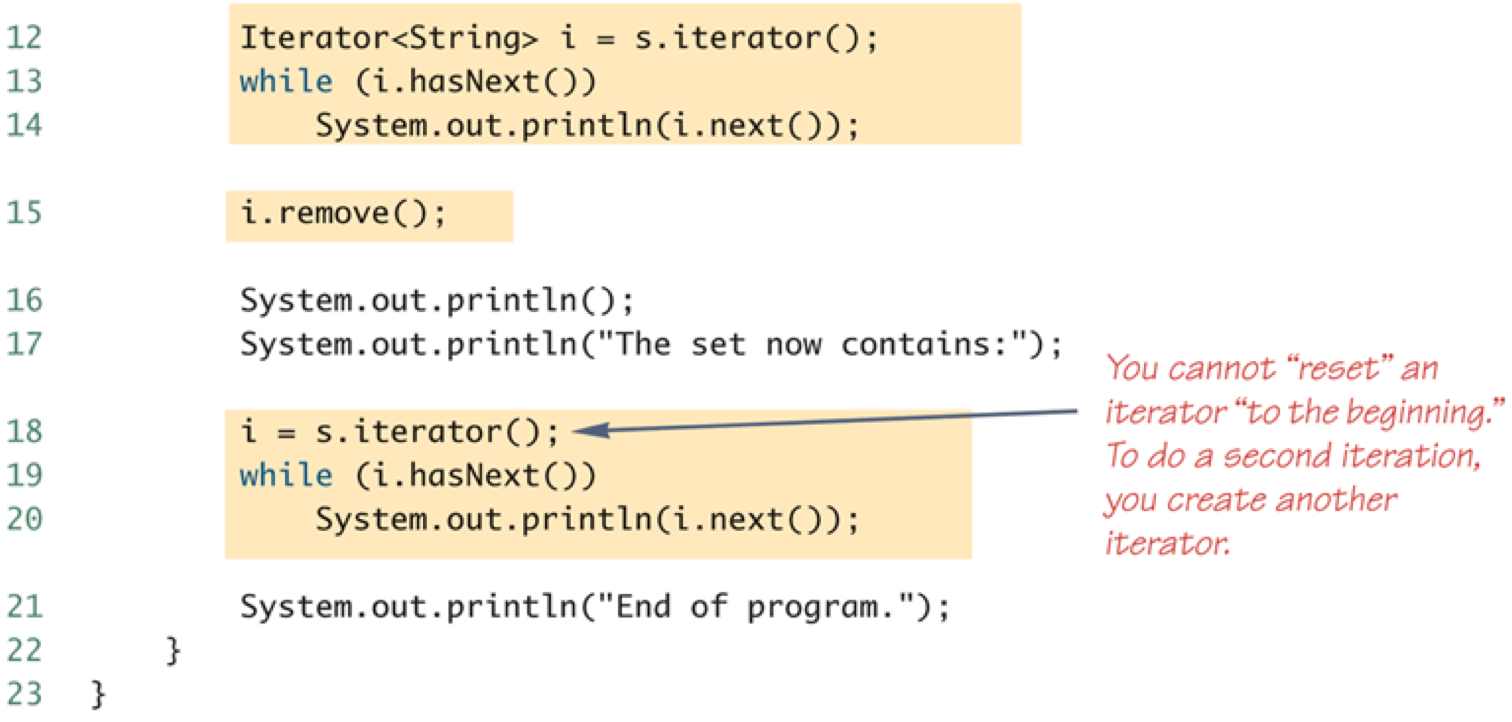
public int hashCode();

public boolean equals(Object obj);

