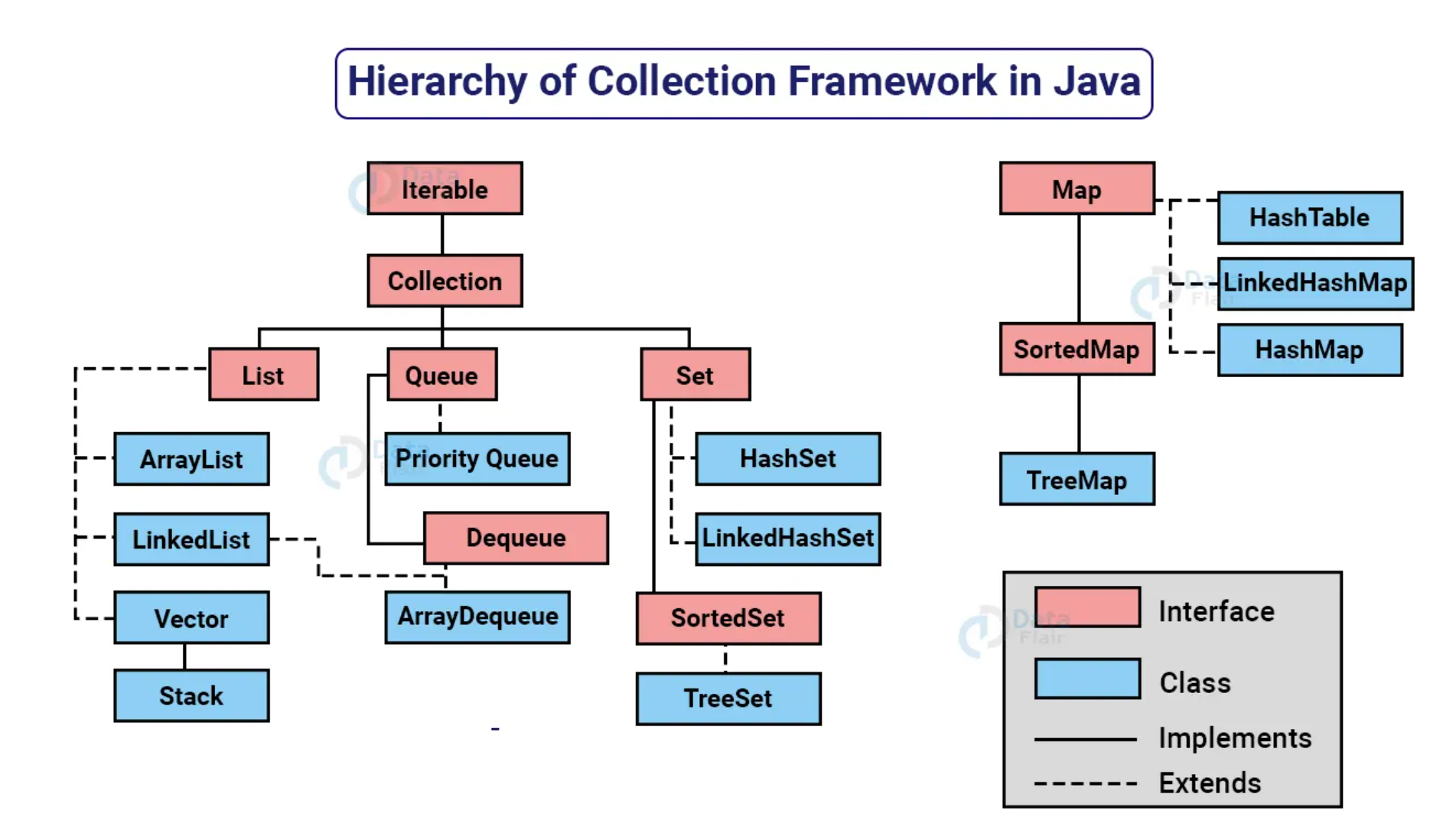
Вопросы для подготовки:

- Какие данные могут хранить коллекции?

