# Fundamentos de Programación

## Colecciones de Objetos

José A. Mañas <jmanas@dit.upm.es>
Dpto. de Ingeniería de Sistemas Telemáticos
**http://www.dit.upm.es/~pepe/doc/fprg/**

3.11.2010

# colecciones

- Cuando tenemos varios objetos del mismo tipo (misma clase)
- De tamaño variable, adaptándose a lo que haga falta
  - listas
  - ArrayList< TIPO >
- De tamaño fijo: N elementos
  - arrays
  - TIPO [ ]

# listas

- Cuando tenemos muchos datos del mismo tipo
  - listas de tipo T
  - ArrayList<T>
- Operaciones
  - añadir elementos
  - eliminar elemento
  - acceder al elemento n-ésimo
  - ¿cuántos elementos?
  - recorrer los elementos

# ArrayList<T>

- Constructor

```
import java.util.ArrayList;

public class Agenda {
  private ArrayList<String> entradas;

  public Agenda() {
    entradas = new ArrayList<String>();
  }
```

# ArrayList<T>

- ¿Cuántos elementos hay?

```
public int numeroDeNotas() {
   return entradas.size();
}
```

# ArrayList<T>

- Añadir elementos
  - se añaden al final de la lista

```
public void guardaNota(String nota) {
   entradas.add(nota);
}
```

# ArrayList<T>

- Acceso al elemento n-ésimo
  - cuenta empezando en 0

```
public String sacaNota(int n) {
  if (n >= 0 && n < numeroDeNotas()) {
    return entradas.get(n);
  }
  return null;
}
```

# ArrayList<T>

- Elimina el elemento n-ésimo
  - cuenta empezando en 0

```
public void eliminaNota(int n) {
   if (n >= 0 && n < numeroDeNotas()) {
      entradas.remove(n);
   }
}
```

# + métodos

- void clear()
- boolean isEmpty()
- void add(int posicion, TIPO elemento)
- TIPO set(int posicion, TIPO elemento)
- boolean contains(TIPO elemento)
- int indexOf(TIPO elemento)

- http://java.sun.com/javase/6/docs/api/java/util/ArrayList.html

# recorrido

- Procesamos todos los términos
  - si sabemos cuántos elementos tiene ...

```
public void imprimeAgenda() {
   System.out.println(entradas.get(0));
   System.out.println(entradas.get(1));
   ...
}
```

# recorrido

*bucle for-each*

- Procesamos todos los términos
  - **iteración**: for each

```
public void imprimeAgenda() {
  for (String nota : entradas)
    System.out.println(nota);
}
```

# recorrido

*bucle for*

- Procesamos todos los términos
  - bucle con contador

```
public void imprimeAgenda() {
   for (int i = 0; i < entradas.size(); i++) {
     System.out.println(entradas.get(i));
   }
}
```

# recorrido

*bucle while*

- Procesamos todos los términos
  - bucle cero o más

```
public void imprimeAgenda() {
   int i = 0;
   while (i < entradas.size()) {
      System.out.println(entradas.get(i));
      i++;
   }
}
```

# recorrido

*bucle do-while*

- Procesamos todos los términos
  - bucle una o más

```
public void imprimeAgenda() {
   int i = 0;
   do {
      System.out.println(entradas.get(i));
      i++;
   } while (i < entradas.size());
}
```

# recorrido

*iterador*

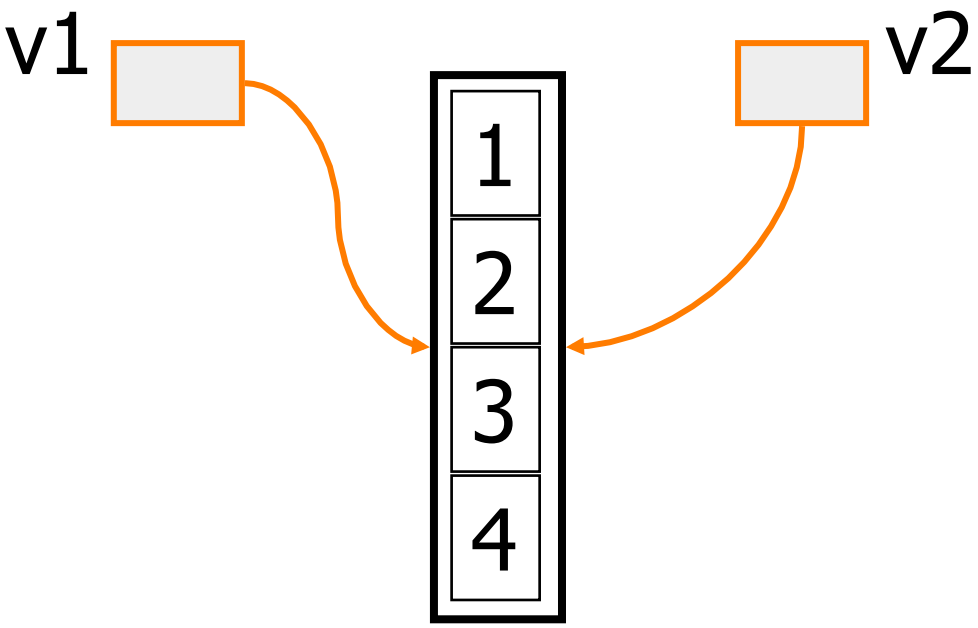
- Procesamos todos los términos
  - iteración: con Iterator<TIPO>

```
import java.util.Iterator;
```

```
public void imprimeAgenda() {
   Iterator<String> it = entradas.iterator();
   while (it.hasNext()) {
      String nota = it.next();
      System.out.println(nota);
   }
}
```