* 1. ordered map
     1. TreeMap<K, V>
     2. LinkedHashMap<K, V>

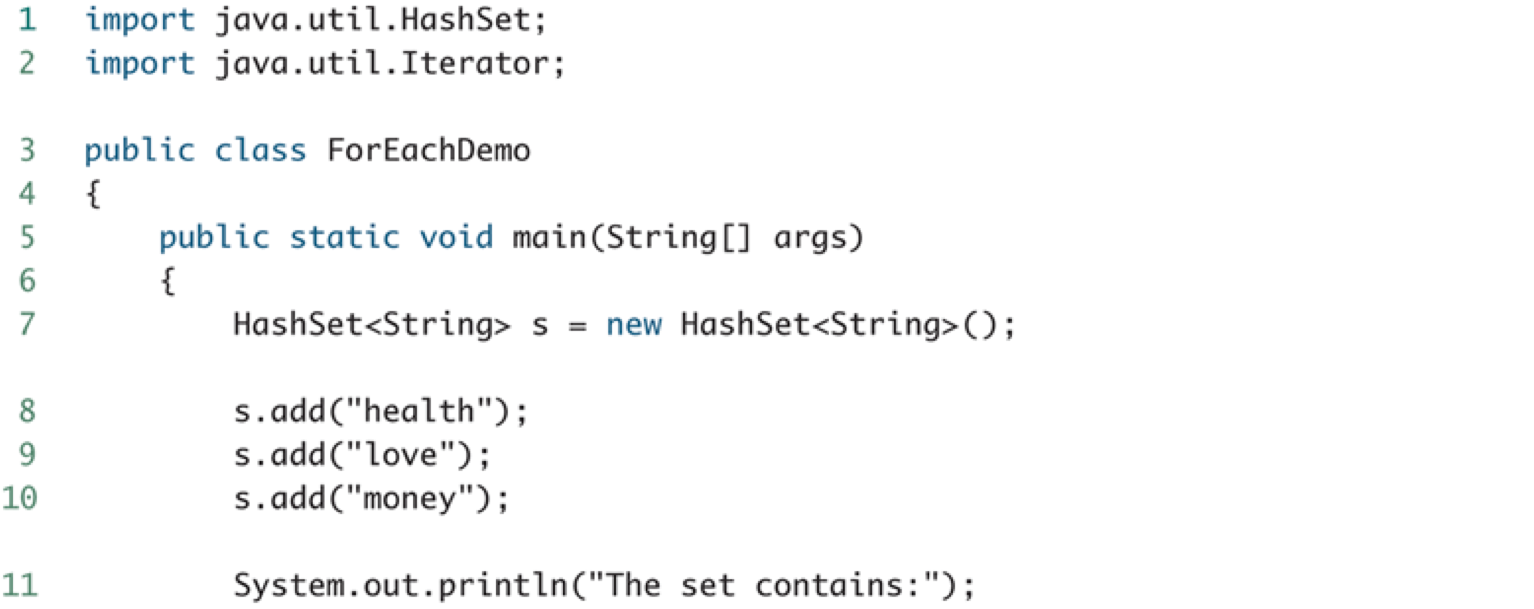
1. Iterators Interface
   1. object that provide sequential access to the collection elements
   2. method headings:
      1. public T next()
      2. public boolean hasNext()
      3. public void remove() (Optional)
   3. HashSet<T> object imposes no order on the elements it contains, but iterator will impose an order on the elements in the hash set

