Programación de Computadores: Java - Collections Framework Lists - Maps - Sets

Juan F. Pérez

Departamento MACC
Matemáticas Aplicadas y Ciencias de la Computación
Universidad del Rosario

juanferna.perez@urosario.edu.co

Segundo Semestre de 2017

Contenidos

Collections Framework

Collections Framework

Clases para representar colecciones de elementos

Collections Framework

Clases para representar colecciones de elementos

Listas

Collections Framework

Clases para representar colecciones de elementos

Listas

Conjuntos

Collections Framework

Clases para representar colecciones de elementos

Listas

Conjuntos

Mapas

Listas (Lists)

Colección de elementos

Listas (Lists)

Colección de elementos

Indexable (`get(i)`, `set(i)`)

Listas (Lists)

Colección de elementos

Indexable (`get(i)`, `set(i)`)

`ArrayList`: representación como arreglo

Listas (Lists)

Colección de elementos

Indexable (`get(i)`, `set(i)`)

`ArrayList`: representación como arreglo

`https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`

Listas (Lists)

Colección de elementos

Indexable (`get(i)`, `set(i)`)

`ArrayList`: representación como arreglo

`https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`

`LinkedList`: representación como sucesión de elementos enlazados

Listas (Lists)

Colección de elementos

Indexable (`get(i)`, `set(i)`)

`ArrayList`: representación como arreglo

`https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`

`LinkedList`: representación como sucesión de elementos enlazados

`https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html`

Listas (Lists)

Colección de elementos

Indexable (`get(i)`, `set(i)`)

`ArrayList`: representación como arreglo

`https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`

`LinkedList`: representación como sucesión de elementos enlazados

`https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html`

Funcionalmente equivalentes

Conjuntos (Sets)

Colección de elementos

Conjuntos (Sets)

Colección de elementos

Elementos no se repiten

Conjuntos (Sets)

Colección de elementos

Elementos no se repiten

No es indexable

Conjuntos (Sets)

Colección de elementos

Elementos no se repiten

No es indexable

`HashSet`: representación como tabla de hashing

Conjuntos (Sets)

Colección de elementos

Elementos no se repiten

No es indexable

`HashSet`: representación como tabla de hashing

`https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html`

Conjuntos (Sets)

Colección de elementos

Elementos no se repiten

No es indexable

`HashSet`: representación como tabla de hashing

`https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html`

`TreeSet`: representación como árbol

Conjuntos (Sets)

Colección de elementos

Elementos no se repiten

No es indexable

`HashSet`: representación como tabla de hashing

`https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html`

`TreeSet`: representación como árbol

`https://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html`

Funcionalmente equivalentes

Mapas (Maps)

Mapa o diccionario

Mapas (Maps)

Mapa o diccionario

Una llave para cada valor

Mapas (Maps)

Mapa o diccionario

Una llave para cada valor

`HashMap`: representación como tabla de hashing

Mapas (Maps)

Mapa o diccionario

Una llave para cada valor

`HashMap`: representación como tabla de hashing

Mapas (Maps)

Mapa o diccionario

Una llave para cada valor

`HashMap`: representación como tabla de hashing

`https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html`

Mapas (Maps)

Mapa o diccionario

Una llave para cada valor

`HashMap`: representación como tabla de hashing

`https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html`

`TreeMap`: representación como árbol

Mapas (Maps)

