Colecciones de Objetos

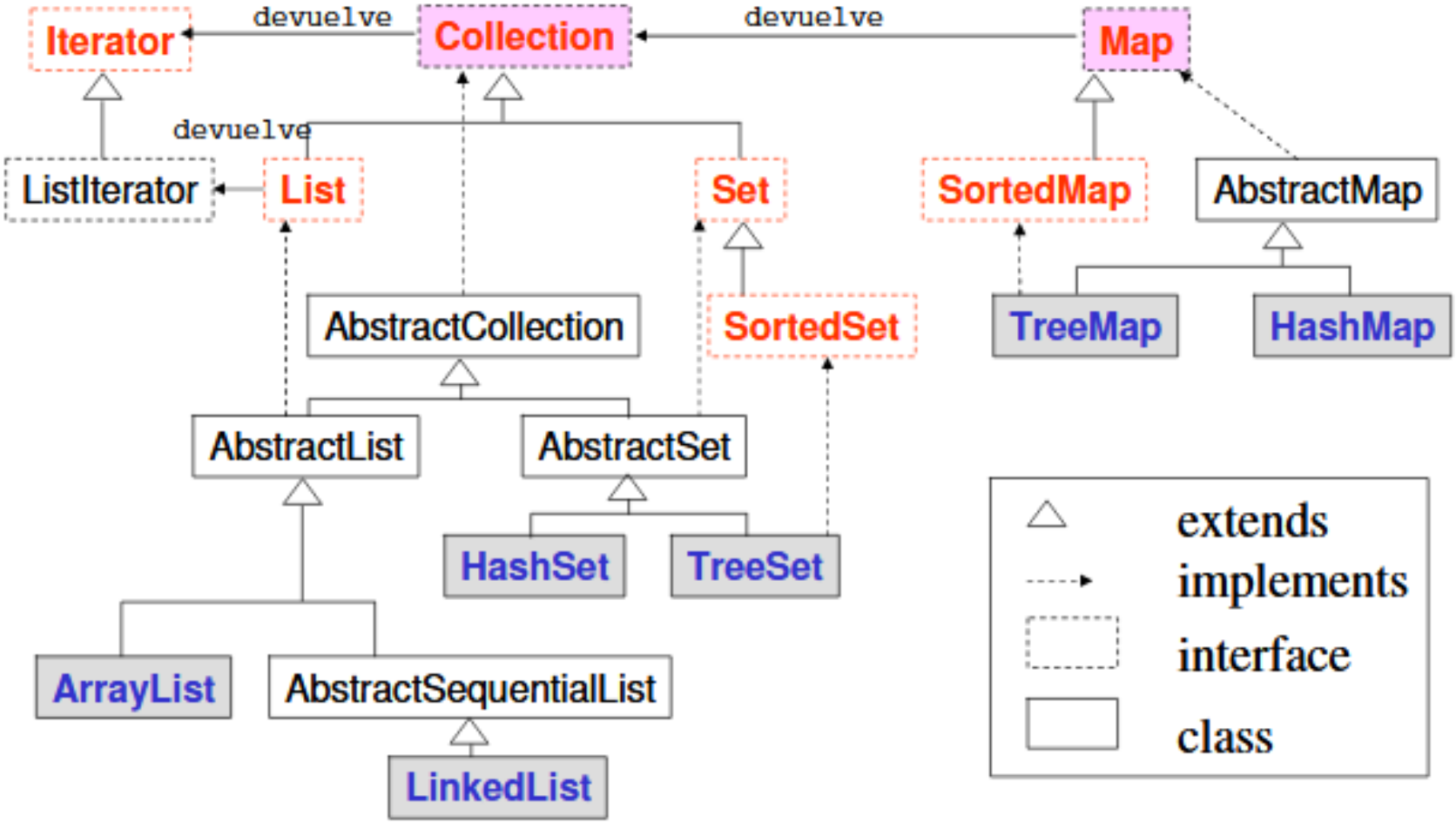
Introducción a las colecciones

- **Colección**: grupo de objetos almacenados de forma conjunta en una misma estructura. A cada objeto se le denomina elemento.