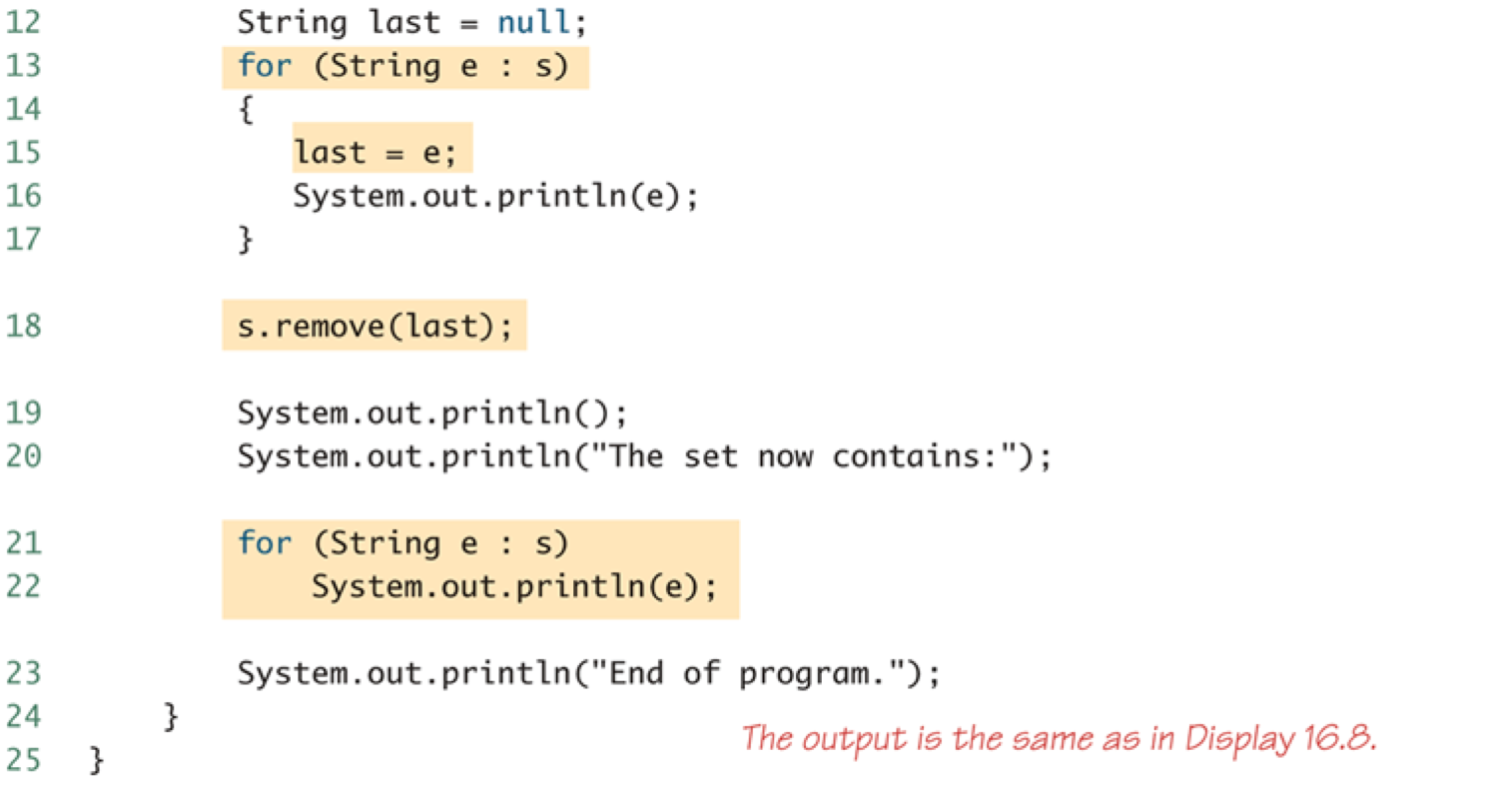




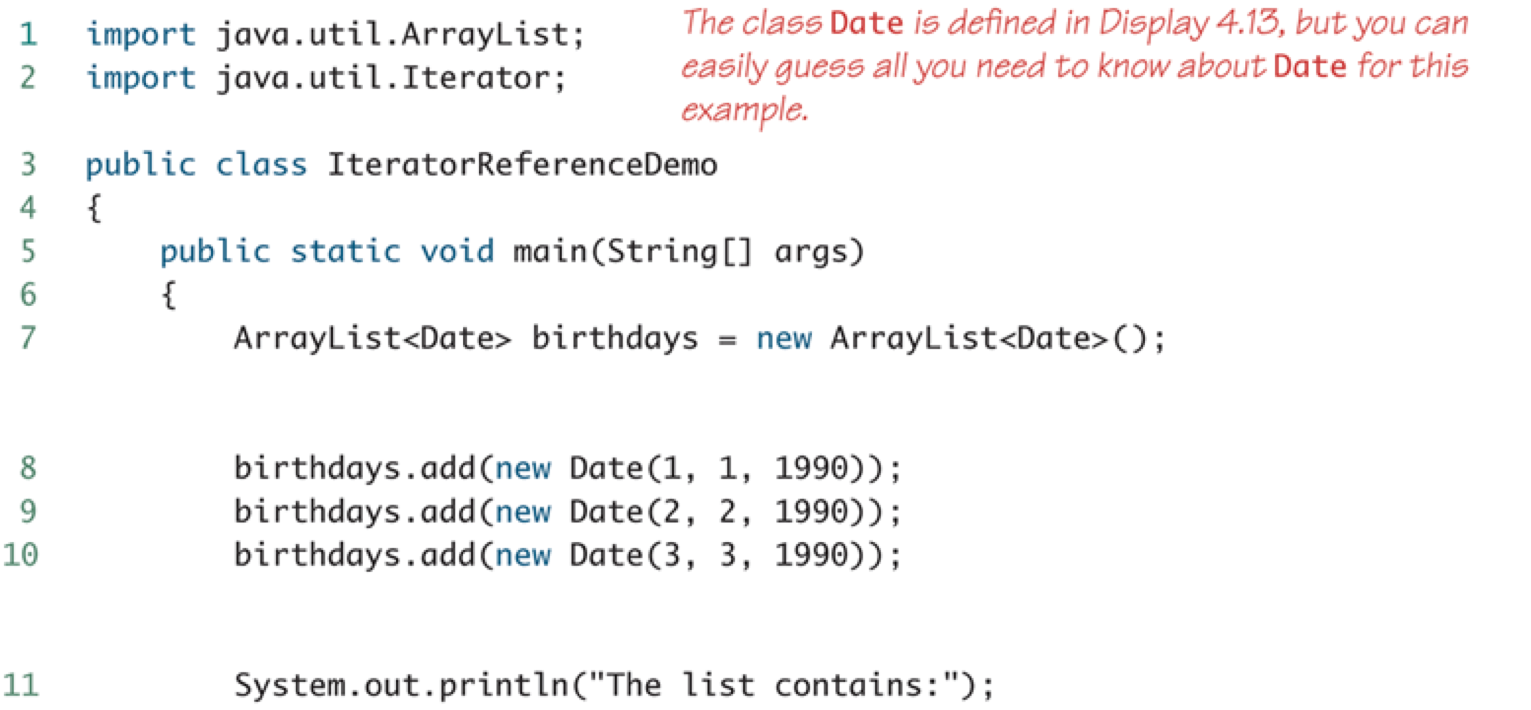