Mapa o diccionario

Una llave para cada valor

`HashMap`: representación como tabla de hashing

`https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html`

`TreeMap`: representación como árbol

`https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html`

Funcionalmente equivalentes

Comportamiento vs. Representación

`ArrayList` vs `LinkedList`

Comportamiento vs. Representación

`ArrayList` vs `LinkedList`

`HashSet` vs `TreeSet`

Comportamiento vs. Representación

`ArrayList` vs `LinkedList`

`HashSet` vs `TreeSet`

`HashMap` vs `TreeMap`

Comportamiento vs. Representación

`ArrayList` vs `LinkedList`

`HashSet` vs `TreeSet`

`HashMap` vs `TreeMap`

Comportamiento: funcionalidad

Comportamiento vs. Representación

`ArrayList` vs `LinkedList`

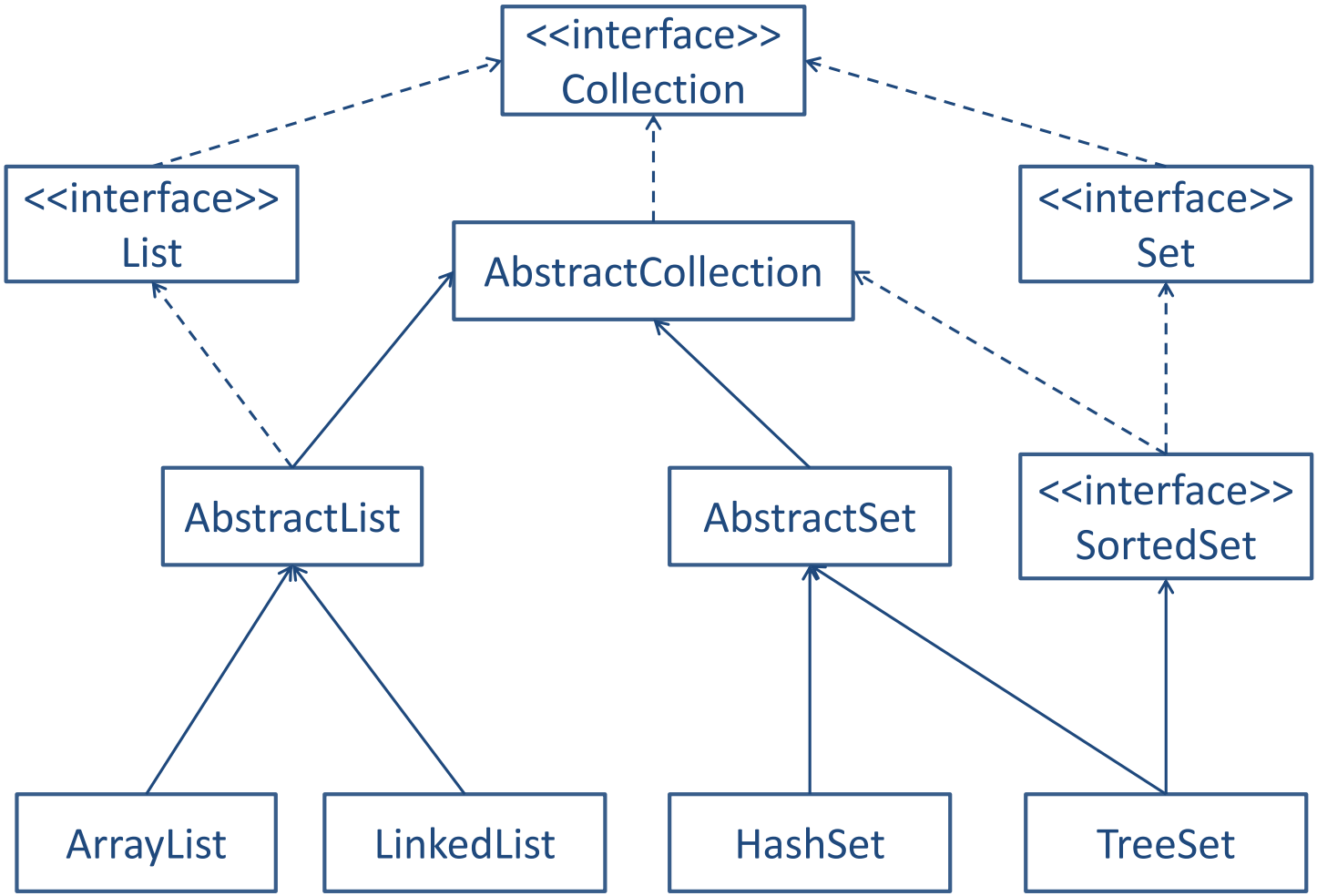
`HashSet` vs `TreeSet`

`HashMap` vs `TreeMap`

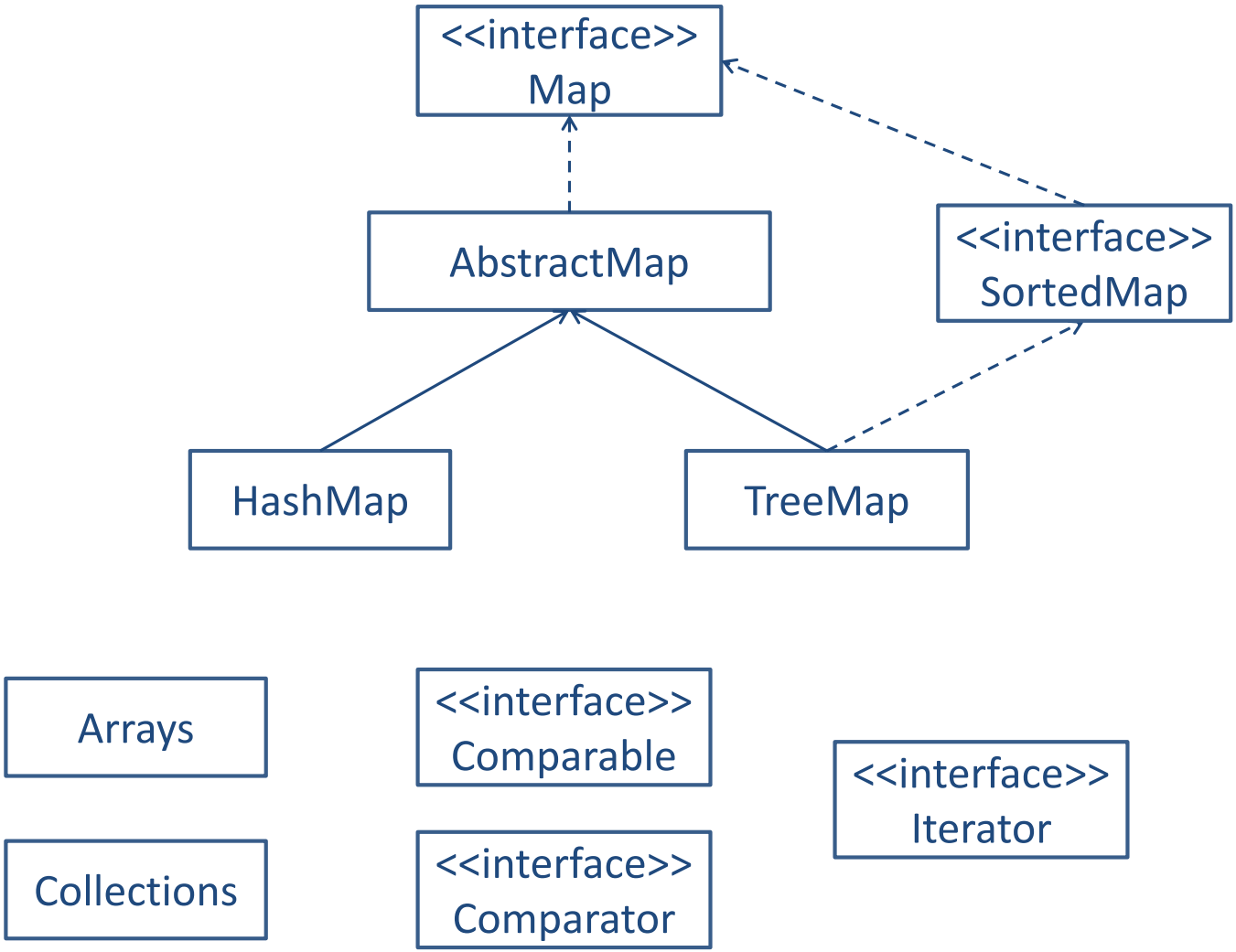
Comportamiento: funcionalidad

Representación: implementación

Collections Framework



Collections Framework



Interfaz Collection<E>

Métodos (que debe implementar toda clase que implemente esta interfaz):

```
boolean add(<E> elemento)
```

