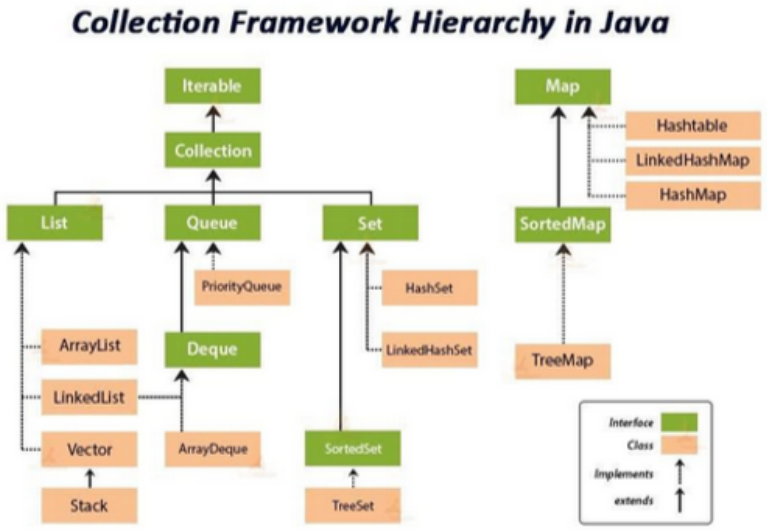
Coleções

- Um conjunto de comandos, entidades e pacotes para programar aplicações.

Java tem uma **API** própria para tratar com coleções .

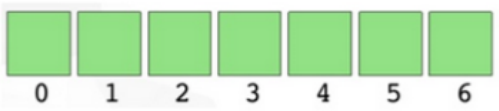
API (Application Programming Interface): Entidades, pacotes e outras definições para auxiliar na programação de aplicações.



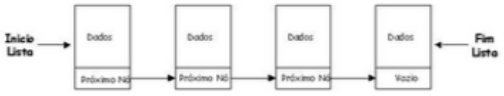
Estruturas de dados

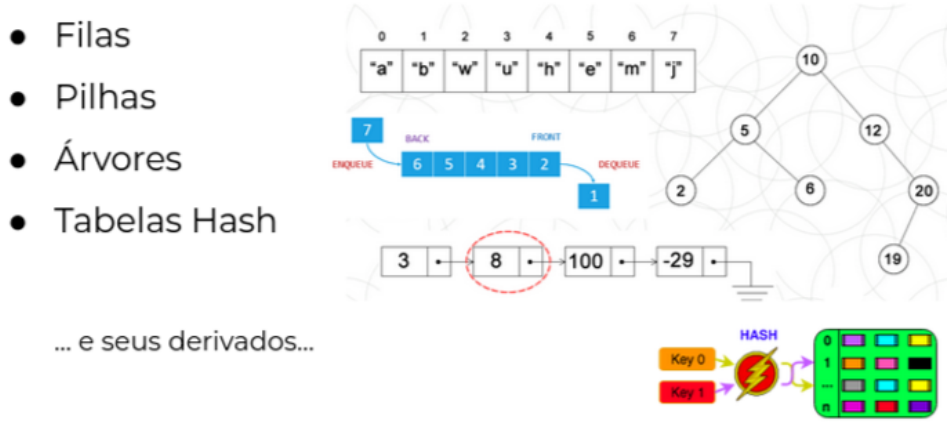
Definem a organização, métodos de acesso e opções de processamento para a informação manipulada pelo programa.

- Arrays



- Listas





- Arrays: estruturas sequenciais de armazenamento estático;
- Listas: estruturas sequenciais de armazenamento dinâmico;
- Pilhas: estruturas sequenciais, princípio LIFO;
- Filas: estruturas sequenciais, princípio FIFO;
- Árvores: estrutura hierárquica;
- Tabelas Hash: estrutura indexada por código hash, ideia de chave/valor.