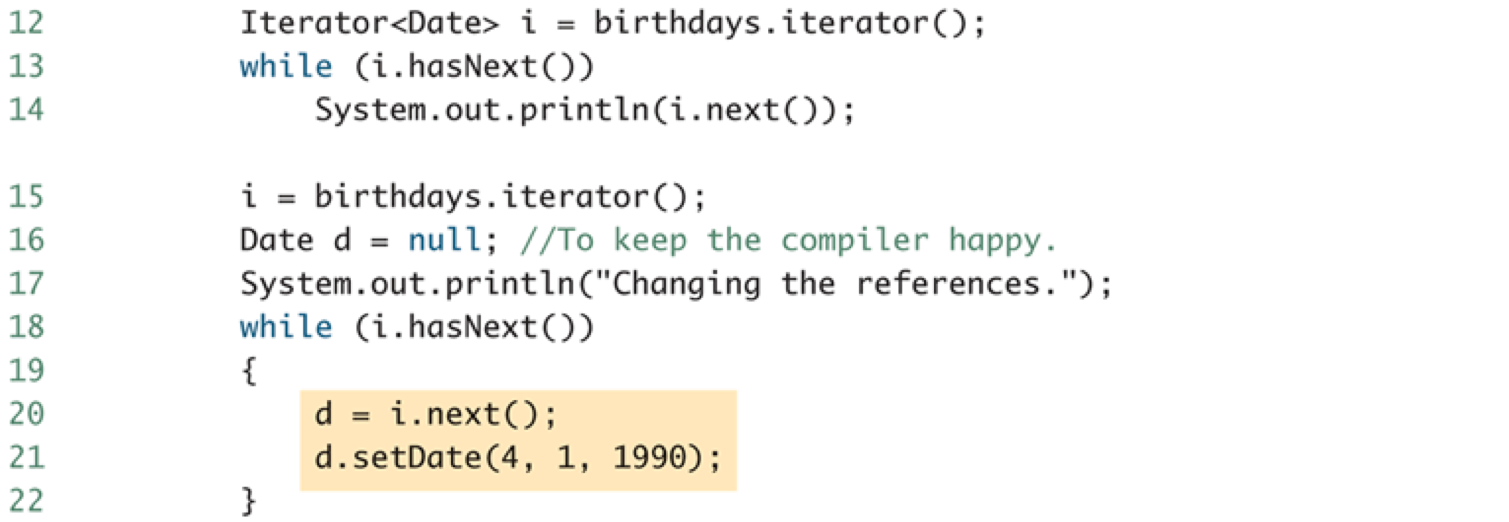
* 1. For-Each Loops can serve the same purpose as an iterator, same results as above