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaz Collection<E>

Métodos (que debe implementar toda clase que implemente esta interfaz):

```
boolean add(<E> elemento)
```

```
boolean remove(<E> elemento)
```

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaz Collection<E>

Métodos (que debe implementar toda clase que implemente esta interfaz):

```
boolean add(<E> elemento)
boolean remove(<E> elemento)
void clear()
```

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaz Collection<E>

Métodos (que debe implementar toda clase que implemente esta interfaz):

```
boolean add(<E> elemento)
boolean remove(<E> elemento)
void clear()
int size()
```

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaz Collection<E>

Métodos (que debe implementar toda clase que implemente esta interfaz):

```
boolean add(<E> elemento)
boolean remove(<E> elemento)
void clear()
int size()
boolean contains(<E> elemento)
```

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaz Collection<E>

Métodos (que debe implementar toda clase que implemente esta interfaz):

```
boolean add(<E> elemento)
boolean remove(<E> elemento)
void clear()
int size()
boolean contains(<E> elemento)
boolean isEmpty()
```

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaz Collection<E>

Métodos (que debe implementar toda clase que implemente esta interfaz):

```
boolean add(<E> elemento)
boolean remove(<E> elemento)
void clear()
int size()
boolean contains(<E> elemento)
boolean isEmpty()
Iterator iterator()
```

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaz List<E>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
void add(int index, E elemento)
```

https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Interfaz List<E>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
void add(int index, E elemento)
```

```
E get(int index)
```

https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Interfaz List<E>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
void add(int index, E elemento)
E get(int index)
E remove(int index)
```

https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Interfaz List<E>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
void add(int index, E elemento)
E get(int index)
E remove(int index)
int size()
```

https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Interfaz List<E>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
void add(int index, E elemento)
E get(int index)
E remove(int index)
int size()
E set(int index, E elemento)
```

https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Interfaz Set

Compare `List` con `Set`

Interfaz Set

Compare `List` con `Set`

`https://docs.oracle.com/javase/8/docs/api/java/util/Set.html`

Interfaz Set

Compare `List` con `Set`

`https://docs.oracle.com/javase/8/docs/api/java/util/Set.html`

No tiene métodos que se refieren a operaciones con índices

Iteradores

Iteradores

Objetivo: recorrer todos los elementos de una colección

Iteradores

Objetivo: recorrer todos los elementos de una colección

Ejemplo con Listas: clase `EjemploArrayList.java`

Iteradores

Objetivo: recorrer todos los elementos de una colección

Ejemplo con Listas: clase `EjemploArrayList.java`

Usando un índice:

```
private void recorrerListaIndice() {
 println("------------------");
 for (int i = 0; i < miLista.size(); i++){
  println(miLista.get(i));
 }
 println("------------------");
}
```

Iteradores

Objetivo: recorrer todos los elementos de una colección

Ejemplo con Listas: clase `EjemploArrayList.java`

Usando un índice:

```
private void recorrerListaIndice() {
 println("------------------");
 for (int i = 0; i < miLista.size(); i++){
  println(miLista.get(i));
 }
 println("------------------");
}
```

Iterador no requiere índices

Iteradores

Objetivo: recorrer todos los elementos de una colección

Ejemplo con Listas: clase `EjemploArrayList.java`

Usando un índice:

```
private void recorrerListaIndice() {
 println("------------------");
 for (int i = 0; i < miLista.size(); i++){
  println(miLista.get(i));
 }
 println("------------------");
}
```

Iterador no requiere índices

Métodos clave de un iterador:

Iteradores

Objetivo: recorrer todos los elementos de una colección

Ejemplo con Listas: clase `EjemploArrayList.java`

Usando un índice:

```
private void recorrerListaIndice() {
 println("------------------");
 for (int i = 0; i < miLista.size(); i++){
  println(miLista.get(i));
 }
 println("------------------");
}
```

Iterador no requiere índices

Métodos clave de un iterador:

```
boolean hasNext()
```

Iteradores

Objetivo: recorrer todos los elementos de una colección

Ejemplo con Listas: clase `EjemploArrayList.java`

Usando un índice:

```
private void recorrerListaIndice() {
 println("------------------");
 for (int i = 0; i < miLista.size(); i++){
  println(miLista.get(i));
 }
 println("------------------");
}
```

Iterador no requiere índices

Métodos clave de un iterador:

```
boolean hasNext()
```

```
E next()
```

Iteradores

Usando un iterador:

```
private void recorrerListaIterador() {
 println("------------------");
 Iterator<String> iter = miLista.iterator();
 while (iter.hasNext()){
  println(iter.next());
 }
 println("------------------");
}
```

Iteradores

Usando un iterador:

```
private void recorrerListaIterador() {
 println("------------------");
 Iterator<String> iter = miLista.iterator();
 while (iter.hasNext()){
  println(iter.next());
 }
 println("------------------");
}
```

Una versión alternativa del iterador:

```
private void recorrerListaIteradorV2() {
 println("------------------");
 for (String elem : miLista){
  println(elem);
 }
 println("------------------");
}
```

Iteradores

Una versión alternativa del iterador:

```
private void recorrerListaIteradorV2() {
 println("------------------");
 for (String elem : miLista){
  println(elem);
 }
 println("------------------");
}
```

Iteradores para Conjuntos - TreeSet

Clase `EjemploTreeSet.java`

Iteradores para Conjuntos - TreeSet

Clase `EjemploTreeSet.java`

Versión 1:

