MÓDULO 1 - EXTRA
APROFUNDANDO EM COLEÇÕES

UNIPDS
EDUCAÇÃO

Database de itens de cardápio

```
public class Database {
  public List<ItemCardapio> listaItensCardapio() {

   List<ItemCardapio> itens = new ArrayList<>();
    var refrescoDoChaves = new ItemCardapio(1L, "Refresco do Chaves",
        "Suco de limão que parece de tamarindo e tem gosto de groselha.",
        BEBIDAS, new BigDecimal("2.99"), null);

    itens.add(refrescoDoChaves);
    //...

    return itens;
  }
}
```

ArrayList

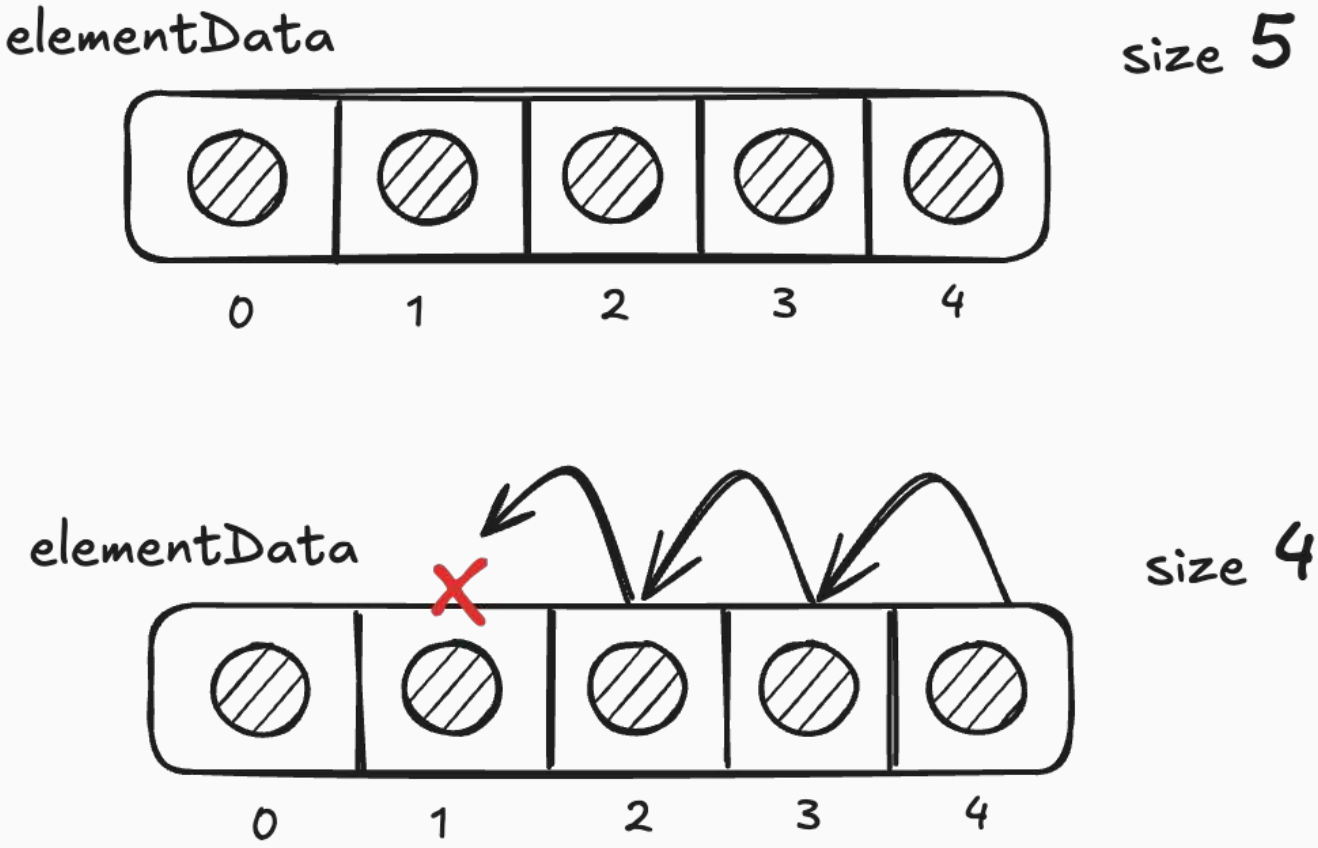
Usando Database

```java
public class Main {
  public static void main(String[] args) {
    Database database = new Database();
    List<ItemCardapio> itens = database.listaItensCardapio();
    System.out.println(itens.size());

    ItemCardapio itemCardapio = itens.get(2);
    System.out.println(itemCardapio.nome());

    itens.remove(1);
    System.out.println(itens.size());
  }
}
```

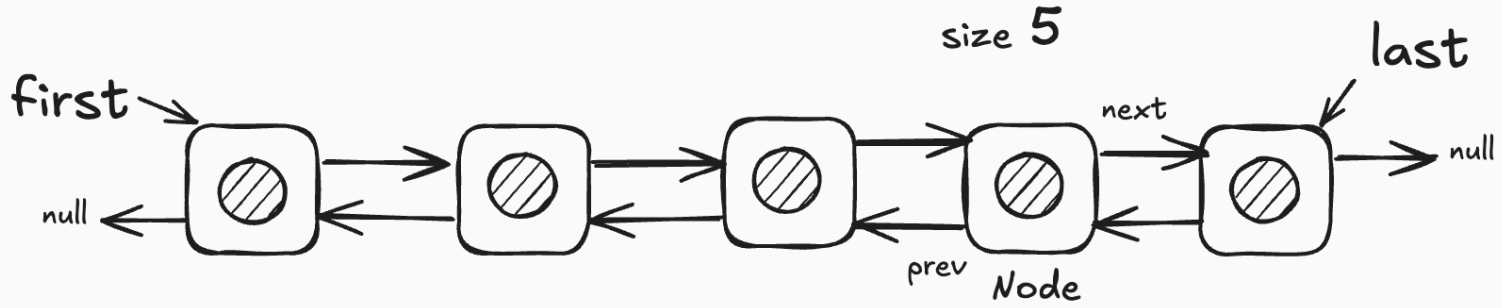
```
9
Torta de Frango da Dona Florinda
8
```