Любые ссылочные типы данных

- Расскажите про иерархию коллекций?



- Расскажите о коллекциях типа List?

ArrayList

реализация интерфейса List. Представляет собой автоматически расширяемый массив. ArrayList может менять свой размер во время исполнения программы. В основе лежит обычный массив.

LinkedList

еще одна реализация интерфейса List. Представляет из себя двусвязный список, где каждый элемент структуры содержит ссылки на предыдущий и следующий элементы. По этим ссылкам можно переходить от одного элемента к другому.

- Расскажите о коллекциях типа Set?

В множествах Set каждый элемент хранится только в одном экземпляре, а разные реализации Set используют разный порядок хранения элементов. В HashSet порядок элементов определяется по сложному алгоритму. Если порядок хранения для вас важен, используйте контейнер TreeSet, в котором объекты хранятся отсортированными по возрастанию в порядке сравнения или LinkedHashSet с хранением элементов в порядке добавления.

Множества часто используются для проверки принадлежности, чтобы вы могли легко проверить, принадлежит ли объект заданному множеству, поэтому на практике обычно выбирается реализация HashSet, оптимизированная для быстрого поиска

HashSet – в основе лежит хэш-таблица, позиция элемента определяется по его хэшкоду

LinkedHashSet

расширяет класс HashSet, не добавляя никаких новых методов. Класс поддерживает связный список элементов набора в том порядке, в котором они вставлялись. Это позволяет организовать упорядоченную итерацию вставки в набор.

TreeSet – это коллекция, которая хранит элементы в виде упорядоченного по значениям дерева. Внутри TreeSet<E> содержится TreeMap<E, Object> который и хранит все эти значения. А этот TreeMap использует красно—черное сбалансированное бинарное дерево для хранения элементов. Поэтому у него очень быстрые операции add, remove, contains.

- Расскажите о коллекциях типа Queue?

Queue — коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки. В дополнение к базовым операциям интерфейса Collection, очередь предоставляет дополнительные операции вставки, получения и контроля.

Очереди обычно, но не обязательно, упорядочивают элементы в FIFO (first-in-first-out, «первым вошел — первым вышел») порядке.

Метод offer() вставляет элемент в очередь, если это не удалось — возвращает false. Этот метод отличается от метода add() интерфейса Collection тем, что метод add() может не выполнить добавление элемента только с использованием unchecked исключения.

Методы remove() и poll() удаляют верхушку очереди и возвращают ее. Какой элемент будет удален (первый или последний) зависит от реализации очереди. Методы remove() и poll() отличаются лишь поведением, когда очередь пустая: метод remove() генерирует исключение, а метод poll() возвращает null.

Методы element() и peek() возвращают (но не удаляют) верхушку очереди.

java.util.Queue<E> реализует FIFO–буфер. Позволяет добавлять и получать объекты. При этом объекты могут быть получены в том порядке, в котором они были добавлены.

Реализации: java.util.ArrayDeque<E>, java.util.LinkedList<E>.

java.util.Deque<E> наследует java.util.Queue<E>. Двунаправленная очередь. Позволяет добавлять и удалять объекты с двух концов. Так же может быть использован в качестве стека.

Реализации: java.util.ArrayDeque<E>, java.util.LinkedList<E>.

- Расскажите о коллекциях типа Map, в чем их принципиальное отличие?