```
private void recorrerConjuntoIterador() {
 println("--------------------");
 Iterator<String> iter = miConjunto.iterator();
 while (iter.hasNext()){
  println(iter.next());
 }
 println("--------------------");
}
```

Versión 2:

```
private void recorrerConjuntoIteradorV2() {
 println("--------------------");
 for (String elem : miConjunto){
  println(elem);
 }
 println("--------------------");
}
```

Iteradores para Conjuntos - HashSet

Clase `EjemploHashSet.java`

Iteradores para Conjuntos - HashSet

Clase `EjemploHashSet.java`

Versión 1:

```
private void recorrerConjuntoIterador() {
 println("-------------------");
 Iterator<String> iter = miConjunto.iterator();
 while (iter.hasNext()){
  println(iter.next());
 }
 println("-------------------");
}
```

Versión 2:

```
private void recorrerConjuntoIteradorV2() {
 println("-------------------");
 for (String elem : miConjunto){
  println(elem);
 }
 println("-------------------");
}
```

TreeSet vs. HashSet

Ejecute `EjemploTreeSet.java`

TreeSet vs. HashSet

Ejecute `EjemploTreeSet.java`

Ejecute `EjemploHashSet.java`

TreeSet vs. HashSet

Ejecute `EjemploTreeSet.java`

Ejecute `EjemploHashSet.java`

Compare los resultados

TreeSet vs. HashSet

Ejecute `EjemploTreeSet.java`

Ejecute `EjemploHashSet.java`

Compare los resultados

`TreeSet` mantiene el conjunto ordenado

TreeSet vs. HashSet

Ejecute `EjemploTreeSet.java`

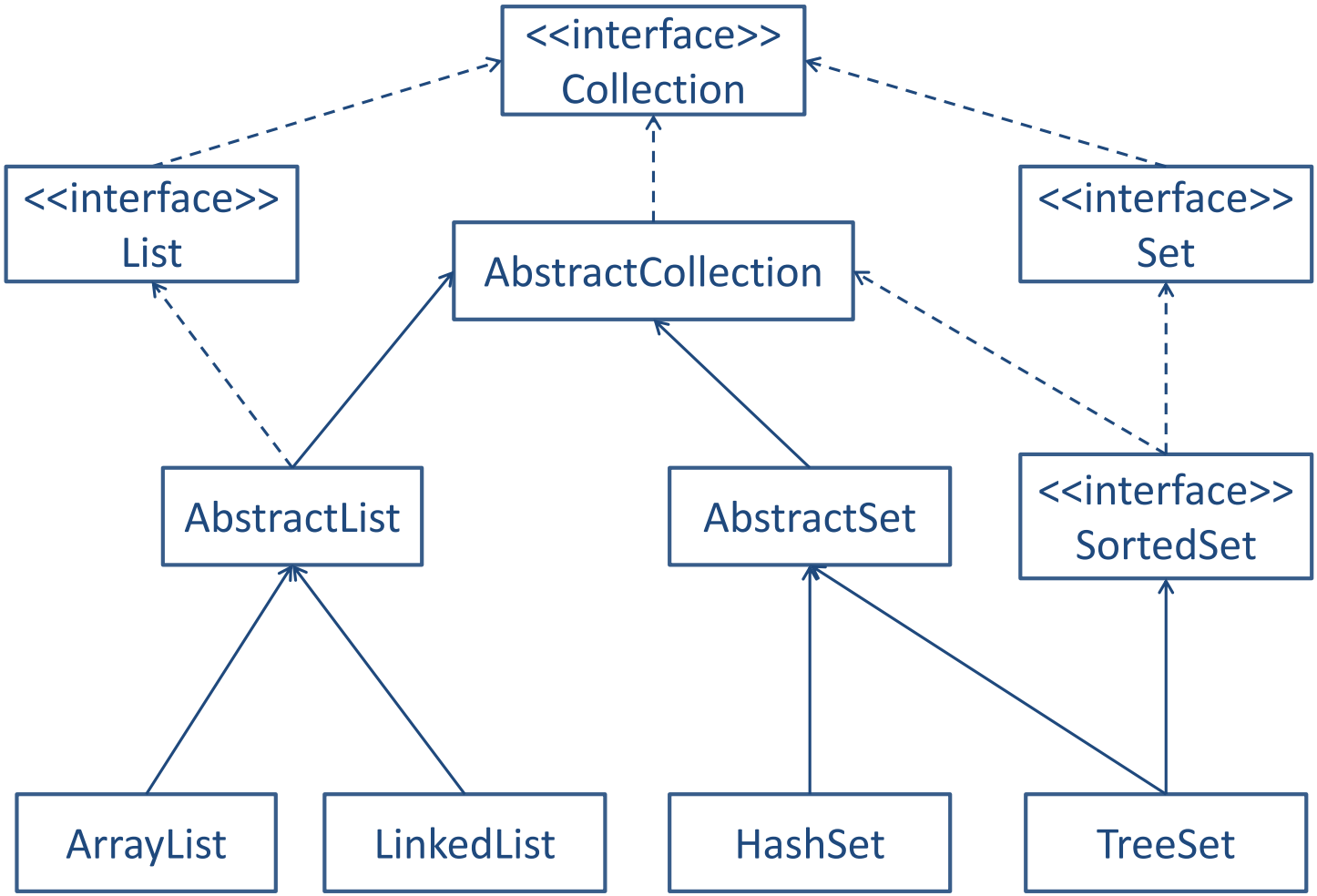
Ejecute `EjemploHashSet.java`

Compare los resultados

`TreeSet` mantiene el conjunto ordenado

`TreeSet` implementa interfaz `SortedSet`

Collections Framework



Comportamiento vs. Representación

Comportamiento vs. Representación

`ArrayList` y `LinkedList`

Comportamiento vs. Representación

`ArrayList` y `LinkedList`

`ArrayList`: representación como arreglo

Comportamiento vs. Representación

`ArrayList` y `LinkedList`

`ArrayList`: representación como arreglo

Agregar un nuevo elemento (`add`): $O(n)$

Comportamiento vs. Representación

`ArrayList` y `LinkedList`

`ArrayList`: representación como arreglo

- Agregar un nuevo elemento (`add`): $O(n)$
- Seleccionar un elemento en la lista (`get(int index)`): $O(1)$

Comportamiento vs. Representación

`ArrayList` y `LinkedList`

`ArrayList`: representación como arreglo

- Agregar un nuevo elemento (`add`): $O(n)$
- Seleccionar un elemento en la lista (`get(int index)`): $O(1)$

`LinkedList`: representación como sucesión de elementos enlazados

Comportamiento vs. Representación

`ArrayList` y `LinkedList`

`ArrayList`: representación como arreglo

- Agregar un nuevo elemento (`add`): $O(n)$
- Seleccionar un elemento en la lista (`get(int index)`): $O(1)$

`LinkedList`: representación como sucesión de elementos enlazados

- Agregar un nuevo elemento (`add`): $O(1)$

Comportamiento vs. Representación

`ArrayList` y `LinkedList`

`ArrayList`: representación como arreglo

- Agregar un nuevo elemento (`add`): $O(n)$
- Seleccionar un elemento en la lista (`get(int index)`): $O(1)$

`LinkedList`: representación como sucesión de elementos enlazados

- Agregar un nuevo elemento (`add`): $O(1)$
- Seleccionar un elemento en la lista (`get(int index)`): $O(n)$

Mapas

Interfaz Map<K,V>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
V get(Object key)
```