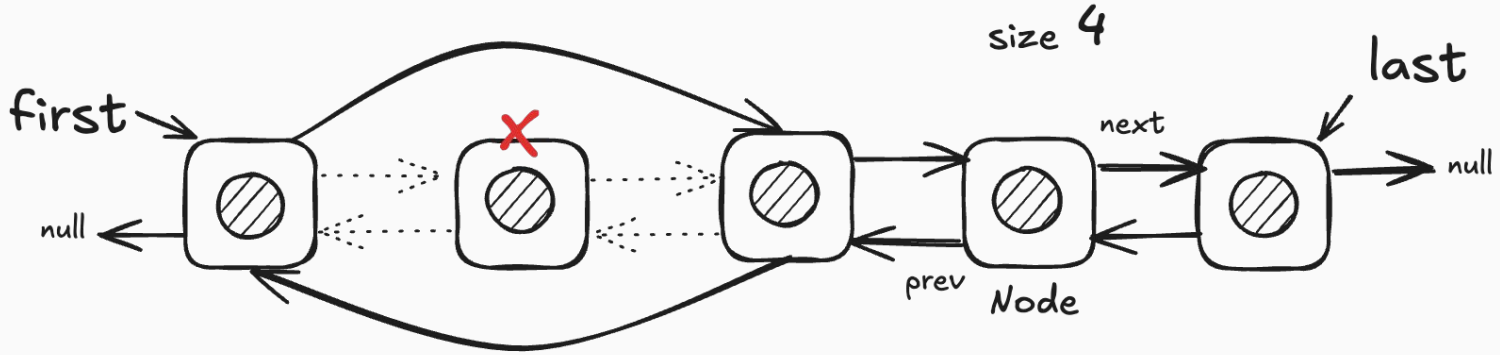
LinkedList

Removendo de ArrayList



Removendo de LinkedList





Mudando para LinkedList

```
public class Database {
  public List<ItemCardapio> listaItensCardapio() {

   List<ItemCardapio> itens = new LinkedList<>();
    var refrescoDoChaves = new ItemCardapio(1L, "Refresco do Chaves",
        "Suco de limão que parece de tamarindo e tem gosto de groselha.",
        BEBIDAS, new BigDecimal("2.99"), null);

    itens.add(refrescoDoChaves);
    //...

    return itens;
  }
}
```

O poder do polimorfismo

```
public class Main {
  public static void main(String[] args) {
    Database database = new Database();
    List<ItemCardapio> itens = database.listaItensCardapio();
    System.out.println(itens.size());

    ItemCardapio itemCardapio = itens.get(2);
    System.out.println(itemCardapio.nome());

    itens.remove(1);
    System.out.println(itens.size());
  }
}
```

Não precisou mudar nada!

E o resultado continua o mesmo!

```
9
Torta de Frango da Dona Florinda
8
```

TAREFA

PRECISO SABER QUAIS AS CATEGORIAS REALMENTE TENHO NO CARDÁPIO

Mostrando categorias

```
for (ItemCardapio item : itens) {
    CategoriaCardapio categoria = item.categoria();
    System.out.println(categoria);
}
```

ou

Stream API

```
itens.stream()
    .map(ItemCardapio::categoria)
    .forEach(System.out::println);
```

Valores repetidos

```
BEBIDAS
PRATOS_PRINCIPAIS
PRATOS_PRINCIPAIS
PRATOS_PRINCIPAIS
BEBIDAS
BEBIDAS
SOBREMESA
SOBREMESA
SOBREMESA
```

HashSet para não repetir

```java
Set<CategoriaCardapio> categoriasUnicas = new HashSet<>();
for (ItemCardapio item : itens) {
    categoriasUnicas.add(item.categoria());
}
for (CategoriaCardapio categoria : categoriasUnicas) {
    System.out.println(categoria);
}
```

HashSet

ou

Stream API

```java
itens.stream()
    .map(ItemCardapio::categoria)
    .collect(Collectors.toSet())
    .forEach(System.out::println);
```

Mas não manteve a ordem...

```
PRATOS_PRINCIPAIS
SOBREMESA
BEBIDAS
```

LinkedHashSet

LinkedHashSet para manter a ordem

```
Set<CategoriaCardapio> categoriasUnicas = new LinkedHashSet<>();
for (ItemCardapio item : itens) {
    categoriasUnicas.add(item.categoria());
}
for (CategoriaCardapio categoria : categoriasUnicas) {
    System.out.println(categoria);
}
```

LinkedHashSet

Mantém a ordem de inserção.
E sem repetir elementos!