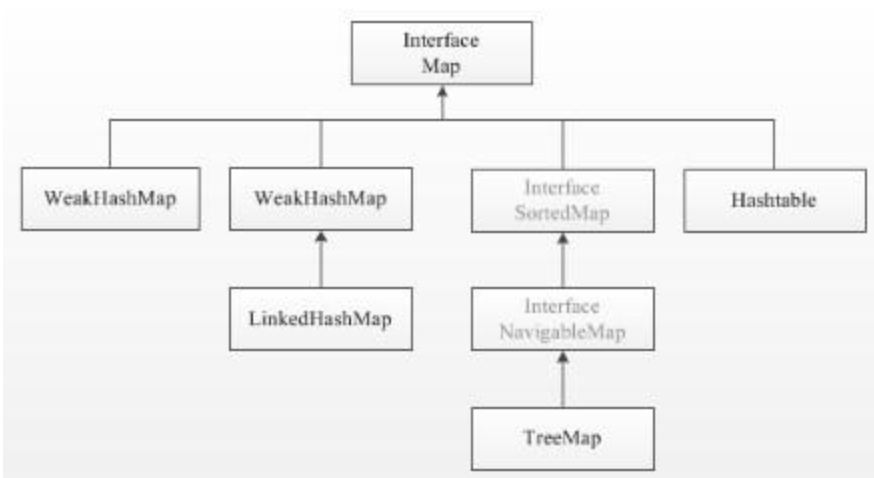
Map — это структура данных, которая содержит набор пар “ключ-значение”. По своей структуре данных напоминает словарь, поэтому ее часто так и называют. В то же время, Map является интерфейсом, и в стандартном jdk содержит основные реализации: Hashmap, LinkedHashMap, Hashtable, TreeMap.

Как упоминалось ранее, существуют 3 основные реализации интерфейса Map. У каждой из них есть свои особенности:

Порядок элементов. HashMap и Hashtable не гарантируют, что элементы будут храниться в порядке добавления. Кроме того, они не гарантируют, что порядок элементов не будет меняться со временем. В свою очередь, TreeMap гарантирует хранение элементов в порядке добавления или же в соответствии с заданным компаратором.

Допустимые значения. HashMap позволяет иметь ключ и значение null, HashTable — нет. TreeMap может использовать значения null только если это позволяет компаратор. Без использования компаратора (при хранении пар в порядке добавления) значение null не допускается.

Синхронизация. Только HashTable синхронизирована, остальные — нет. Если к мапе не будут обращаться разные потоки, рекомендуется использовать HashMap вместо HashTable.



- Назовите основные реализации List, Set, Map.

List: arraylist, linkedlist, vector, stack

Set: hashset, treeset, sortedset

Map: hashmap, treemap, sortedmap, hashtable

- Какие реализации SortedSet вы знаете и в чем их особенность?

java.util.SortedSet<E> наследует java.util.Set<E>. Реализации этого интерфейса, помимо того что следят за уникальностью хранимых объектов, поддерживают их в порядке возрастания. Отношение порядка между объектами может быть определено, как с помощью метода compareTo интерфейса java.lang.Comparable<T>, так и при помощи специального класса-компаратора, наследующего интерфейс java.util.Comparator<T>.

Реализации: java.util.TreeSet<E> — коллекция, которая хранит свои элементы в виде упорядоченного по значениям дерева. TreeSet инкапсулирует в себе TreeMap, который в свою очередь использует сбалансированное бинарное красно-черное дерево для хранения элементов. TreeSet хорош тем, что для операций add, remove и contains потребуется гарантированное время log(n).

- Что разного/общего у классов ArrayList и LinkedList, когда лучше использовать ArrayList, а когда LinkedList?

Если необходимо вставлять (или удалять) в середину коллекции много элементов, то лучше использовать LinkedList. Во всех остальных случаях – ArrayList.

LinkedList требует больше памяти для хранения такого же количества элементов, потому что кроме самого элемента хранятся еще указатели на следующий и предыдущий элементы списка, тогда как в ArrayList элементы просто идут по порядку

-В каких случаях разумно использовать массив, а не ArrayList?

Oracle пишет — используйте ArrayList вместо массивов.

массивы могут быть быстрее и потреблять меньше памяти. Списки теряют в производительности из-за возможности автоматического увеличения размера и сопутствующих проверок. Плюс к этому, что размер списка увеличивается не на 1, а в 1.5 раза. Так же доступ к [10] в массиве может быть быстрее чем вызов get(10) у списка.

- Чем отличается ArrayList от Vector?

Vector is deprecated. У Vector некоторые методы синхронизированы и поэтому они медленные. В любом случае Vector не рекомендуется использовать вообще.

