

JAVA COLLECTIONS

Z

DOMINE COMO UM
SAYAJIN



Cintia Silva

CAPÍTULO 1

List - “O primeiro passo do guerreiro Z”

Ei! Antes de se transformar em Super Sayajin no mundo Java, você precisa dominar as Listas! Elas são como os primeiros treinos com o Mestre Kame: simples, mas essenciais.



Quando usar?



Use quando quiser armazenar itens em ordem e acessar por posição. Tipo uma fila de guerreiros esperando sua vez no Torneio do Poder.

Exemplo:

```
● ● ● Torneio.java

import java.util.ArrayList;
import java.util.List;

public class Torneio {
    public static void main(String[] args) {
        List<String> guerreiros = new ArrayList<>();
        guerreiros.add("Goku");
        guerreiros.add("Vegeta");
        guerreiros.add("Gohan");

        for (String guerreiro : guerreiros) {
            System.out.println("Pronto para a luta: " + guerreiro);
        }
    }
}
```

CAPÍTULO 2

Set - "Sem Duplicatas, Como uma Fusão Perfeita"

Um Set é como uma fusão bem feita: cada elemento é único! Se tentar colocar o mesmo guerreiro duas vezes, ele só entra uma vez no campo de batalha.



Quando usar?



Quando não quiser elementos repetidos. É perfeito pra guardar as Esferas do Dragão — só pode ter uma de cada!

Exemplo:

```
••• Esferas.java

import java.util.HashSet;
import java.util.Set;

public class Esferas {
    public static void main(String[] args) {
        Set<Integer> esferas = new HashSet<>();
        esferas.add(1);
        esferas.add(2);
        esferas.add(3);
        esferas.add(1); // Tentar repetir não funciona!

        for (int esfera : esferas) {
            System.out.println("Esfera do Dragão número: " + esfera);
        }
    }
}
```

CAPÍTULO 3

Map - "Chave + Valor = Combição Lendária"

Map é tipo o Radar do Dragão: você procura algo (chave) e recebe a informação (valor). Perfeito pra guardar o poder de luta dos guerreiros Z!



Quando usar?



Quando precisar associar um valor a uma chave, tipo nome do guerreiro e seu poder de luta.

Exemplo:



NivelDePoder.java

```
import java.util.HashMap;
import java.util.Map;

public class NivelDePoder {
    public static void main(String[] args) {
        Map<String, Integer> poderes = new HashMap<>();
        poderes.put("Goku", 9000);
        poderes.put("Vegeta", 8500);

        System.out.println("Poder do Goku: " + poderes.get("Goku"));
    }
}
```

CAPÍTULO 4

Os Mestres das Collections – Classes Lendárias em Campo!

Quando você domina as interfaces, ainda falta aprender com os mestres lendários: as classes que implementam essas interfaces e fazem a mágica acontecer na prática! Bora conhecer quem realmente entra na arena?



ArrayList - O Velocista da Ordem



ArrayList é o guerreiro padrão do List. Ele é rápido pra acessar elementos (como mirar um Kamehameha) e ótimo quando você precisa guardar itens em ordem.

Exemplo:

```
TecnicasZ.java

import java.util.ArrayList;
import java.util.List;

public class TecnicasZ {
    public static void main(String[] args) {
        List<String> tecnicas = new ArrayList<>();
        tecnicas.add("Kamehameha");
        tecnicas.add("Kaioken");

        System.out.println(tecnicas.get(0)); // Kamehameha
    }
}
```

💡 Use quando: quiser acessar por índice ou iterar rápido.
⚠️ Cuidado: Inserções no meio da lista podem ser lentas.

LinkedList - O Ninja da Flexibilidade



LinkedList também implementa List, mas é bom em inserir e remover elementos, especialmente no começo ou meio da lista.

Exemplo:

```
MissaoZ.java

import java.util.LinkedList;
import java.util.List;

public class MissaoZ {
    public static void main(String[] args) {
        List<String> fila = new LinkedList<>();
        fila.add("Kuririn");
        fila.add(0, "Goku"); // Insere no início

        for (String nome : fila) {
            System.out.println(nome);
        }
    }
}
```

- 💡 Use quando: inserir/remover muito no início/fim.
- ⚠️ Acesso por índice é mais lento comparado ao ArrayList.

HashSet - O Guardião das Esferas



HashSet é a escolha número 1 pra quem quer um Set que não aceita duplicatas e não se importa com a ordem.