https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Interfaz Map<K,V>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
V get(Object key)
```

```
V put(K key, V value)
```

https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Interfaz Map<K,V>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
V get(Object key)
V put(K key, V value)
V remove(Object key)
```

https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Interfaz Map<K,V>

Algunos Métodos (que debe implementar toda clase que implemente esta interfaz):

```
V get(Object key)
V put(K key, V value)
V remove(Object key)
Set<K> keySet()
```

https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Ejemplo HashMap

Clase EjemploHashMap

```
private void agregarElementos() {
 for (int i = 0; i < codigos.length; i++){
  miMap.put(codigos[i], paises[i]);
 }
}
```

Ejemplo HashMap

Clase EjemploHashMap

```
private void imprimirAlgunosElementos() {
 String llave = "DZ";
 imprimirElemento(llave);

 llave = "AU";
 imprimirElemento(llave);
}
```

Ejemplo HashMap

Clase EjemploHashMap

```
private void imprimirTodosElementos() {
 println("----------------------------");
 for (int i = 0; i < codigos.length; i++){
  imprimirElemento(codigos[i]);
 }
 println("----------------------------");
}
```

Otro Ejemplo: `TreeMap`

Clase `EjemploTreeMap`

Igual a Clase `EjemploHashMap`

Otro Ejemplo: `TreeMap`

Clase `EjemploTreeMap`

- Igual a Clase `EjemploHashMap`
- `TreeMap` y `HashMap` ofrecen las mismas funcionalidades

Otro Ejemplo: `TreeMap`

Clase `EjemploTreeMap`

- Igual a Clase `EjemploHashMap`
- `TreeMap` y `HashMap` ofrecen las mismas funcionalidades
- Diferencia en la implementación: árbol vs. tabla hash

Otro Ejemplo: `TreeMap`

Clase `EjemploTreeMap`

- Igual a Clase `EjemploHashMap`
- `TreeMap` y `HashMap` ofrecen las mismas funcionalidades
- Diferencia en la implementación: árbol vs. tabla hash
- ¿Qué es una tabla hash?

Mapas y Tablas Hash

HashMap y Tabla Hash

La clase HashMap se implementa usando una tabla hash

HashMap y Tabla Hash

La clase HashMap se implementa usando una tabla hash

Objetivo: encontrar elementos en $O(1)$

HashMap y Tabla Hash

La clase HashMap se implementa usando una tabla hash

Objetivo: encontrar elementos en $O(1)$

Dificultad: buscar una llave toma $O(n)$ con búsqueda lineal y $O(\log n)$ con búsqueda binaria

HashMap y Tabla Hash

La clase HashMap se implementa usando una tabla hash

Objetivo: encontrar elementos en $O(1)$

Dificultad: buscar una llave toma $O(n)$ con búsqueda lineal y $O(\log n)$ con búsqueda binaria

Arreglos logran $O(1)$ para encontrar elementos gracias a los índices, pero es necesario saber el índice de la llave que se busca

HashMap y Tabla Hash

La clase HashMap se implementa usando una tabla hash

Objetivo: encontrar elementos en $O(1)$

Dificultad: buscar una llave toma $O(n)$ con búsqueda lineal y $O(\log n)$ con búsqueda binaria

Arreglos logran $O(1)$ para encontrar elementos gracias a los índices, pero es necesario saber el índice de la llave que se busca

Solución: usar la llave misma para (casi) generar el índice del objeto que se busca

Código Hash

Código Hash: representa/resume un objeto con un entero

Código Hash

Código Hash: representa/resume un objeto con un entero

Representación no es única: objetos diferentes pueden tener el mismo código Hash

Código Hash

Código Hash: representa/resume un objeto con un entero

Representación no es única: objetos diferentes pueden tener el mismo código Hash

Variedad de objetos puede ser más grande que variedad en enteros (32 bits)

Código Hash

Código Hash: representa/resume un objeto con un entero

Representación no es única: objetos diferentes pueden tener el mismo código Hash

Variedad de objetos puede ser más grande que variedad en enteros (32 bits)