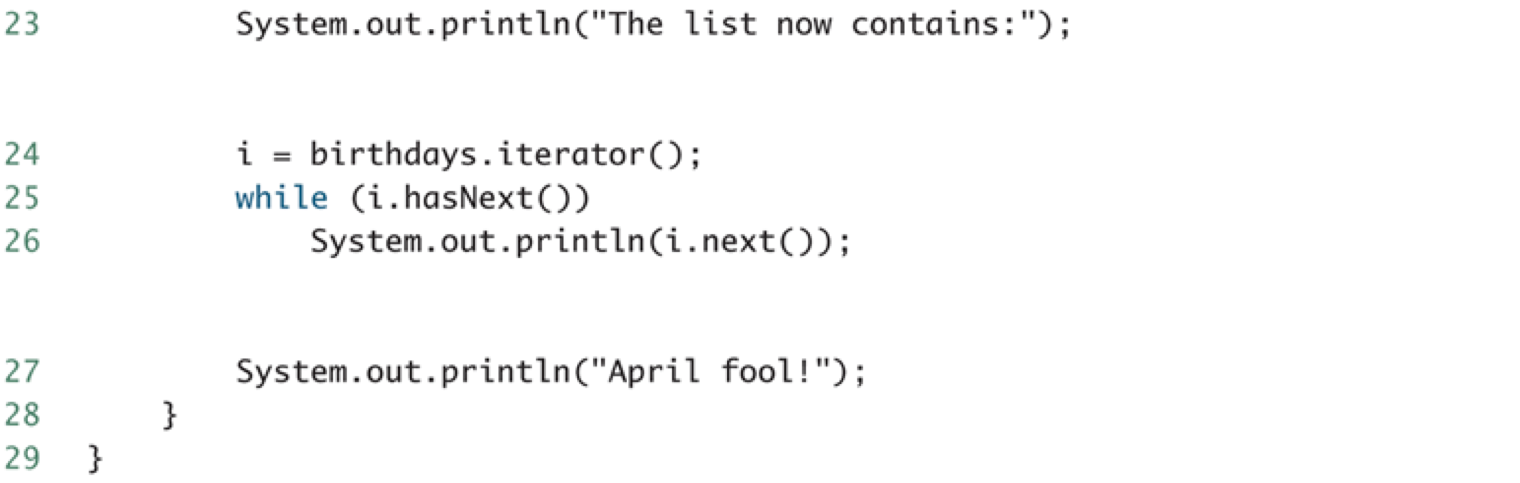


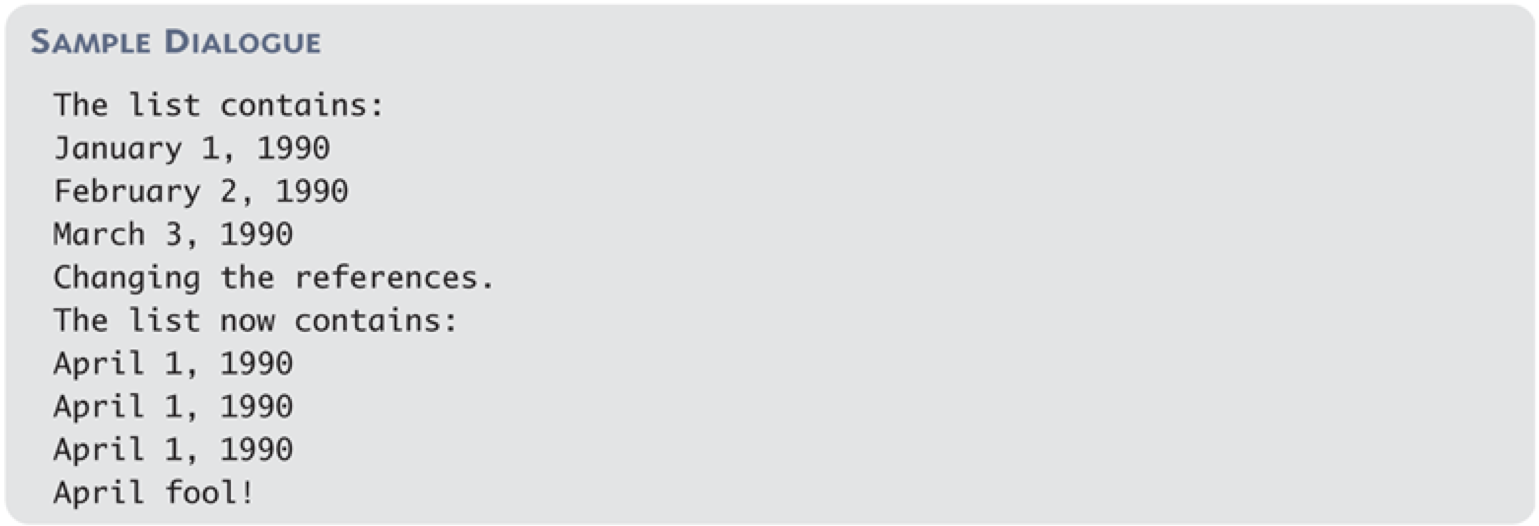


1. ListIterator<T> interface
   1. extends Iterator<T> interface
   2. designed to work with collections that satify the List<T> interface
   3. can move in either direction
   4. method headings
      1. public T next()
      2. public T previous()
      3. public boolean hasNext()
      4. public int nextIndex()
      5. public int previoudIndex()
      6. public void add(T newElement) (Optional)
      7. public void remove() (Optional)
      8. public void set(T newElement) (Optional)
   5. for example:









1. Programmer-defined Iterator
   1. suggested: derive from library collection classes
   2. if collection class must be customized in some other way, then an iterator class should be defined as an inner class of the collection class