

Unidad 9

Estructuras dinámicas de datos

Programación
1º D.A.M.

1

Contenido

1. Estructuras estáticas vs. dinámicas
2. Interfaces Collection e Iterator
3. Listas enlazadas
4. Árboles
5. Mapas
6. Vectores
7. Pilas
8. Colas

2

1. Estructuras estáticas vs. dinámicas

3

1. Estructuras estáticas vs. dinámicas

■ Estructuras de almacenamiento de colecciones

■ Estáticas

- Tamaño conocido en tiempo de compilación
- Ejemplo
 - Arrays

■ Dinámicas

- Tamaño variable en tiempo de ejecución
- Tamaño ilimitado
- Ejemplos
 - Pilas, colas, listas enlazadas, árboles, grafos, ...
- Implementación
 - C → Punteros
 - Java → Clases especiales

4

2. Interfaces Collection e Iterator

1. Interfaz Collection

2. Interfaz Iterator

5

2.1. Interfaz Collection

- `java.util.Collection`
- Interfaz fundamental para estructuras dinámicas
- Define métodos interesantes para trabajar con listas
 - `boolean add(Object o)`
 - `boolean remove(Object o)`
 - `int size()`
 - `boolean isEmpty()`
 - `boolean contains(Object o)`
 - `void clear()`
 - `boolean addAll(Collection otra)`
 - `boolean removeAll(Collection otra)`
 - `boolean retainAll(Collection otra)`
 - `boolean containsAll(Collection otra)`
 - `Object[] toArray()`
 - `Iterator iterator()`

6

2.2. Interfaz Iterator

- `java.util.Iterator`
- Creación de objetos para recorrer elementos de una colección
- Métodos definidos
 - `Object next()`
 - `boolean hasNext()`
 - `void remove()`

7

3. Listas enlazadas

1. Descripción de lista enlazada
2. Interfaz List
3. Interfaz ListIterator
4. Interfaz Set
5. Clase ArrayList
6. Clase LinkedList
7. Clase HashSet

8

3.1. Descripción de lista enlazada

- Colección en que importa posición de los objetos
- Interfaces interesantes
 - `List`
 - `ListIterator`
 - `Set`
- Clases interesantes
 - `ArrayList`
 - `LinkedList`
 - `HashSet`

9

3.2. Interfaz List

- `java.util.List`
- Definición de listas enlazadas
- Deriva de `Collection`
 - Hereda todos sus métodos
- Métodos nuevos aportados
 - `void add(int indice, Object elemento)`
 - `void remove(int indice)`
 - `Object set(int indice, Object elemento)`
 - `int indexOf(Object elemento)`
 - `int lastIndexOf(Object elemento)`
 - `void addAll(int indice, Collection colección)`
 - `ListIterator listIterator()`
 - `ListIterator listIterator(int indice)`

10

3.3. Interfaz ListIterator

- Define clases de objetos para recorrer listas
- Hereda de la interfaz Iterator
- Métodos aportados
 - `void add(Object elemento)`
 - `void set(Object elemento)`
 - `Object previous()`
 - `boolean hasPrevious()`
 - `int nextIndex()`
 - `int previousIndex()`
 - `List subList(int desde, int hasta)`

11

3.4. Interfaz Set

- Define métodos para crear listas de elementos sin duplicados
- Deriva de la interfaz Collection
 - Mismos métodos
 - Diferencia en el uso de duplicados
 - Método `equals` redefinido

12

3.5. Clase ArrayList

- Implementa la interfaz List
- Crea listas en que aumenta el final frecuentemente
- Constructores
 - `ArrayList()`
 - `ArrayList(int capacidadInicial)`
 - `ArrayList(Collection c)`

13

3.6. Clase LinkedList

- Implementa la interfaz List
- Crea listas de adición doble
 - Desde el principio y el final
- Simplifica implantación de pilas y colas
- Métodos nuevos incorporados
 - `Object getFirst()`
 - `Object getLast()`
 - `void addFirst(Object o)`
 - `void addLast(Object o)`
 - `void removeFirst()`
 - `void removeLast()`