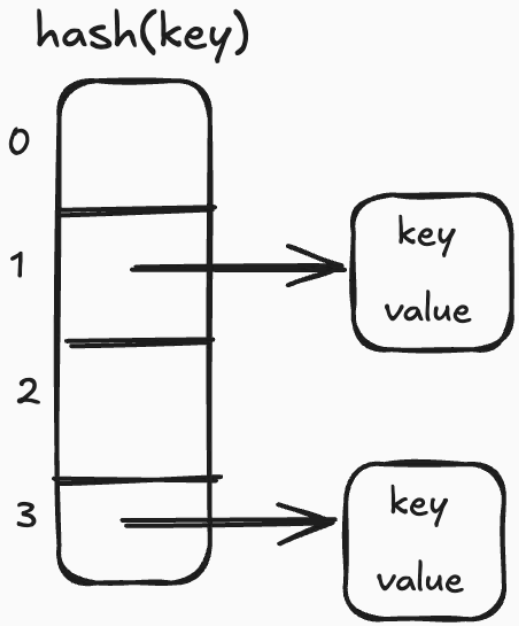
ou

Stream API

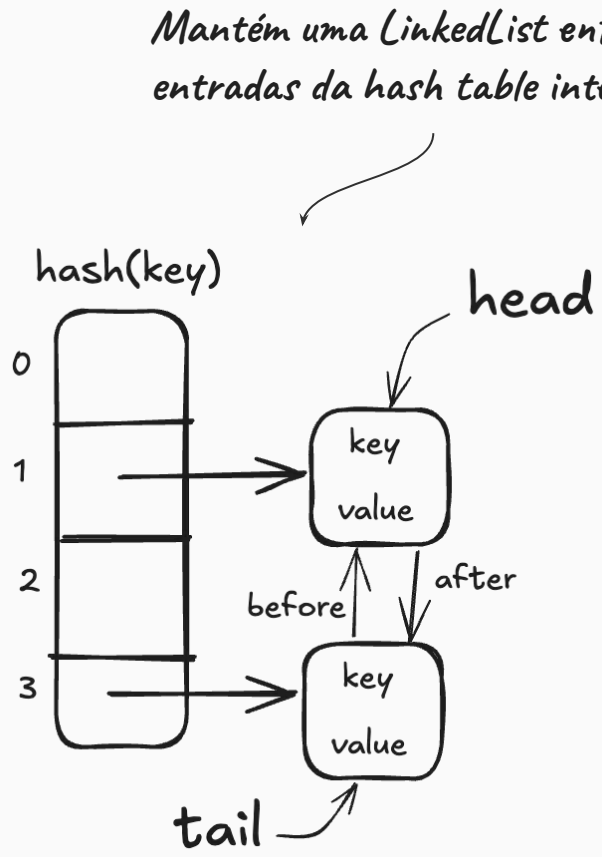
```
itens.stream()
  .map(ItemCardapio::categoria)
  .collect(Collectors.toCollection(LinkedHashSet::new))
  .forEach(System.out::println);
```

```
BEBIDAS
PRATOS_PRINCIPAIS
SOBREMESA
```

Como isso é possível?



HashSet



LinkedHashSet

TreeSet

TreeSet para ordenar

```
Set<CategoriaCardapio> categoriasUnicas = new TreeSet<>();
for (ItemCardapio item : itens) {
    categoriasUnicas.add(item.categoria());
}
for (CategoriaCardapio categoria : categoriasUnicas) {
    System.out.println(categoria);
}
```

TreeSet

ou

Stream API

```
itens.stream()
  .map(ItemCardapio::categoria)
  .collect(Collectors.toCollection(TreeSet::new))
  .forEach(System.out::println);
```

Mantém a ordenado, na ordem de enum...

```
PRATOS_PRINCIPAIS
BEBIDAS
SOBREMESA
```

TreeSet para ordenar

```
Comparator<CategoriaCardapio> categoriaComparator = Comparator.comparing(CategoriaCardapio::name);

Set<CategoriaCardapio> categoriasUnicas = new TreeSet<>(categoriaComparator);
for (ItemCardapio item : itens) {
    categoriasUnicas.add(item.categoria());
}
for (CategoriaCardapio categoria : categoriasUnicas) {
    System.out.println(categoria);
}
```

Comparator

ou

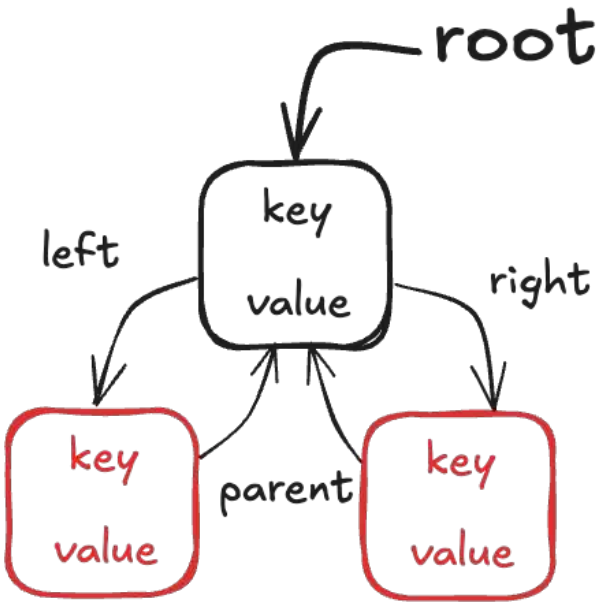
Stream API

```
itens.stream()
  .map(ItemCardapio::categoria)
  .collect(Collectors.toCollection(
              () -> new TreeSet<>(categoriaComparator)))
  .forEach(System.out::println);
```

Mantém a ordenado, na ordem do comparator!

```
BEBIDAS
PRATOS_PRINCIPAIS
SOBREMESA
```

Como isso é possível?



TreeSet

Árvore rubro-negra

Esse não

TAREFA

PRECISO SABER QUANTOS ITENS DE CADA CATEGORIA EXISTEM NO CARDÁPIO.