Se requiere una fórmula que permita obtener el entero a partir del objeto

Código Hash

Código Hash: representa/resume un objeto con un entero

Representación no es única: objetos diferentes pueden tener el mismo código Hash

Variedad de objetos puede ser más grande que variedad en enteros (32 bits)

Se requiere una fórmula que permita obtener el entero a partir del objeto

Ejemplo: para un String s de longitud n, su código Hash en Java se calcula como

$$\sum_{i=0}^{n-1} s[i] * 31^{n-1-i}$$

donde $s[i]$ es el i-ésimo caracter del string s

Código Hash

Código Hash: representa/resume un objeto con un entero

Representación no es única: objetos diferentes pueden tener el mismo código Hash

Variedad de objetos puede ser más grande que variedad en enteros (32 bits)

Se requiere una fórmula que permita obtener el entero a partir del objeto

Ejemplo: para un String s de longitud n, su código Hash en Java se calcula como

$$\sum_{i=0}^{n-1} s[i] * 31^{n-1-i}$$

donde $s[i]$ es el i-ésimo caracter del string s

`https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#hashCode()`

Código Hash

Si código Hash es único para cada objeto en un conjunto: hash perfecto

Código Hash

Si código Hash es único para cada objeto en un conjunto: hash perfecto

Código Hash debe ser consistente con método `equals()`

Código Hash

Si código Hash es único para cada objeto en un conjunto: hash perfecto

Código Hash debe ser consistente con método `equals()`

Si dos objetos son iguales (`equals()`$= 0$) deben tener el mismo código Hash

Tabla Hash

Almacenar objetos en buckets

Tabla Hash

Almacenar objetos en buckets

Usar el código hash de la llave para determinar en qué bucket va el objeto

Tabla Hash

Almacenar objetos en buckets

Usar el código hash de la llave para determinar en qué bucket va el objeto

Puede haber menos buckets que objetos

Tabla Hash

Almacenar objetos en buckets

Usar el código hash de la llave para determinar en qué bucket va el objeto

Puede haber menos buckets que objetos

Objetos diferentes pueden tener el mismo código Hash

Tabla Hash

Almacenar objetos en buckets

Usar el código hash de la llave para determinar en qué bucket va el objeto

Puede haber menos buckets que objetos

Objetos diferentes pueden tener el mismo código Hash

Varios objetos pueden ir en el mismo bucket

Tabla Hash

Almacenar objetos en buckets

Usar el código hash de la llave para determinar en qué bucket va el objeto

Puede haber menos buckets que objetos

Objetos diferentes pueden tener el mismo código Hash

Varios objetos pueden ir en el mismo bucket

Necesario buscar en cada bucket (busqueda lineal, como en lista enlazada)