14

3.7. Clase HashSet

- Implementa la interfaz Set

15

4. Árboles

1. Descripción de árbol
2. Interfaz SortedSet
3. Clase TreeSet

16

4.1. Descripción de árbol

- Colección ordenada de elementos
- Al recorrerlo, los elementos aparecen en el orden correcto
- Adición de elementos más lenta
- Recorrido ordenado de elementos más eficiente

17

4.2. Interfaz SortedSet

- Define la estructura árbol
- Deriva de la interfaz `Collection`
- Nuevos métodos incorporados
 - `Object first()`
 - `Object last()`
 - `SortedSet headSet(Object o)` (`<`)
 - `SortedSet tailSet(Object o)` (`>=`)
 - `SortedSet subSet(Object menor, Object mayor)`
 - `Comparator comparator()`

18

4.3. Clase TreeSet

- Implementa la interfaz `SortedSet`
- Los objetos incluidos deben ser **comparables**
 - Para determinar su orden en el árbol
 - Posibilidades
 - Implementar interfaz `Comparable`
 - Incluye método `compareTo`
 - Un argumento `Object`
 - Implementar interfaz `Comparator`
 - Incluye método `compare`
 - Dos argumentos `Object`

19

5. Mapas

1. Descripción de mapa
2. Interfaz Map
3. HashMap, SortedMap y TreeMap

20

5.1. Descripción de mapa

- Colección de elementos clave – valor
- Localiza valor a partir de su clave
- Útiles y rápidos

21

5.2. Interfaz Map

- Define la estructura de mapa
- Métodos
 - `Object get(Object clave)`
 - `Object put(Object clave, Object valor)`
 - `Object remove(Object clave)`
 - `boolean containsKey(Object clave)`
 - `boolean containsValue(Object valor)`
 - `void putAll(Map mapa)`
 - `Set keySet()`
 - `Collection values()`
 - `Set entrySet()`

22

5.2. Interfaz Map

- Objeto interno `Map.Entry`
 - Representa un objeto pareja clave – valor
 - Métodos
 - `Object getKey()`
 - `Object getValue()`
 - `Object setValue(Object valor)`

23

5.3. HashMap, SortedMap y TreeMap

- `HashMap`
 - Implementa la interfaz `Map`
- `SortedMap`
 - Interfaz para mapa ordenado por claves
- `TreeMap`
 - Clase que implementa `SortedMap`

24

6. Vectores

1. Descripción de vector
2. Clase Vector

25

6.1. Descripción de vector

- Array dinámico
- Incluye cualquier tipo de objeto
- Tamaño variable en ejecución

26

6.2. Clase Vector

- Implementa la interfaz `List`
- Casi igual que `ArrayList`
- De las primeras usadas para estructuras dinámicas
- Uso actualmente desaconsejado
- Implementa métodos con `synchronized`
 - Posibilidad de uso con varios threads
 - Uso recomendado sólo en dicho caso
 - Ejecución más lenta
 - Por eso se recomienda mejor usar `ArrayList`
- Permite usar la interfaz `Enumeration`
 - `hashMoreElements`
 - `nextElement`

27

7. Pilas

1. Descripción de pila
2. Clase Stack

28

7.1. Descripción de pila

- Estructura dinámica de datos
- Colección de elementos LIFO
 - *Last In First Out*
 - Último en entrar es el primero en salir

29

7.2. Clase Stack

- Implementa la estructura pila
- Derivada de `Vector`
- Métodos
 - `Object push(Object elemento)`
 - `Object pop()`
 - `Object peek()`

30

8. Colas

1. Descripción de cola
2. Interfaz Queue

31

8.1. Descripción de cola

- Estructura dinámica de datos
- Colección de elementos FIFO
 - *First In First Out*
 - Primero en entrar es el primero en salir

32

8.2. Interfaz Queue

- Define la estructura cola
- Deriva de la interfaz Collection
- Métodos
 - `boolean add(Object elemento)`
 - `boolean offer(Object elemento)`
 - `Object element()`
 - `Object peek()`
 - `Object poll()`
 - `Object remove()`

33

Unidad 9

Estructuras dinámicas de datos

Programación
1º D.A.M.

34