# Contenido compartido

- Podemos compartir contenido
  - ArrayList<T> v2 = v1;




=> paso de referencia en llamada a métodos

# Copia del contenido

- Podemos hacer una copia independiente

```
ArrayList<T> v2= new ArrayList<T>();
for (T v: v1)
   v2.add(v);
```

```
ArrayList<T> v2= new ArrayList<T>(v1);
```

# Igualdad

- TRUE si se trata del mismo array
  - v1 == v2
- Para comparar contenidos

```
boolean compara(ArrayList<T> v1, ArrayList<T> v2) {
   if (v1 == null || v2 == null) return false;
   if (v1.size() != v2.size()) return false;
   for (int i= 0; i < v1.size(); i++)
      if (v1.get(i) != v2.get(i)) return false;
   return true;
}
```

# ejemplo

- crear una listapara los 100 primeros enteros positivos y cargarlo con el cuadrado de cada uno
  - [ 0, 1, 4, 9, 16, 25, 36, 49, ... ]

# ejemplo

- crear una listade boolean para los 100 primeros enteros positivos, siendo true si es impar
  - [ false, true, false, true, false, true, ... ]

# ejemplo

- crear una listade boolean para los 100 primeros enteros positivos, y llenarlo de números aleatorios entre 0 y 1000
  - [ 30, 17, 514, 216, 701, 0, ... ]

# ejemplo

- escriba un método que recibe una lista y un número y los multiplica
  - [ 30, 17, 514, 216, 701, 0, ... ] * 2
  - [ 60, 34, 1028, 432, 1402, 0, ... ]

# ejemplo

- escriba un método que recibe una lista y devuelve el valor mínimo
  - [ 30, 17, 514, 216, 701, 0 ]
  - 0
- algo similar, devolviendo la posición del mínimo
  - 5

# ejemplo

- dado una lista, calcular la suma de sus valores
  - [ 30, 17, 514, 216, 701, 0 ]
  - 1478

# ejemplo

- dada una lista
  - [ 30, 17, 514, 216, 701, 0 ]
- invertir sus términos de orden
  - [ 0, 701, 216, 514, 17, 30 ]
- de dos formas:
  - generando otra lista
  - reordenar en la misma lista

# ejemplo

- dada una lista
  - [ 30, 17, 514, 216, 701, 0 ]
- permutarla aleatoriamente
  - [ 0, 701, 17, 514, 30, 216 ]

# ejemplo

- dada una lista
  - [ 30, 17, 514, 216, 701, 0 ]
- escribir un método que devuelve el valor medio
  - 246.33

# ejemplo

- escribir un método que determine si una lista está en orden creciente o no

| 10 | 4 | 0 | 1 | -3 |
|---|---|---|---|---|

false

| -3 | 0 | 0 | 1 | 10 |
|---|---|---|---|---|

true

# ejemplo

- escriba un método que recibe 2 listas de letras, cada una ordenada, y devuelve una lista que combina ambas respetando el orden
  - [ ‘a’, ‘e’, ‘i’, ‘o’, ‘u’ ]
  - [ ‘d’, ‘m’, ‘y’, ‘z’ ]
  - [ ‘a’, ‘d’, ‘e’, ‘i’, ‘m’, ‘o’, ‘u’, ‘y’, ‘z’ ]

# búsqueda

- int busca (int x, ArrayList<Integer> tabla)
  - devuelve i, tal que x == tabla[i]
  - devuelve –1, si no está
- búsqueda lineal
  - sobre cualquier lista