HashMap para associar

```java
Map<CategoriaCardapio, Integer> itensPorCategoria = new HashMap<>();
for (ItemCardapio item : itens) {
    int quantidade;
    if (itensPorCategoria.containsKey(item.categoria())) {
        quantidade = itensPorCategoria.get(item.categoria()) + 1;
    } else {
        quantidade = 1;
    }
    itensPorCategoria.put(item.categoria(), quantidade);
}
for (CategoriaCardapio categoria : itensPorCategoria.keySet()) {
    Integer quantidade = itensPorCategoria.get(categoria);
    System.out.printf("%s: %d\n", categoria, quantidade);
}
```

HashMap

Não mantém a ordem...

```
PRATOS_PRINCIPAIS: 3
SOBREMESA: 1
BEBIDAS: 2
```

HashMap com Stream API

```
itens.stream()
  .collect(Collectors.groupingBy(
      ItemCardapio::categoria,
      Collectors.counting()
  ))
  .forEach((categoria, quantidade) ->
      System.out.printf("%s: %d\n", categoria, quantidade));
```

Stream API

Mesmo resultado

```
PRATOS_PRINCIPAIS: 3
SOBREMESA: 1
BEBIDAS: 2
```

LinkedHashMap

LinkedHashMap para ordem das chaves

```java
Map<CategoriaCardapio, Integer> itensPorCategoria = new LinkedHashMap<>();
for (ItemCardapio item : itens) {
    int quantidade;
    if (itensPorCategoria.containsKey(item.categoria())) {
        quantidade = itensPorCategoria.get(item.categoria()) + 1;
    } else {
        quantidade = 1;
    }
    itensPorCategoria.put(item.categoria(), quantidade);
}
for (CategoriaCardapio categoria : itensPorCategoria.keySet()) {
    Integer quantidade = itensPorCategoria.get(categoria);
    System.out.printf("%s: %d\n", categoria, quantidade);
}
```

LinkedHashMap

Mantém a ordem de inserção das chaves

```
BEBIDAS: 2
PRATOS_PRINCIPAIS: 3
SOBREMESA: 1
```

LinkedHashMap com Stream API

```
itens.stream()
   .collect(Collectors.groupingBy(
       ItemCardapio::categoria,
       LinkedHashMap::new,
       Collectors.counting()
   ))
   .forEach((categoria, quantidade) ->
       System.out.printf("%s: %d\n", categoria, quantidade));
}
```

Stream API

Mantém a ordem de inserção das chaves

```
BEBIDAS: 2
PRATOS_PRINCIPAIS: 3
SOBREMESA: 1
```

TreeMap

TreeMap para ordenar as chaves

```
Map<CategoriaCardapio, Integer> itensPorCategoria = new TreeMap<>();
for (ItemCardapio item : itens) {
    int quantidade;
    if (itensPorCategoria.containsKey(item.categoria())) {
        quantidade = itensPorCategoria.get(item.categoria()) + 1;
    } else {
        quantidade = 1;
    }
    itensPorCategoria.put(item.categoria(), quantidade);
}
for (CategoriaCardapio categoria : itensPorCategoria.keySet()) {
    Integer quantidade = itensPorCategoria.get(categoria);
    System.out.printf("%s: %d\n", categoria, quantidade);
}
```

TreeMap

Ordem da enum

```
PRATOS_PRINCIPAIS: 3
BEBIDAS: 2
SOBREMESA: 1
```

Poderia usar Comparator

TreeMap com Stream API

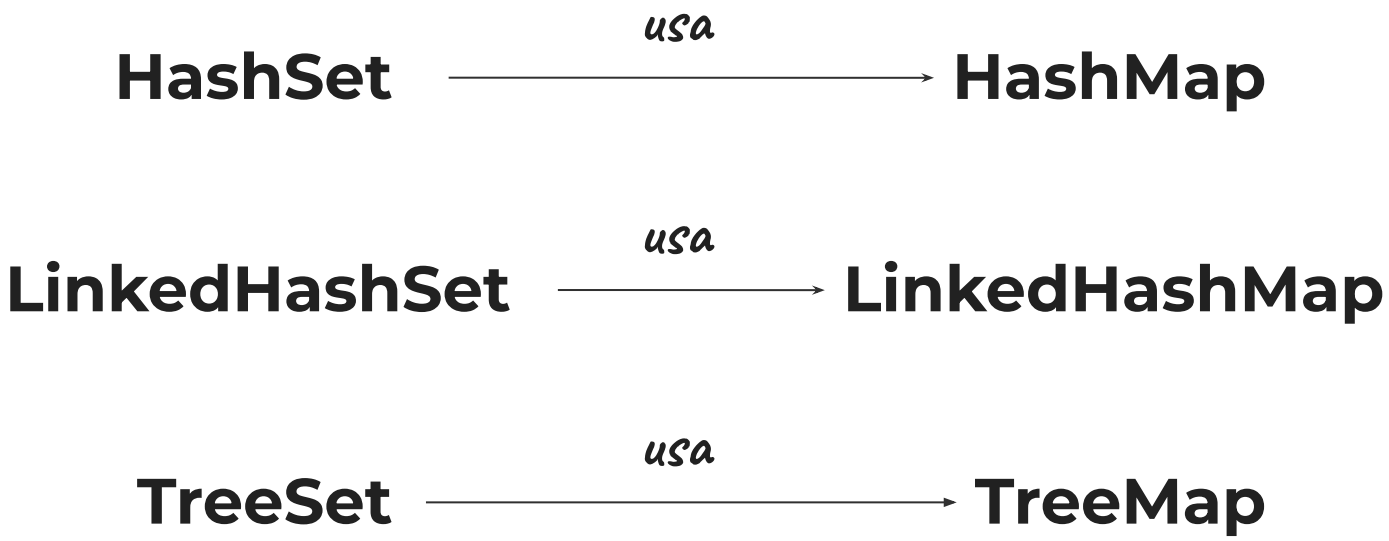
```
itens.stream()
   .collect(Collectors.groupingBy(
       ItemCardapio::categoria,
       TreeMap::new,
       Collectors.counting()
   ))
   .forEach((categoria, quantidade) ->
       System.out.printf("%s: %d\n", categoria, quantidade));
}
```

Stream API

Mesmo resultado

```
PRATOS_PRINCIPAIS: 3
BEBIDAS: 2
SOBREMESA: 1
```

Set e Map


HashSet
usa
HashMap
LinkedHashSet
usa
LinkedHashMap
TreeSet
usa
TreeMap

EXERCÍCIO

MUDAR O DATABASE PARA TER UMA BUSCA POR ID ALÉM DA LISTAGEM.

