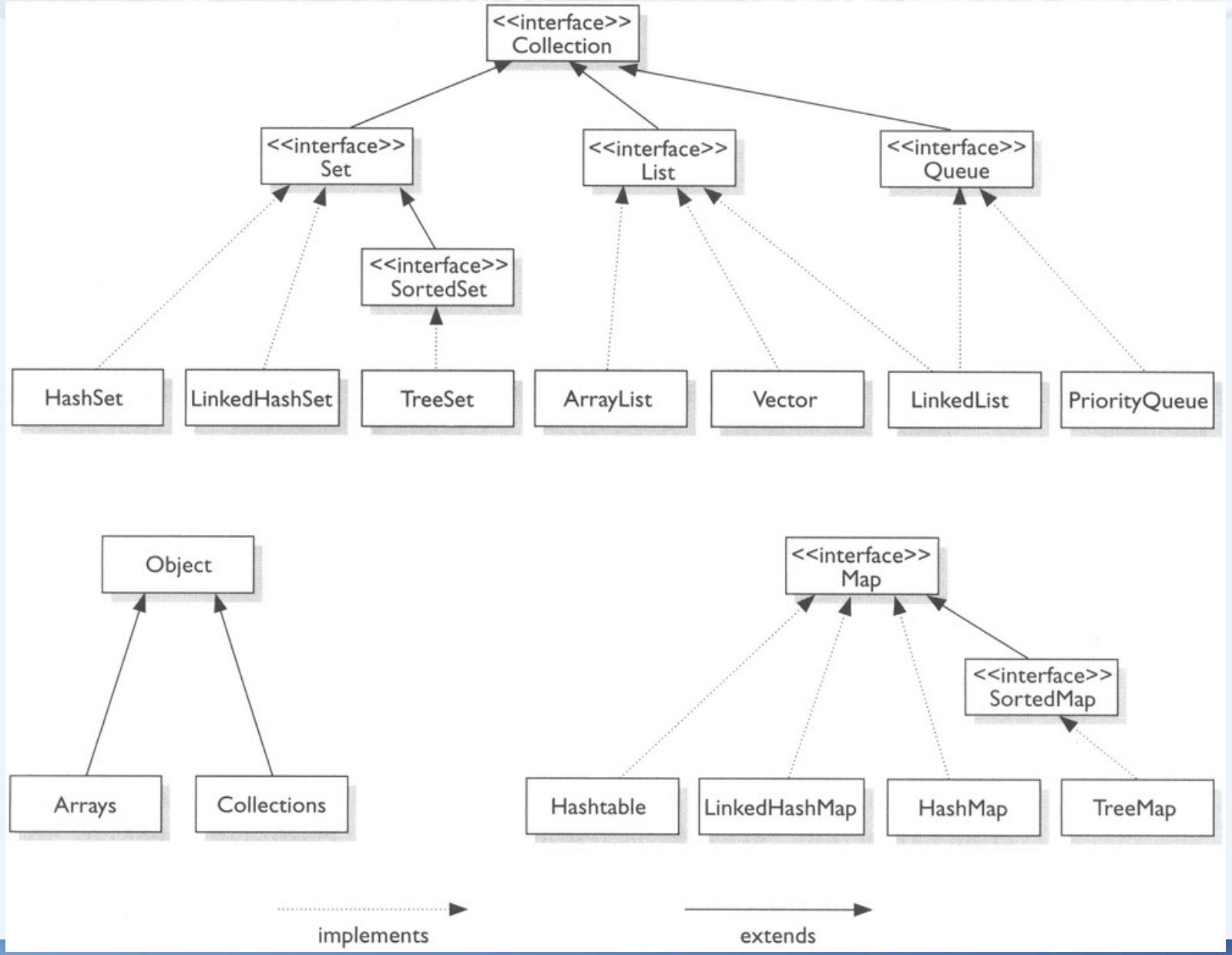
Collecions Framework



Collections (1/3)

- Jerarquia de classes i interfícies que permet treballar amb diferents implementacions per tal de facilitar la gestió de conjunts d'elements
  - Accés per posició:
    - Llistes(List), conjunts(Set), vectors (Vector)
  - Accés per clau:
    - Diccionaris(Map), arbres(TreeSet, TreeMap)
- Separen els interfícies de la implementació
  - Permet canviar de implementació sense modificar el codi
- Es troben a java.util
- Utilitzen els genèrics per donar més flexibilitat
- Es poden recórrer amb Iterator
- A partir de Java8 es pot fer servir forEach i expressions Lambda
- Per ordenar elements cal que siguin Comparable o implementar un Comparator

Collections (2/3)

- Jerarquia de classes i interfícies que permet treballar amb diferents implementacions per tal de facilitar la gestió de conjunts d'elements
  - Accés per posició:
    - Llistes(List), conjunts(Set), vectors (Vector)
  - Accés per clau:
    - Diccionaris(Map), arbres(TreeSet, TreeMap)
- Separen els interfícies de la implementació
  - Permet canviar de implementació sense modificar el codi
- Es troben a java.util
- Utilitzen els genèrics per donar més flexibilitat
- Es poden recórrer amb Iterator
- A partir de Java8 es pot fer servir forEach i expressions Lambda
- Per ordenar elements cal que siguin Comparable o implementar un Comparator

Collections (3/3)

- Algunes operacions no es troben implementades en totes les classes
  - Llencen UnsupportedOperationException
- Al constructor es pot passar un altra Collection
  - Copia tots els elements en la nova Collection