Exemplo:

```
InventarioZ.java

import java.util.HashSet;
import java.util.Set;

public class InventarioZ {
    public static void main(String[] args) {
        Set<String> itens = new HashSet<>();
        itens.add("Esfera de 1 estrela");
        itens.add("Esfera de 2 estrelas");
        itens.add("Esfera de 1 estrela"); // Ignorado!
        itens.forEach(System.out::println);
    }
}
```

💡 Use quando: quiser evitar duplicações com rapidez.
⚠️ A ordem dos elementos não é garantida.

TreeSet - O Estrategista da Ordem



TreeSet mantém os itens em ordem natural (alfabética ou numérica).

Exemplo:

```
RankingZ.java

import java.util.Set;
import java.util.TreeSet;

public class RankingZ {
    public static void main(String[] args) {
        Set<String> guerreiros = new TreeSet<>();
        guerreiros.add("Vegeta");
        guerreiros.add("Goku");
        guerreiros.add("Gohan");

        guerreiros.forEach(System.out::println); // Ordem alfabética
    }
}
```

💡 Use quando: precisar dos itens ordenados automaticamente.

⚠️ Um pouco mais lento que o HashSet.

HashMap - O Radar do Poder



O mestre do Map! Com ele, você busca um valor por uma chave, tipo saber o poder de luta de cada guerreiro.

Exemplo:

```
PoderZ.java

import java.util.HashMap;
import java.util.Map;

public class PoderZ {
    public static void main(String[] args) {
        Map<String, Integer> poderes = new HashMap<>();
        poderes.put("Goku", 9001);
        poderes.put("Vegeta", 8500);

        System.out.println("Goku tem: " + poderes.get("Goku") + " de poder!");
    }
}
```

💡 Use quando: precisar associar chave → valor com rapidez.

⚠️ A ordem dos pares não é garantida.

TreeMap - O Arquivista Intergaláctico



Quer os pares chave-valor em ordem pelas chaves?
Chame o TreeMap!

Exemplo:

```
ClassificacaoZ.java

import java.util.Map;
import java.util.TreeMap;

public class ClassificacaoZ {
    public static void main(String[] args) {
        Map<String, Integer> ranking = new TreeMap<>();
        ranking.put("Gohan", 7500);
        ranking.put("Goku", 9001);
        ranking.put("Vegeta", 8500);

        ranking.forEach((nome, poder) → System.out.println(nome + ": " + poder));
    }
}
```

💡 Use quando: quiser ordenar suas chaves automaticamente.

⚠ Mais lento que HashMap, mas ordenado.

Finalizando o CapítuloMap - O Arquivista Intergaláctico



Cada interface tem seus guerreiros de elite. E como no Torneio do Poder, não existe o melhor absoluto, e sim o mais adequado pra cada luta.

Interface	Classe Principal	Força Principal
List	ArrayList / LinkedList	Ordem e acesso por índice
Set	HashSet / TreeSet	Evitar duplicatas
Map	HashMap / TreeMap	Chave → Valor

CAPÍTULO 5

A Saga das Collections - "Do Fraco ao Lendário"



⌚ Saga 1: ArrayList em Missão de Treinamento



TreinamentoBasico.java

```
import java.util.ArrayList;
import java.util.List;

public class TreinamentoBasico {
    public static void main(String[] args) {
        List<String> tecnicas = new ArrayList<>();
        tecnicas.add("Kamehameha");
        tecnicas.add("Kaioken");
        tecnicas.add("Genki Dama");

        for (String tecnica : tecnicas) {
            System.out.println("Técnica aprendida: " + tecnica);
        }
    }
}
```

Use ArrayList quando quiser guardar elementos em ordem, como suas técnicas favoritas para a próxima batalha!

🌀 Saga 2: Set na Proteção das Esferas



Inventario.java

```
import java.util.HashSet;
import java.util.Set;

public class Inventario {
    public static void main(String[] args) {
        Set<String> itens = new HashSet<>();
        itens.add("Esfera de 1 estrela");
        itens.add("Esfera de 2 estrelas");
        itens.add("Esfera de 1 estrela"); // Duplicada!

        for (String item : itens) {
            System.out.println("Item no inventário: " + item);
        }
    }
}
```

Use Set quando não quiser duplicatas. Só pode ter uma esfera de cada, lembra?