Map de itens por id

```java
public class Database {

  private final Map<Long, ItemCardapio> itensPorId = new HashMap<>();

  public Database() {
      var refrescoDoChaves = new ItemCardapio(1L, "Refresco do Chaves",
              "Suco de limão que parece de tamarindo e tem gosto de groselha.",
              BEBIDAS, new BigDecimal("2.99"), null);
      itensPorId.put(refrescoDoChaves.id(), refrescoDoChaves);
      // outros itens...
  }

  public List<ItemCardapio> listaItensCardapio() {
    return new LinkedList<>(itensPorId.values());
  }

  public Optional<ItemCardapio> itemCardapioPorId(Long id) {
    return Optional.ofNullable(itensPorId.get(id));
  }

}
```

HashMap<Long, ItemCardapio>

Populando no construtor

map.values() para obter a lista de todos os itens

map.get() para obter o item por id

Usando a busca por id

```
Optional<ItemCardapio> optionalItemCardapio = database.itemCardapioPorId(id);
if (optionalItemCardapio.isPresent()) {
    ItemCardapio itemCardapio = optionalItemCardapio.get();
    System.out.println(itemCardapio);
} else {
    System.out.printf("Item de id %d não encontrado%n", id);
}
```

ou

```
String mensagem = database.itemCardapioPorId(id)
    .map(ItemCardapio::toString)
    .orElse("Item de id %d não
encontrado".formatted(id));
System.out.println(mensagem);
```

Estilo mais funcional

Resultado

```
ItemCardapio[id=1,
nome=Resfresco ...,
categoria=BEBIDAS,
preco=2.99,
precoPromocional=null]
```

TAREFA

PRECISO MANTER AS CATEGORIAS QUE ESTÃO EM PROMOÇÃO.

Categorias em Promoção com Set

```
Set<CategoriaCardapio> categoriasEmPromocao = new TreeSet<>();
categoriasEmPromocao.add(CategoriaCardapio.SOBREMESA);
categoriasEmPromocao.add(CategoriaCardapio.ENTRADAS);
categoriasEmPromocao.forEach(System.out::println);
```

Ordem do enum

```
ENTRADAS
SOBREMESA
```

ou

```
Set.of(CategoriaCardapio.SOBREMESA, CategoriaCardapio.ENTRADAS)
    .forEach(System.out::println);
```

Ordem não definida, não modificável

```
SOBREMESA
ENTRADAS
```

EnumSet

Categorias em Promoção com EnumSet

EnumSet

```
Set<CategoriaCardapio> categoriasEmPromocao =
    EnumSet.of(CategoriaCardapio.SOBREMESA, CategoriaCardapio.ENTRADAS);
categoriasEmPromocao.forEach(System.out::println);
```

Ordem do enum, semelhante ao TreeSet

```
ENTRADAS
SOBREMESA
```

- Extremamente eficiente
- Representado internamente com bits

EnumMap

Categorias em Promoção com EnumMap


```
EnumMap<CategoriaCardapio, String> promocoes = new EnumMap<>(CategoriaCardapio.class);
promocoes.put(CategoriaCardapio.SOBREMESA, "O doce perfeito para você!");
promocoes.put(CategoriaCardapio.ENTRADAS, "Comece sua refeição com um toque de sabor!");

String descricao = promocoes.get(CategoriaCardapio.ENTRADAS);
System.out.printf("Entradas: %s\n", descricao);
```

- Extremamente eficiente
- Representado internamente com bits

```
Entradas: Comece
sua refeição com
um toque de sabor!
```

TAREFA

PRECISO DE UM HISTÓRICO DE VISUALIZAÇÃO DO CARDÁPIO.

Histórico de visualizações com HashMap

```
public class HistoricoVisualizacao {

  private Map<ItemCardapio, LocalDateTime> visualizacoes = new HashMap<>();
  private final Database database;

  public HistoricoVisualizacao(Database database) {
      this.database = database;
  }

}
```

Histórico de visualizações com HashMap

```
public class HistoricoVisualizacao {
  //...

  public void registrarVisualizacao(Long itemId) {
    Optional<ItemCardapio> optionalItem = database.itemCardapioPorId(itemId);
    if (optionalItem.isEmpty()) {
      System.out.printf("Item não encontrado: %d", itemId);
      return;
    }
    ItemCardapio itemCardapio = optionalItem.get();
    LocalDateTime agora = LocalDateTime.now();
    visualizacoes.put(itemCardapio, agora);
    System.out.printf("'%s' visualizado em '%s'\n", itemCardapio.nome(), agora);
  }
}
```

Histórico de visualizações com HashMap

```
public class HistoricoVisualizacao {
  //...
  public void listaVisualizacoes() {
    if (visualizacoes.isEmpty()) {
        System.out.println("Nenhum item visualizado ainda.");
        return;
    }
    System.out.println("\nHistórico de visualizações:");
    visualizacoes.forEach((item, hora) ->
        System.out.printf(" - '%s' (id %d) em '%s'\n", item.nome(), item.id(), hora));
    System.out.println();
  }
  public void totalItensVisualizados() {
      System.out.printf("Total de itens únicos visualidos: " + visualizacoes.size());
  }
}
```

Usando o Histórico de visualizações

```
HistoricoVisualizacao historico = new HistoricoVisualizacao(database);
historico.registrarVisualizacao(1L); // refresco
historico.registrarVisualizacao(2L); // sanduiche
historico.registrarVisualizacao(6L); // pipoca
historico.registrarVisualizacao(1L); // refresco (de novo)

historico.listaVisualizacoes();
historico.totalItensVisualizados();
```