Tabla Hash

Muchos buckets: mucho espacio en memoria

Tabla Hash

Muchos buckets: mucho espacio en memoria

Pocos buckets: muchos objetos por bucket, muchas búsquedas lineales

Tabla Hash

Muchos buckets: mucho espacio en memoria

Pocos buckets: muchos objetos por bucket, muchas búsquedas lineales

Ejemplo: `SimpleStringHashMap.java`

Tabla Hash Ejemplo

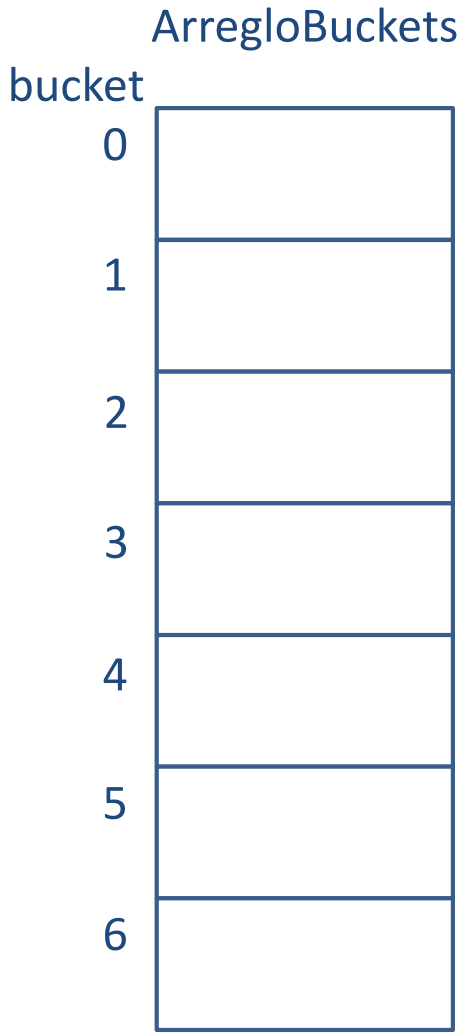


Tabla Hash Ejemplo

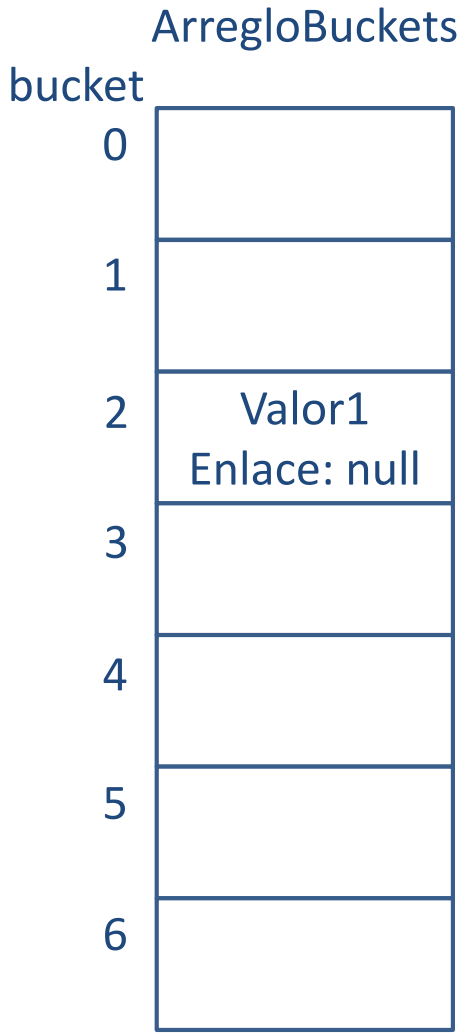


Tabla Hash Ejemplo

Tabla Hash Ejemplo

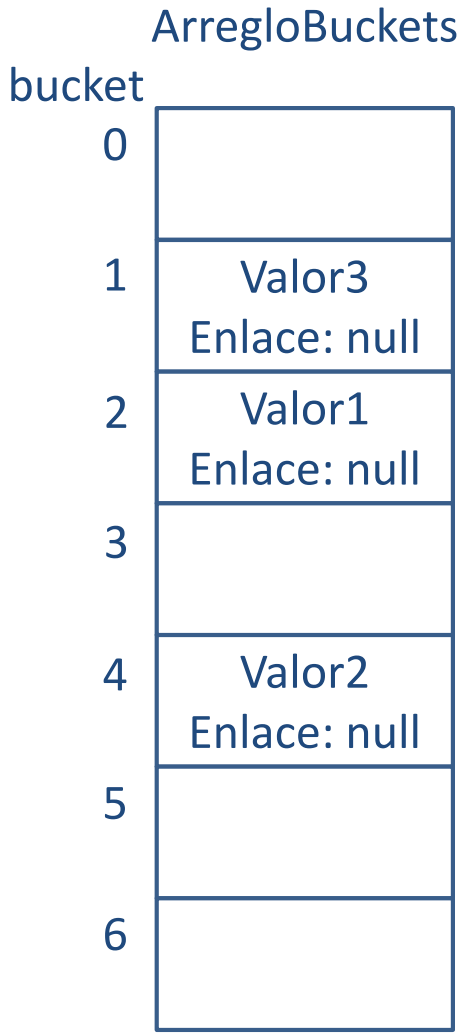


Tabla Hash Ejemplo

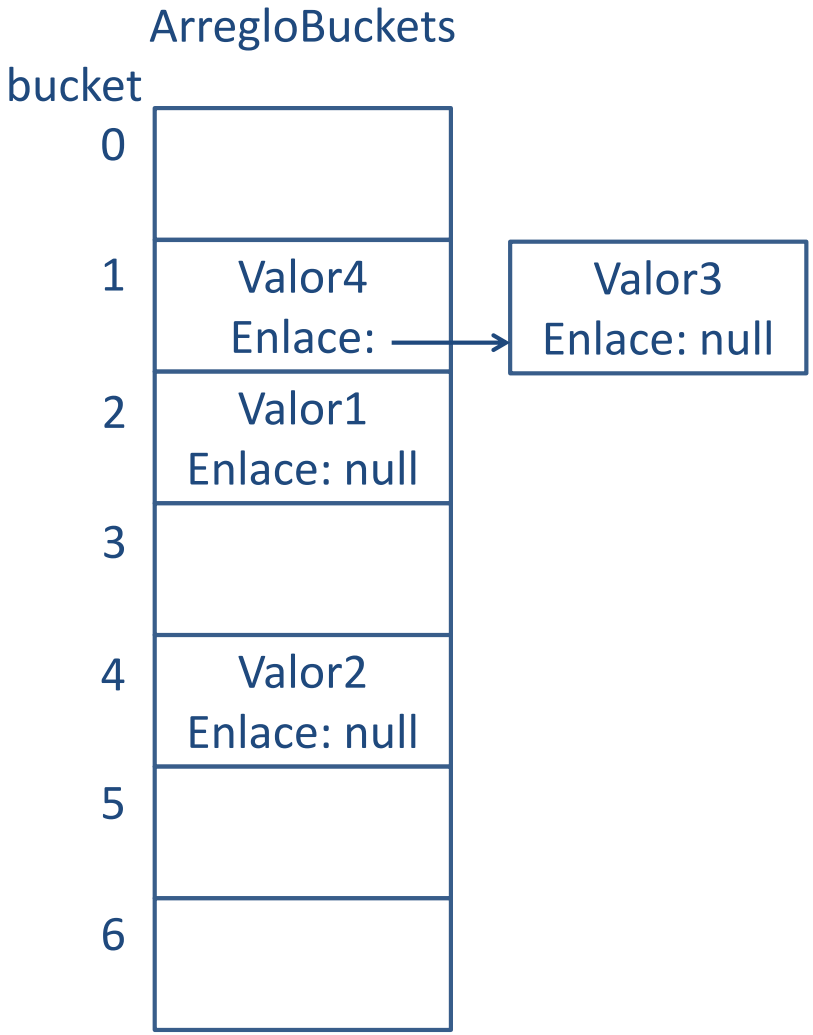


Tabla Hash Ejemplo

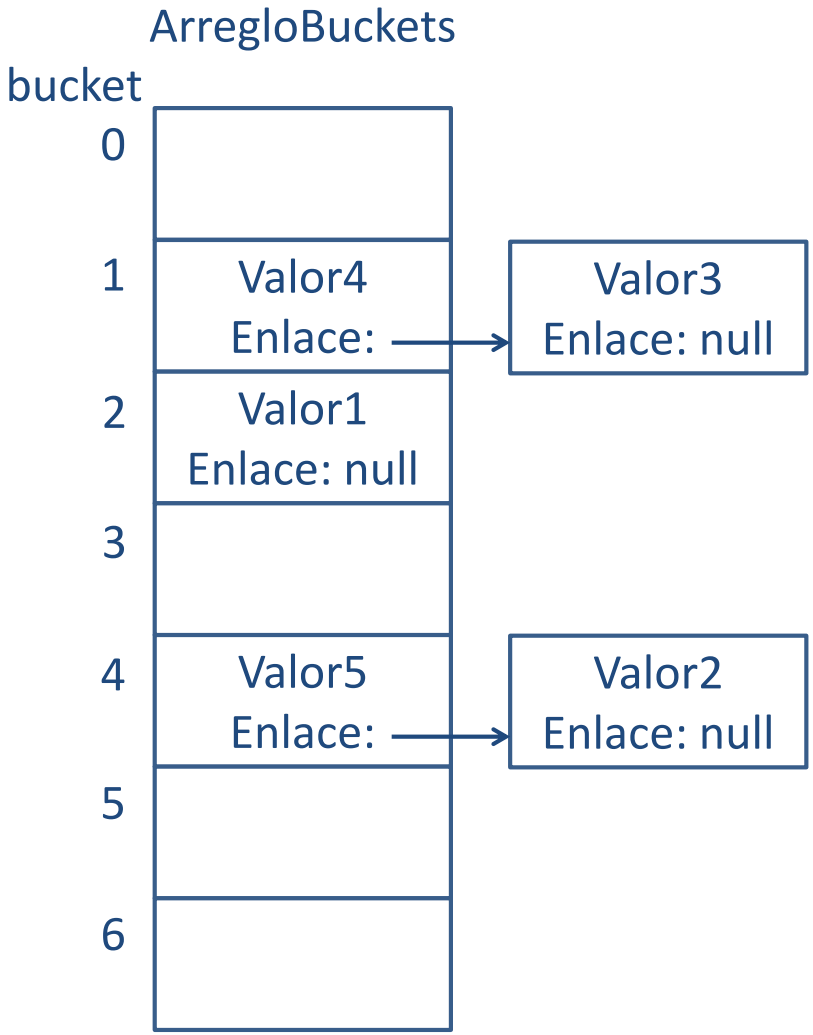


Tabla Hash Ejemplo

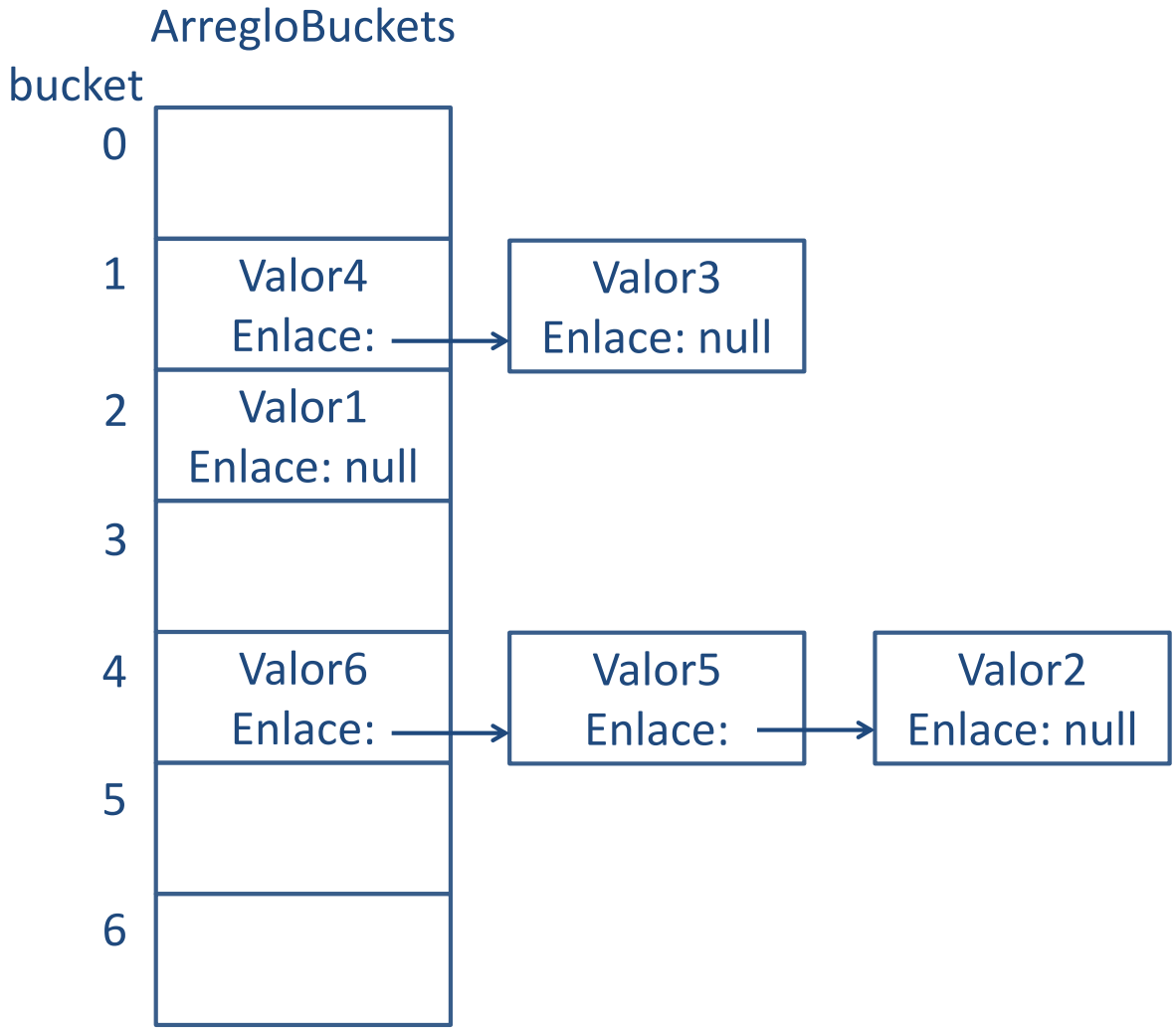


Tabla Hash Ejemplo