ArrayList: Estrutura baseada em array dinâmico, que cresce 50% quando todas as posições estão ocupadas.
Possui excelente desempenho para busca de informações (acesso por índice é O(1)), mas é menos eficiente para adição e remoção em posições específicas, pois exige realocação dos elementos.
Ideal para cenários com alta frequência de leitura e poucas modificações na estrutura.

LinkedList: Estrutura baseada em lista duplamente ligada, onde cada elemento (nó) armazena três informações: o ponteiro para o elemento anterior, o ponteiro para o próximo e o valor atual.
Cresce conforme a demanda, sem necessidade de realocação. Apresenta desempenho inferior na busca (acesso sequencial, O(n)), mas é eficiente para inserções e remoções em qualquer posição, pois só requer ajuste dos ponteiros.
Ideal para cenários com muitas modificações na estrutura (adições/remoções frequentes).

List: Coleção ordenada que permite elementos repetidos. A ordem é mantida conforme a inserção.
Exemplo: ArrayList, LinkedList.

Set: Coleção que não permite elementos duplicados. A ordem pode ser imprevisível (ex: HashSet), ou mantida conforme inserção (ex: LinkedHashSet), ou ordenada (ex: TreeSet).
Exemplo: Cadastro de CPFs, onde repetição não faz sentido.

Map: Estrutura que associa chaves únicas a valores (pares chave-valor). Não é uma lista, mas um mapeamento.

Exemplo: Dicionário, onde a chave “palavra” está associada a um valor “definição”.

HashSet: Implementação de Set baseada em tabela hash. Não garante ordem dos elementos e permite acesso rápido. Ideal para garantir unicidade de elementos sem se preocupar com a ordem.

LinkedHashSet: Variante do HashSet que mantém a ordem de inserção dos elementos. Usa uma tabela hash e uma lista duplamente ligada para rastrear a ordem. Útil quando é necessário eliminar duplicatas e manter a ordem.

TreeSet: Implementação baseada em árvore rubro-negra. Garante ordenação natural dos elementos (ou via Comparator), mas é mais lenta que HashSet. Ideal quando a ordenação é necessária.

HashMap: Implementação de Map baseada em tabela hash. Permite uma chave null e acesso rápido aos valores via chave. Não garante ordem das entradas.

LinkedHashMap: Variante do HashMap que mantém a ordem de inserção. Útil para aplicações onde a ordem previsível das entradas é importante, como geração de relatórios ou implementação de cache LRU.

TreeMap: Implementação de Map baseada em árvore rubro-negra. Mantém as chaves ordenadas (ordem natural ou personalizada). Ideal para estruturas que exigem ordenação ou navegação entre intervalos de chave.