Mantém apenas a última visualização

```
'Refresco do Chaves' visualizado em '2025-08-03T14:29:51.509318'
'Sanduíche de Presunto do Chaves' visualizado em '2025-08-03T14:29:51.510507'
'Pipoca do Quico' visualizado em '2025-08-03T14:29:51.510578'
'Refresco do Chaves' visualizado em '2025-08-03T14:29:51.510655'

Histórico de visualizações:
 - 'Sanduíche de Presunto do Chaves' (id 2) em '2025-08-03T14:29:51.510507'
 - 'Pipoca do Quico' (id 6) em '2025-08-03T14:29:51.510578'
 - 'Refresco do Chaves' (id 1) em '2025-08-03T14:29:51.510655'

Total de itens únicos visualidos: 3
```

TAREFA

PRECISO REMOVER UM ITEM DE CARDÁPIO.

Removendo item do cardápio

```
public class Database {
  //...
  public boolean removeItemCardapio(Long id) {
      ItemCardapio removido = itensPorId.remove(id);
      return removido != null;
  }
}

long idParaRemover = 1L;
boolean removido = database.removeItemCardapio(idParaRemover); // refresco
System.out.printf("Item %d %s\n", idParaRemover,
          (removido ? "removido" : "não encontrado"));
database.listaItensCardapio()
          .forEach(System.out::println);
```

Na Main

```
Item 1 removido
ItemCardapio[id=2, nome=Sanduíche...]
ItemCardapio[id=5, nome=Torta...]
ItemCardapio[id=6, nome=Pipoca...]
ItemCardapio[id=7, nome=Água...]
ItemCardapio[id=9, nome=Churros...]
```

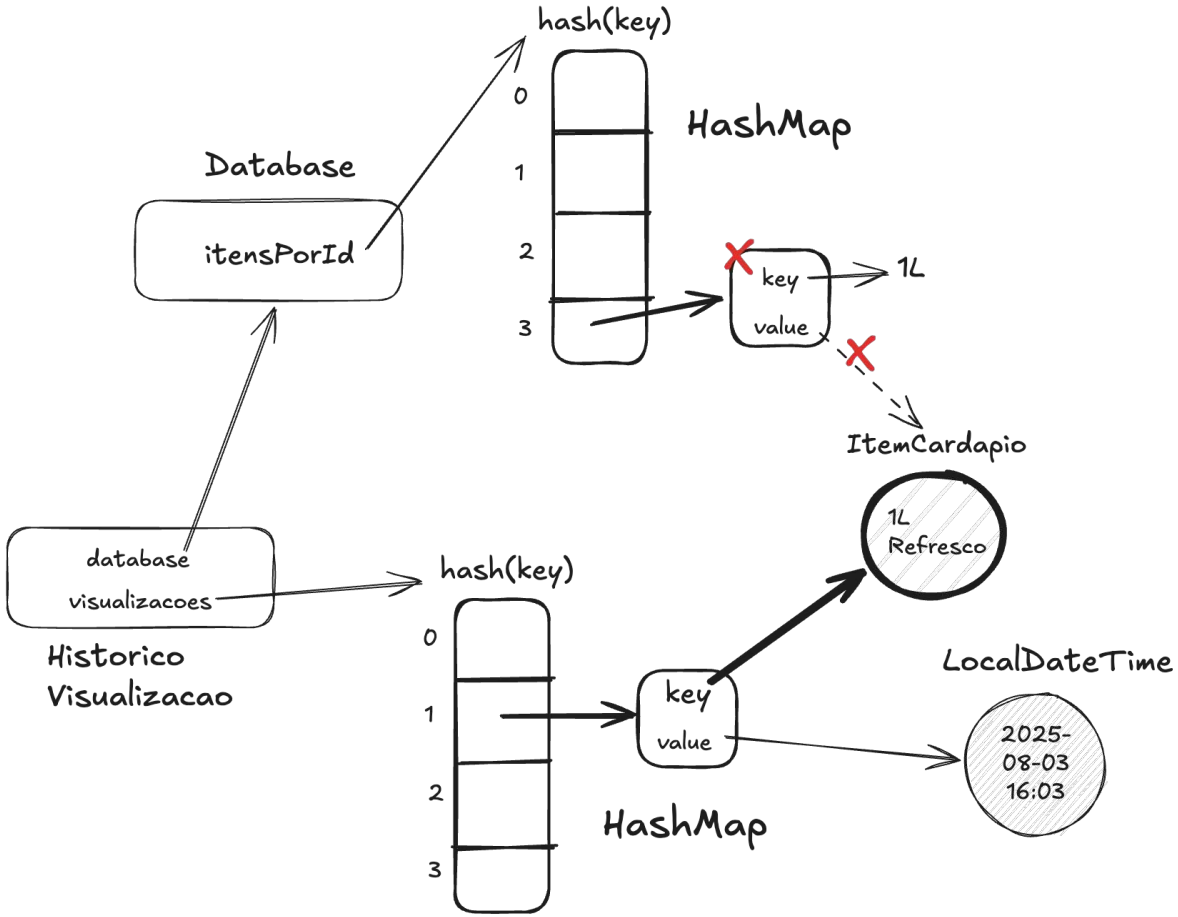
Item removido ainda no histórico

```
System.out.println("Solicitando GC...");
System.gc();
Thread.sleep(500); // tempo para o GC agir


historico.listaVisualizacoes();
historico.totalItensVisualizados();
```