🌀 Saga 3: Map para Monitorar Níveis de Poder



CentralDeEnergia.java

```
import java.util.HashMap;
import java.util.Map;

public class CentralDeEnergia {
    public static void main(String[] args) {
        Map<String, Integer> nivelDePoder = new HashMap<>();
        nivelDePoder.put("Trunks", 7000);
        nivelDePoder.put("Piccolo", 6500);

        for (String nome : nivelDePoder.keySet()) {
            System.out.println(nome + " tem poder de luta: " + nivelDePoder.get(nome));
        }
    }
}
```

Use Map quando quiser ligar uma chave a um valor. Perfeito pra saber quem é o próximo a liberar todo o poder!

⌚ Saga 4: Queue na Fila do Torneio



```
... TorneioDoPoder.java ...  
  
import java.util.LinkedList;  
import java.util.Queue;  
  
public class TorneioDoPoder {  
    public static void main(String[] args) {  
        Queue<String> filaDeGuerreiros = new LinkedList<>();  
        filaDeGuerreiros.add("Goku");  
        filaDeGuerreiros.add("Vegeta");  
        filaDeGuerreiros.add("Android 18");  
  
        while (!filaDeGuerreiros.isEmpty()) {  
            System.out.println("Próximo lutador: " + filaDeGuerreiros.poll());  
        }  
    }  
}
```

Use Queue quando a ordem de entrada for importante. O primeiro a entrar é o primeiro a lutar, como no Torneio do Poder!

⌚ Saga 5: TreeMap para Ordem no Torneio Intergaláctico



```
... • • ...  
OrdemDoTorneio.java  
  
import java.util.Map;  
import java.util.TreeMap;  
  
public class OrdemDoTorneio {  
    public static void main(String[] args) {  
        Map<String, Integer> guerreiros = new TreeMap<>();  
        guerreiros.put("Vegeta", 8500);  
        guerreiros.put("Goku", 9800);  
        guerreiros.put("Gohan", 7000);  
  
        for (String nome : guerreiros.keySet()) {  
            System.out.println(nome + " está com nível: " + guerreiros.get(nome));  
        }  
    }  
}
```

Use TreeMap quando quiser organizar suas chaves em ordem alfabética ou natural. Ideal para deixar tudo nos trinques antes da luta começar!

Saga 6: Stack para Técnicas em Combo



ComboDeTecnicas.java

```
import java.util.Stack;

public class ComboDeTecnicas {
    public static void main(String[] args) {
        Stack<String> tecnicas = new Stack<>();
        tecnicas.push("Kamehameha");
        tecnicas.push("Teletransporte");
        tecnicas.push("Genki Dama");

        while (!tecnicas.isEmpty()) {
            System.out.println("Usando técnica: " + tecnicas.pop());
        }
    }
}
```

Use Stack quando quiser usar os últimos elementos primeiro. É tipo aquele combo surpresa que você guardou pro final da batalha!

Fusão das Collections: Quando os Guerreiros se Unem!



BaseDeDadosDoGuerreiroZ.java

```
import java.util.*;  
  
public class BaseDeDadosDoGuerreiroZ {  
    public static void main(String[] args) {  
        List<String> listaDeGuerreiros = new ArrayList<>(List.of("Goku", "Vegeta", "Gohan"));  
        Set<String> tecnicasUnicas = new HashSet<>(Set.of("Kamehameha", "Kaioken", "Genki Dama"));  
        Map<String, Integer> poderDeLuta = new HashMap<>();  
        poderDeLuta.put("Goku", 9000);  
        poderDeLuta.put("Vegeta", 8500);  
        Stack<String> ultimosAtaques = new Stack<>();  
        ultimosAtaques.push("Soco Rápido");  
        ultimosAtaques.push("Chute Aéreo");  
  
        System.out.println("---- Guerreiros ----");  
        listaDeGuerreiros.forEach(System.out::println);  
  
        System.out.println("\n---- Técnicas Únicas ----");  
        tecnicasUnicas.forEach(System.out::println);  
  
        System.out.println("\n---- Poder de Luta ----");  
        poderDeLuta.forEach((g, p) → System.out.println(g + ": " + p));  
  
        System.out.println("\n---- Últimos Ataques ----");  
        while (!ultimosAtaques.isEmpty()) {  
            System.out.println(ultimosAtaques.pop());  
        }  
    }  
}
```

Essa é a verdadeira fusão: usar List, Set, Map e Stack juntas! Cada uma com seu papel numa batalha épica contra o caos do código.

CAPÍTULO 6

Dica Final do Goku

Treinar Collections é como treinar o Ki: você precisa de foco, paciência e prática. Comece com o List, depois experimente Set, e quando estiver pronto, domine o Map. Cada estrutura tem um momento certo na batalha do código! Continue treinando, jovem guerreiro Z do Java! 