- Что вы знаете о реализации классов HashSet и TreeSet?

HashSet реализован на основе хеш-таблицы, а TreeSet — на основе бинарного дерева.

- Чем отличаются HashMap и TreeMap? Как они устроены и работают? Что со временем доступа к объектам, какие зависимости?

HashMap работает строго быстрее TreeMap.

TreeMap реализован на красно-черном дереве, время добавления/поиска/удаления элемента — O(log N), где N — число элементов в TreeMap на данный момент.

У HashMap время доступа к отдельному элементу — O(1) при условии, что хэш-функция (Object.hashCode()) определена нормально (что является правдой в случае Integer).

- Что такое Hashtable, чем она отличается от HashMap?

Некоторые методы HashTable синхронизированы, поэтому она медленнее HashMap.

HashTable синхронизирована, а HashMap нет.

HashTable не позволяет иметь null ключи или значения. HashMap позволяет иметь один null ключ и сколько угодно null значений.

У HashMap есть подкласс LinkedHashMap, который добавляет возможности по итерации. Если вам нужна эта функциональность, то можно легко переключаться между классами.

Общее замечание — не рекомендуется использовать HashTable даже в многопоточных приложениях. Для этого есть ConcurrentHashMap.

- Что будет, если в Map положить два значения с одинаковым ключом?

Последнее значение перезапишет предыдущее.

- Как задается порядок следования объектов в коллекции, как отсортировать коллекцию?

Класс ТгееМар полностью реализует интерфейс SortedMap. Он реализован как бинарное дерево поиска, значит его элементы хранятся в упорядоченном виде. Это значительно ускоряет поиск нужного элемента. Порядок задается либо естественным следованием элементов, либо объектом, реализующим интерфейс сравнения Comparator.

- Дайте определение понятию “итератор”.

Итератор — объект, позволяющий перебирать элементы коллекции. Например foreach реализован с использованием итератора. Одним из ключевых методов интерфейса Collection является метод Iterator<E> iterator(). Он возвращает итератор — то есть объект, реализующий интерфейс Iterator.

- Какую функциональность представляет класс Collections?

|  |  |
| --- | --- |
| Collections.sort(List myList) | Сортирует список в естественном порядке. |
| Collections.sort(List, Comparator c) | Сортировка с использованием компаратора. |
| Collections.shuffle(List myList) | Перемешивает коллекцию в случайном порядке. |
| Collections.reverse(List myList) | Переворачивает коллекцию в обратном порядке. |
| Collections.binarySearch(List mlist, T key) | поиск в коллекции по ключу с использованием бинарного поиска. |
| Collections.copy(List dest, List src) | Копирует коллекцию источник src в dest. |
| Collections.frequency(Collection c, Object o) | Возвращает число вхождений объекта в коллекции. |
| Collections.synchronizedCollection(Collection c) | Возвращает синхронизированную (потокобезопасную) коллекцию. |

- Как получить не модифицируемую коллекцию?

Collections.unmodifiableList(list);

Collections.unmodifiableSet(set);

Collections.unmodifiableMap(map);

- Какие коллекции синхронизированы?

Для этого используется пакет Concurrent. А так @Deprecated HashTable, Vector.

- Как получить синхронизированную коллекцию из не синхронизированной?

Collections.synchronizedList(list);

Collections.synchronizedSet(set);

Collections.synchronizedMap(map);

- Как получить коллекцию только для чтения?

Collections.unmodifiableList(list);

Collections.unmodifiableSet(set);

Collections.unmodifiableMap(map);

- Почему Map не наследуется от Collection?

Они не совместимы, т.к. созданы для различных структур данных. Map использует пару ключ-значение.

- В чем разница между Iterator и Enumeration?

Enumeration в два раза быстрее Iterator и использует меньше памяти. Iterator потокобезопасен, т.к. не позволяет другим потокам модифицировать коллекцию при переборе. Enumeration можно использовать только для read-only коллекций. Так же у него отсутствует метод remove();

Enumeration: hasMoreElement(), nextElement()

Iterator: hasNext(), next(), remove()

- Как реализован цикл foreach?