Temtativa de influenciar o GC

Refresco se mantém no histórico mesmo depois de removido

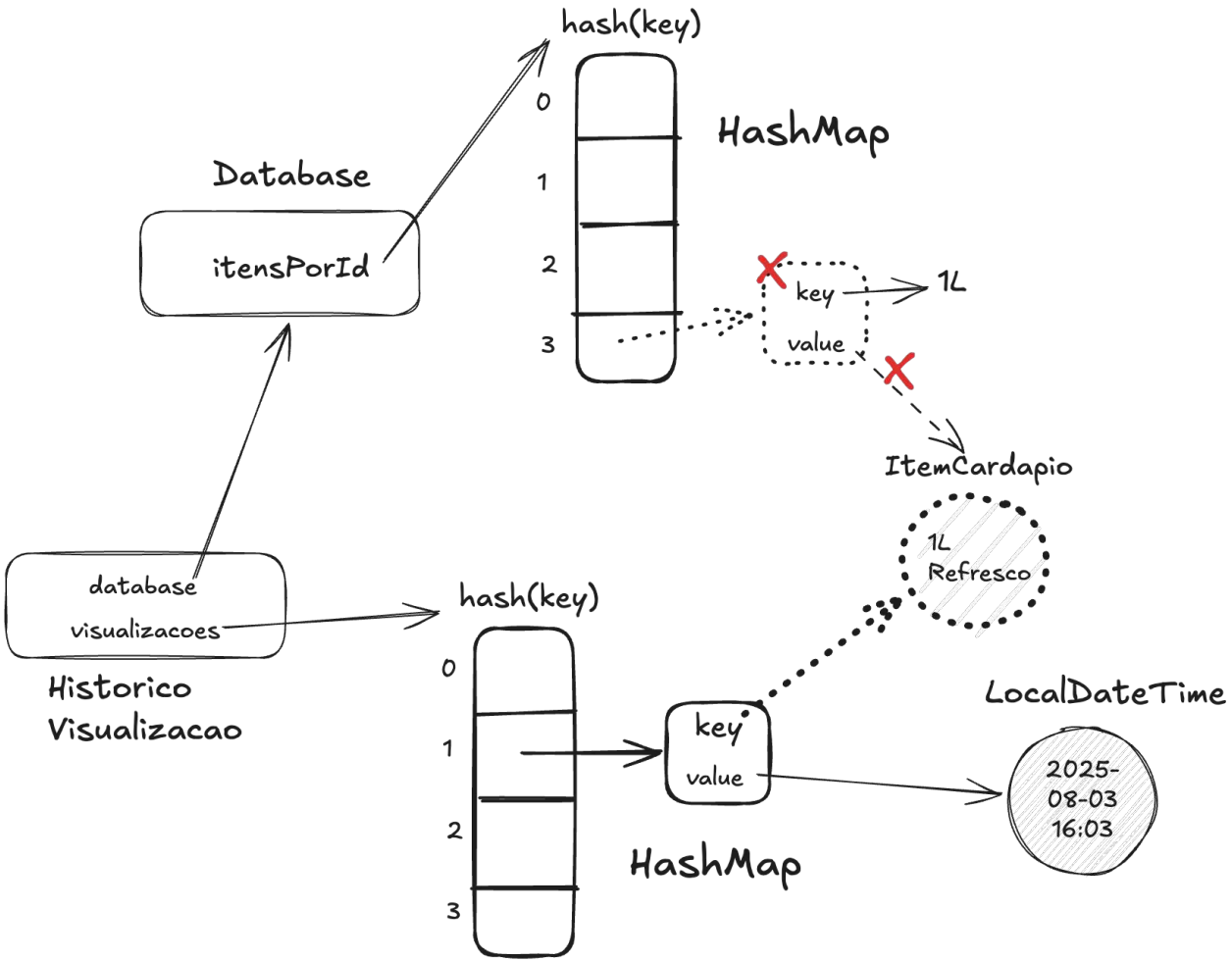
```
Histórico de visualizações:
 - 'Sanduíche de Presunto do Chaves' (id 2) em '2025-08-03T14:29:51.510507'
 - 'Pipoca do Quico' (id 6) em '2025-08-03T14:29:51.510578'
 - 'Refresco do Chaves' (id 1) em '2025-08-03T14:29:51.510655'

Total de itens únicos visualidos: 3
```


hash(key)
0
1
2
3
HashMap
Database
itensPorId
key
value
1L
ItemCardapio
1L
Refresco
database
visualizacoes
Historico
Visualizacao
hash(key)
0
1
2
3
key
value
HashMap
LocalDateTime
2025-
08-03
16:03

WeakHashMap

Histórico de visualizações com WeakHashMap

```
public class HistoricoVisualizacao {

  private Map<ItemCardapio, LocalDateTime> visualizacoes = new WeakHashMap<>();

  //...
}
```


hash(key)
0
1
2
3
HashMap
Database
itensPorId
key
1L
value
ItemCardapio
1L
Refresco
database
visualizacoes
Historico
Visualizacao
hash(key)
0
1
2
3
key
value
HashMap
LocalDateTime
2025-
08-03
16:03

Item removido do histórico

```
System.out.println("Solicitando GC...");
System.gc();
Thread.sleep(500); // tempo para o GC agir


historico.listaVisualizacoes();
historico.totalItensVisualizados();
```

Temtativa de influenciar o GC

Refresco também é removido do histórico!

```
Histórico de visualizações:
 - 'Sanduíche de Presunto do Chaves' (id 2) em '2025-08-03T14:29:51.510507'
 - 'Pipoca do Quico' (id 6) em '2025-08-03T14:29:51.510578'

Total de itens únicos visualidos: 2
```

TAREFA

PRECISO ALTERAR O PREÇO DE UM ITEM DE CARDÁPIO.