Collection i Map

- **Collection**: Conjunt d'elements
  - List
    - Els elements es mantenen en ordre d'inserció
    - Pot haver repetits
    - Podem fer servir ListIterator per iterar en tots dos sentits
  - Set
    - L'ordre depèn de la implementació. HashSet no en garanteix cap.
    - No pot haver elements repetits (tampoc més d'un null)
  - Queue
    - Cua on els elements es van afegint al final
- **Map**: Desa parells de clau-valor, permet accés per clau
  - No pot tenir claus duplicades
  - No garanteix l'ordre
  - SortedMap: Manté les claus ordenades

Implementacions de Collection

- **LinkedList**: Llista doblement enllaçada
  - Eficient per afegir i eliminar: O(1)
  - Lent per accedir de forma aleatòria a un element.
  - Es pot fer servir com a pila, cua o cua doble
- **ArrayList**: Llista sobre un vector ampliable
  - Lent per afegir i eliminar: O(n)
  - Eficient per accedir de forma aleatòria.
- **HashSet**: Utilitza una taula de Hash
  - Bona opció general. Afegir, eliminar i cercar no depenen de la mida: O(1)
- **TreeSet**: Arbre binari equilibrat.
  - Manté els elements ordenats
  - Requereix que els element siguin Comparable (o un Comparator en el constructor)
  - Més Lent per afegir eliminar o cercar que HashSet: O(log(n))
- **LinkedHashSet**: Set sobre una llista doblement enllaçada (com LinkedList)
  - Manté l'ordre d'inserció

Mètodes de Collection (i Set) 1/2

- boolean **add**(Object elem)
- boolean **remove**(Object elem)
- boolean **contains**(Object elem)
- void **clear**()
- int **size**()
- boolean **isEmpty**()
- Iterator **iterator**()
- Object[] **toArray**(), Object[] toArray(Object dest[])

Mètodes de Collection (i Set) 2/2

- boolean **containsAll** (Collection c)
- boolean **addAll** (Collection c)
- boolean **removeAll** (Collection c)
- boolean **retainAll** (Collection c)

Mètodes de List

- void **add**(int index, Object element)
- Object **remove**(int index)
- Object **get**(int index)
- Object **set**(int index, Object element)
- int **indexOf**(Object o)
- int **lastIndexOf**(Object o)
- List **subList**(int min, int max)
- ListIterator **listIterator**()
- List **subList** (int from, int to)

Mètodes de LinkedList

- boolean **add**(int index, E element)
- boolean **addLast**()
- Object **getFirst**()
- Object **getLast**()
- Object **removeFirst**()
- Object **removeLast**()
- boolean **removeFirstOccurrence**(Object o)
- boolean **removeLastOccurrence**(Object o)

Mètodes de SortedSet

- Object **first**()
- Object **last**()
- SortedSet **subSet**(Object fromEle, Object toEle)
- SortedSet **headSet**(Object toElement)
- SortedSet **tailSet**(Object fromElement)

Map (o diccionari)

- Format per Map.Entry
  - Parella de clau i valor
  - Permet accedir per clau
- No pot haver claus duplicades
- Permet accedir a claus, valors o tots dos
  - Set KeySet() → Claus
  - Collection values() → valors (pot haver duplicats)
  - Set EntrySet() → Parell de clau i valor (Map.Entry)

Mètodes de Map

- Object **put**(Object key, Object value);
- Object **remove**(Object key);
- Object **get**(Object key);
- containsKey,
- containsValue,
- isEmpty,
- size
- keySet
- values
- entrySet

Mètodes de SortedMap

- Object **firstKey**()
- Object **lastKey**()
- SortedMap **subMap**(Object minKey, Object maxKey),
- SortedMap **headMap**(Object maxKey),
- SortedMap **tailMap**(Object minKey)

Implementacions de Map

- HashMap: Basada en taula hash
  - Temps constant de modificació, cerca i inserció
- TreeMap: Basada en arbre binari equilibrat
  - Manté els elements ordenats per clau
  - Cal que siguin elements Comparable
  - Més lent que HashMap quan l'ordre no és important

Iterator

- Permeten recorrer una col·lecció
- Diferents implementacions
  - Iterator
    - boolean hasNext();
    - Object next();
    - void remove();
  - ListIterator: En llistes ordenades
    - Permet iterar endavant i endarrera

Excepcions

- UnsupportedOperationException
  - La implementació no suporta aquesta operació.
- ClassCastException
  - Es vol inserir un element no adient
- IllegalArgumentException
  - Es vol inserir una dada no adient
- NoSuchElementException
  - Es demana un element en una col·lecció buida
- NullPointerException
  - Objecte sense referència (null)

Mètodes útils de Collection

- public static Object **min**(Collection col)
- public static Object **max**(Collection col)
- public static Object **min**(Collection col, Comparator comp)
- public static Object **max**(Collection col, Comparator comp)
- public static void **reverse**(List lista)
- public static void **copy**(List dst, List fnt)
- public static void **sort**(List lista)
- public static void **sort**(List lista, Comparator comp)
- public static int **binarySearch**(List lista, Object clave)
- public static int **binarySearch**(List lista, Object clave, Comparator comp)

Quin Escollir?

- Cal elements únics → Set
- Cal mantenir ordre inserció → List, LinkedHashSet
- Parella de clau-valor → Map
  - Ordre ascendent de claus → TreeMap
  - Ordre d'inserciò → LinkedHashSet
  - No importa l'ordre → HashSet
- Moltes cerques per índex → ArrayList
- Moltes cerques per clau → HashSet

Ordre de les implementacions

- HashSet → No definit
- HashMap → No definit
- ArrayList → Ordre d'inserció
- LinkedList → Ordre d'inserció
- LinkedHashSet → Ordre d'inserció
- LinkedHashMap
  - Ordre de inserció de les claus (per defecte)
  - Ordre d'accés LRU (constructor especial)
- TreeSet → Ordre ascendent (Comparable/Comparator)
- TreeMap → ascendent de claus (Comparable/Comparator)