Реализован на основе Iterator.

for(тип итер\_пер : коллекция)

- Почему нет метода iterator.add() чтобы добавить элементы в коллекцию?

Единственная задача итератора это перебор коллекции

- Почему в классе iterator нет метода для получения следующего элемента без передвижения курсора?

Итератор похож на указатель своими основными операциями: он указывает на отдельный элемент коллекции объектов (предоставляет доступ к элементу) и содержит функции для перехода к другому элементу списка (следующему или предыдущему). Контейнер, который реализует поддержку итераторов, должен предоставлять первый элемент списка, а также возможность проверить, перебраны ли все элементы контейнера (является ли итератор конечным). Таким образом без курсора просто нельзя будет реализовать безошибочное передвижение по коллекции.

- В чем разница между Iterator и ListIterator?

Iterator может использоваться для перебора элементов Set, List и Map. В отличие от него, ListIterator может быть использован только для перебора элементов коллекции List

Iterator позволяет перебирать элементы только в одном направлении, при помощи метода next(). Тогда как ListIterator позволяет перебирать список в обоих направлениях, при помощи методов next() и previous()

При помощи ListIterator вы можете модифицировать список, добавляя/удаляя элементы с помощью методов add() и remove(). Iterator не поддерживает данную функциональность.

- Какие есть способы перебора всех элементов List?

Цикл с итератором

Цикл for

Расширенный цикл for

Цикл while

- В чем разница между fail-safe и fail-fast свойствами?

В противоположность fail-fast, итераторы fail-safe не вызывают никаких исключений при изменении структуры, потому что они работают с клоном коллекции вместо оригинала.

Итератор коллекции CopyOnWriteArrayList и итератор представления keySet коллекции ConcurrentHashMap являются примерами итераторов fail-safe.

- Что делать, чтобы не возникло исключение ConcurrentModificationException?

Первым делом, можно подобрать другой итератор, работающий по принципу fail-safe. К примеру, если вы используете List, то можете взять ListIterator. Если же вам нужна устаревшая коллекция — то используйте перечислители.

В том случае, когда вышеизложенное вам не подходит, у вас есть три варианта:

При использовании JDK 1.5 или выше, вам подойдут классы ConcurrentHashMap и CopyOnWriteArrayList. Это самый лучший вариант

Вы можете преобразовать список в массив и перебирать массив

Вы можете блокировать изменения списка на время перебора с помощью блока synchronized

Обратите внимание, что последние два варианта негативно скажутся на производительности.

- Что такое стек и очередь, расскажите в чем их отличия?

Коллекции созданы для того чтобы хранить элементы для дальнейшей обработки. Кроме базовых операций интерфейса Collection, очереди поддерживают дополнительные операции добавления, удаления и проверки состояния элемента.

Обычно, но не обязательно очереди работают по принципу FIFO — первым пришел, первым ушел.

Стэк — почти как очередь, но работает по принципу LIFO — последним пришел, первым ушел.

Независимо от порядка добавления/удаления, голова очереди это элемент, который будет удален при вызове методов remove() или poll(). Также обратите внимание на то, что Stack и Vector оба потокобезопасны.

- В чем разница между интерфейсами Comparable и Comparator?

Интерфейс Comparable помогает сохранять естественную сортировку, тогда как Comparator позволяет сортировать элементы по разным особым шаблонам. Экземпляр компаратора обычно передается конструктору коллекции, если коллекция это поддерживает. Следует отметить, что интерфейс Comparable может быть реализован именно элементами коллекции или ключами Map, а Comparator реализуется отдельным объектом (это удобно, так как можно заготовить несколько реализаций для разных правил сортировок, не меняя при этом код элементов коллекции/ключей Map).

- Почему коллекции не наследуют интерфейсы Cloneable и Serializable?

Потому что это не нужно и расходует много памяти.