List<?>

* ArrayList<String>() // BUSCAR
  + Dimensionado dinamicamente (sempre que necessário o seu tamanho aumenta 50% do tamanho da lista - ela cria uma lista 50% maior e copia os elementos para a nova e apaga a antiga)
  + Vantagens:
    - Pode acessar qualquer elemento diretamente através do seu index
  + Desvantagens:
    - Não remove os elementos duplicados
    - Não são sincronizados (se sua aplicação precisa trabalhar com thread-safe, descarte a utilização de ArrayList)
* Vector // CONCORRÊNCIA (custo alto)
  + Muito similar ao ArrayList
  + Diferença em relação ao ArrayList:
    - São sincronizados (se sua aplicação precisar trabalhar com thread-safe, use Vector)
    - Enquanto ArrayList aumenta 50% (ou seja mais metade da quantidade antiga), Vector aumenta o dobro.
* LinkedList<Integer>() // MODIFICADOR
  + Vantagem:
    - Melhor performance nos métodos add e remove em relação ao ArrayList (rápido para adicionar e remover)
  + Desvantagem:
    - Pior performance nos métodos get e set em relação ao ArrayList (lento para buscar elemento)
  + Comparação:
    - add(E element) - LinkedList O(1) - ArrayList O(n) no pior caso
    - add(int index, E element) - LinkedList O(n) - ArrayList O(n) no pior caso)
    - remove(int index) - LinkedList é melhor se remover no começo e ArrayList é melhor se remover no final
    - get(int index) - LinkedList O(n) - ArrayList O(1)

Set<?>

(São conhecidos por aceitar apenas valores únicos)

* HashSet<Carro> TEM QUE FORNECER O EQUALS E O HASHCODE
  + Vantagem:
    - É o mais rápido de todos (a complexidade para os métodos add, remove e contains são O(1)), (não importa o que você adicione, remova, retire o tempo de execução sempre será O(1))
  + Desvantagem:
    - Não tem ordem
* TreeSet<Carro> TEM QUE FORNECER O EQUALS E O HASHCODE E O COMPARE TO
  + Vantagem:
    - Possui o método SortedSet, que ordena e tira as duplicatas automaticamente
  + Desvantagem:
    - Como possui o método SortedSet (a complexidade para os métodos add, remove e contains são O(log n))

Map<?>

* HashMap<K,V>
  + Utiliza chaves
  + Usa o método put() para inserir um novo elemento na lista
    - Ex: hmap.put(“K14”, new Pessoa(“nome”, idade))
* HashTable<K,V>
  + Muito similar ao HashMap
  + Vantagem:
    - São sincronizados (se sua aplicação precisar trabalhar com thread-safe, use HashTable)