What is a collection?
- A number of things that are grouped together in some way, e.g.,
- A grocery cart contains all of the items that a customer wants to buy
- A video store contains videos possibly grouped by genre, alphabetical order, ...
- A dictionary lists words along with their definition
- A class list contains student names
- Different kinds, e.g. duplicates/no duplicates, ordered/non ordered
- Java offers several classes to support the concept of collections

Some collections in Java
- ArrayList: collection whose elements are in a specific order (given by the index)
- Vector: same as ArrayList, but designed to be used safely in a multithreaded environment (synchronization).
- HashMap: collection of key/value pairs (as in a dictionary). The HashMap uses a hashcode to store the items in the collection (makes the retrieve operation efficient).
- Available in the package java.util

Typical ArrayList operations (1)
- Items in an ArrayList are ordered by their index value (starts at 0)
  // append at the end of the list
  public boolean add(Object o)
  // remove the Object at location index
  public Object remove(int index)
  // insert at location index
  public boolean add(int index, Object o)

Typical ArrayList operations (2)
- Getting an element from an ArrayList
  public Object get(int index)
  - always get an Object (what get returns)
  - To get back an instance with the actual type use a cast:
    ArrayList l = new ArrayList();
    l.add(new String("ABC"));
    String s = l.get(0); // Error
    String s = (String)l.get(0); // OK
- Other common methods
  public boolean contains(Object o)
  public int size()
  public boolean isEmpty()
  Iterator iterator() // see next slide

Iterating through an ArrayList
- Using the index value
  for(int i=0; i<l.size(); i++){
    Object o = l.get(i);
    // process o
  }
- Using an iterator
  Iterator i = l.iterator();
  while(i.hasNext()){
    Object o = i.next();
    // process o
  }
- An iterator works with any type of collection
The Collections class

- A powerful class that contains many static methods to operate on many types of collections (e.g., to synchronize, to make a collection read only, to sort...)
- e.g., to sort

```java
// to sort items that can be compared
public static void sort(List list) // Note: an ArrayList is a List
// to sort items according to some supplied
// comparator
public static void sort(List list, Comparator c)
```

ArrayList example

- Input and print a class list in alphabetical order

```java
// l is the list of students
ArrayList l = new ArrayList();
// Get the students’ names
String s;
do{
    s=JOptionPane.showInputDialog(null, "Student Name");
    if (s!=null) l.add(s);
}while(s!=null);
// Sort the list in alphabetical order
Collections.sort(l);
// Print the list
Iterator i = l.iterator();
while(i.hasNext())
    System.out.println(i.next());
```

HashMap

- Example: the IRS list of taxpayers can be thought of as a map. Two items
- key: SSN
- value: taxpayer’s info (name, income...)
- Typical HashMap operations
  - put returns the old value or null if none
  - get(Object key)
  - remove(Object key)
  - size()
- What is a hash?
  Store the values in a table. The location of a value in the table is determined by a function (the hash function) applied on the key.

HashMap example

- A phone book

```java
HashMap d = new HashMap();
// Create the phone book
String r,s;
do{
    r=JOptionPane.showInputDialog(null,"Name");
    s=JOptionPane.showInputDialog(null,"Number");
    if (s!=null && r!=null)
        d.put(r,s);
}while(s!=null && r!=null);
// Use the phone book
r=JOptionPane.showInputDialog(null,"Name");
System.out.println("The phone number of " +r+ " is " + d.get(r));
```

Other collections

- HashSet
  - for an unordered collection with no duplicates
- TreeMap
  - for an ordered map (the ordering is done on the keys)