Alterando preco no ItemCardapio

```
public record ItemCardapio(Long id, String nome, String descricao,
                           CategoriaCardapio categoria, BigDecimal preco,
                                                BigDecimal precoPromocional) {
  //...

  public ItemCardapio alteraPreco(BigDecimal novoPreco) {
    return new ItemCardapio(id, nome, descricao, categoria,
                                    novoPreco, precoPromocional);
  }

}
```

Atualizando item no Database

```
public class Database {
  //...
  public boolean alteraPrecoItemCardapio(Long id, BigDecimal novoPreco) {
      ItemCardapio item = itensPorId.get(id);
      if (item == null ) {
         return false;
      }
      ItemCardapio itemPrecoAlterado = item.alteraPreco(novoPreco);
      itensPorId.put(id, itemPrecoAlterado);
      return true;
  }
}
```

Usando alteração de preço na Main

```
ItemCardapio item = database.itemCardapioPorId(1L).orElseThrow();
System.out.printf("'%s': R$ %.2f\n", item.nome(), item.preco());

boolean alterado = database.alteraPrecoItemCardapio(1L, new BigDecimal("3.99"));
System.out.printf("%s\n", alterado ? "Preço alterado." : "Não encontrado.");

ItemCardapio itemAlterado = database.itemCardapioPorId(1L).orElseThrow();
System.out.printf("'%s': R$ %.2f\n", itemAlterado.nome(), itemAlterado.preco());
```

```
'Refresco do Chaves': R$ 2.99
Preço alterado.
'Refresco do Chaves': R$ 3.99
```

TAREFA

PRECISO AUDITAR TODA MUDANÇA DE PREÇO DOS ITENS.

Atualizando item no Database

```
public class Database {
  //...
  private final Map<ItemCardapio, BigDecimal> auditoriaPrecos = new HashMap<>();
  //...
  public boolean alteraPrecoItemCardapio(Long id, BigDecimal novoPreco) {
      ItemCardapio item = itensPorId.get(id);
      //...
      auditoriaPrecos.put(item, novoPreco);
      return true;
  }
  public void rastroAuditoriaPrecos() {
     System.out.println("\nAuditoria de preços:");
     auditoriaPrecos.forEach((item, preco) ->
          System.out.printf(" - %s: %s => %s\n", item.nome(), item.preco(), preco));
     System.out.println();
  }
}
```

Item velho, preço novo

Auditando na Main

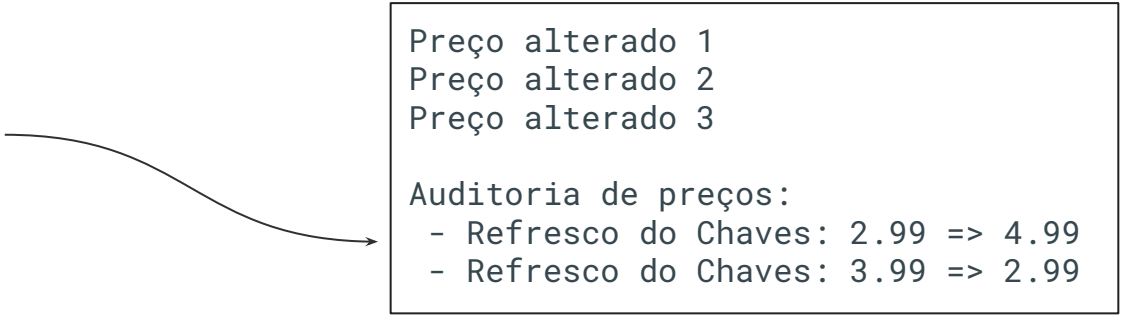
```
database.rastroAuditoriaPrecos();
```

```
'Refresco do Chaves': R$ 2.99
Preço alterado.
'Refresco do Chaves': R$ 3.99

Auditoria de preços:
 - Refresco do Chaves: 2.99 => 3.99
```

Mudando preços na Main

```
boolean alterado1 = database.alteraPrecoItemCardapio(1L, new BigDecimal("3.99"));
System.out.printf("%s\n", alterado1 ? "Preço alterado 1" : "Não encontrado.");

boolean alterado2 = database.alteraPrecoItemCardapio(1L, new BigDecimal("2.99"));
System.out.printf("%s\n", alterado2 ? "Preço alterado 2" : "Não encontrado.");

boolean alterado3 = database.alteraPrecoItemCardapio(1L, new BigDecimal("4.99"));
System.out.printf("%s\n", alterado3 ? "Preço alterado 3" : "Não encontrado.");

database.rastroAuditoriaPrecos();
```

Mas não foram 3 alterações?



IdentityHashMap

Atualizando item no Database

```
public class Database {
  //...
  private final Map<ItemCardapio, BigDecimal> auditoriaPrecos = new IdentityHashMap<>();

}
```


```
Preço alterado 1
Preço alterado 2
Preço alterado 3

Auditoria de preços:
 - Refresco do Chaves: 3.99 => 2.99
 - Refresco do Chaves: 2.99 => 4.99
 - Refresco do Chaves: 2.99 => 3.99
```

Mas como isso funciona?

```
ItemCardapio item = database.itemCardapioPorId(1L).orElseThrow(); // 2.99

database.alteraPrecoItemCardapio(1L, new BigDecimal("3.99")); // 2.99 => 3.99
ItemCardapio item1 = database.itemCardapioPorId(1L).orElseThrow(); // 3.99

database.alteraPrecoItemCardapio(1L, new BigDecimal("2.99")); // 3.99 => 2.99
ItemCardapio item2 = database.itemCardapioPorId(1L).orElseThrow(); // 2.99

database.alteraPrecoItemCardapio(1L, new BigDecimal("4.99")); // 2.99 => 4.99
ItemCardapio item3 = database.itemCardapioPorId(1L).orElseThrow(); // 4.99

System.out.println("== " + (item == item2));
System.out.println("equals() " + (item.equals(item2)));
System.out.println("hashCode() " + (item.hashCode() == item2.hashCode()));
```

```
== false
equals() true
hashCode() true
```

https://github.com/unipds-projetos/modulo1-fundamentos-java-extra-cardapio/tree/aula2-aprofundando-em-colecoes