- Interfaz **Collection**:
  - Es genérica y permite almacenar cualquier tipo de objeto.
  - Define métodos comunes: añadir, eliminar, buscar, …

- Tipos de colecciones:
  - Listas y conjuntos (interfaces List y Set, que heredan de Collection).
  - Mapas (interfaz Map, que no hereda de Collection, pero la usa)

Clases e interfaces en Java



Interfaz Set

- Define una colección que no puede contener elementos duplicados.
- Para comprobar la duplicidad es necesario que la clase a la que pertenecen los objetos que contiene implementen correctamente los métodos equals y hashCode.
- Clases que implementan la interfaz Set:
  - **HashSet**: almacena los elementos en una tabla hash, por lo que es la que mejor rendimiento proporciona. Si el código hash de dos objetos es igual, compara mediante el método equals. No almacena de forma ordenada.
  - **LinkedHashSet**: utiliza tablas hash y listas enlazadas. Permite recuperar los elementos en orden de inserción.
  - **TreeSet**: utiliza árboles binarios rojo-negro para almacenar los objetos ordenados según sus valores. Es bastante más lento que HashSet. La clase a la que pertenecen los objetos que contiene debe implementar la interfaz Comparable, lo que implica sobrescribir el método compareTo.

Interfaz List

- Define una colección que puede contener elementos duplicados y están dispuestos en cierto orden.
- Añade métodos para acceder a los elementos por posición, para realizar consultas de elementos y para realizar operaciones sobre subconjunto de elementos.
- Clases que implementan la interfaz List:
  - **ArrayList**: se basa en un vector redimensionable. Es rápida en el acceso por posición, pero la inserción y el borrado implican muchas operaciones.
  - **LinkedList**: se basa en una lista doblemente enlazada y mejora el rendimiento con respecto a la anterior.

Recorrido de colecciones Set y List

- Estructura for-each: una variable va tomando los valores de los distintos elementos de la colección

```
for (Integer dato : coleccion) {
        System.out.println("Elemento almacenado: " + dato);
}
```

- Interfaz Iterator: determina métodos para recorrer una colección:
  - boolean hasNext(): devuelve verdadero si quedan más elementos para recorrer en la colección o falso en caso contrario.
  - E next(): devuelve el siguiente elemento de la colección. Si no hay siguiente lanza la excepción NoSuchElementException.
  - remove(): elimina de la colección el último elemento devuelto en la última llamada a next. Si no ha sido llamado lanza una excepción.

- Interfaz ListIterator: hereda de Iterator y añade nuevos métodos para modificar y recorrer listas en ambos sentidos.

Interfaz Map

- Los mapas son estructuras que almacenan parejas de clave-valor. No pueden contener claves duplicadas y cada clave solo puede tener asociado un valor. Tampoco se permite la inserción de objetos nulos.

- Actualmente, tanto la clave como el valor viene determinado por un tipo genérico, anteriormente se usaba la clase Object para aportar esa generalidad.

- Clases que implementan la interfaz Map:
  - **HashMap**: almacena las clave en una tabla hash. Tiene el mejor rendimiento aunque no garantiza ningún orden.
  - **TreeMap**: almacena las claves por orden según sus valores. Es bastante más lento que HashMap. La clase a la que pertenecen los objetos que contiene la clave debe implementar la interfaz Comparable, lo que implica sobrescribir el método compareTo. Aunque también es posible utilizar una clase Comparator en el constructor de la colección.
  - **LinkedHashMap**: almacena las claves por orden de inserción. Es un poco más lento que HashMap.

Recorrido de Map

- Los mapas no derivan de Collection y no se pueden recorrer con un iterador.

- En cambio contienen unos métodos que facilitan el recorrido de todos los elementos:
  - **entrySet**: devuelve un conjunto con las parejas o entradas (clase Entry) del mapa. Se puede acceder a su clave con getKey() y a su valor con getValue().
  - **keySet**: devuelve un conjunto con las claves del mapa. Se puede acceder al valor asociado de una clave con:
    - get(E clave): devuelve el valor del mapa para la clave que se pasa como parámetro.

Uso de colecciones

Diagrama de decisión para uso de colecciones Java