Estruturas Básicas de JAVA e suas características

- Mapas:
  - Associam chave(Key) a um valor (Value);
  - Não tem ordem posicional;
  - Aceita duplicatas de elementos, desde que suas chaves sejam diferentes.
- Listas e Conjuntos: Ambos são uma coleção de objetos;
  - Adicionar e remover elemento;
  - Verificar número de elementos;
  - Verificar se é vazio;
  - Verificar tamanho;
  - Verificar se tal elemento existe.

- Listas:
  - Pode ter elementos repetidos;
  - Elementos tem ordem posicional;
  - Acesso por posição (índice) na lista;
- Conjuntos:
  - Não admite elementos repetidos;
  - Não tem odem posicional;
  - Elementos não são acessíveis por posição (índice).

```
ArrayList<Integer> lista = new ArrayList<>();
HashSet<Integer> conjunto = new HashSet<>();

lista.add(new Integer(1));
lista.add(new Integer(2));
lista.add(new Integer(1));
conjunto.add(new Integer(1));
conjunto.add(new Integer(2));
conjunto.add(new Integer(1));

System.out.println(lista.size()); // 3
System.out.println(conjunto.size()); // 2
```

ArrayList

- Implementa a interface List;
- Métodos prontos para a manipulação do array;
- Capacidade aumenta dinamicamente;
- Use quando desejar guardar a ordem de inserção dos elementos.

Criando uma lista:

```
package ArrayList;

import java.util.ArrayList; //mantém um array interno para ar

public class TestandoArrayList{

    public static viod main(String[] args){
        String cliente1 = "Antonio Carlos";
        String cliente2 = "José Cláudio";
        String cliente3 = "Norma Maria";

        ArrayList <String> cadastro = new ArrayList<String>()

        cadastro.add(cliente1); //Esse método realiza a adiçã
        cadastro.add(cliente2); //add(ind index, Element e):
        cadastro.add(cliente3);

        cadastro.remove(0); // remove(int index): remove o ob

        for (int i=0; i<cadastro.size(); i++){
            System.out.println("Player: "+cadastro.get(i)); /
        }

        // for(elemento) percorre a lista, elemento por eleme
        for (String c: cadastro){
            System.out.println("1 Cliente: " +c);
        }

        //for (inteiro) percorre a lista, usando variável de
        for(int i=0; i<cadsastro.size(); i++){
            System.out.println("2 Cliente: "+ cadastro.get(i)
        }

        //cadastro.forEach() percorre a lista e realiza uma a
        cadastro.forEach(c -> {System.out.println("3 Cliente:

        System.out.println(cadastro);
```

```
    }
}
```

HashSet

Implementação padrão de conjunto. Bastante eficiente para adicionar e localizar elementos.

Use quando desejar armazenar elementos sem repetição.

Internamente ele reserva um espaço na memória (array) para os objetos de acordo com o hashcode.

Para localizar um objeto, olhamos se ele está na posição indicada pelo hashcode (ultra rápido).

- Problema 1: Objetos diferentes podem tem o mesmo hashcode.

  Como solução, no lugar de cada posição armazenar um objeto, ele armazena todos os objetos que tem o mesmo hashcode. Dessa coleção de objetos, usamos o equals como um critério de desempate.

- Problema 2: Alocar um array de tamanho 2^{31}é custoso (i.e. para todos os hashCodes);

  Como solução, usamos o módulo do hashcode para a localização dos elementos.

- **Operações Básicas:**

Método	Descrição
add(Objeto)	Adiciona um elemento ao nosso Set, retorna true se o elemento foi inserido
remove(Objeto)	Remove o elemento da nossa coleção, retorna true se o elemento foi removido
iterator()	Retorna um objeto do tipo Iterator
size()	Retorna um int (inteiro) com a quantidade de elementos da coleção
contains(Objeto)	Retorna true se o elemento já existe dentro do Set
clear()	Elimina todos os elementos do Set

- **Operações com HashSet:**

```
private HashSet<BolaDeGude> bolinhas = new HashSet<>();

public boolean adicionaElemento(BolaDeGude elemento){
    return bolinhas.add(elemento);
}

public boolean pertence(BolaDeGude elemento){
    return bolinhas.contains(elemento);
}

public boolean remove(BolaDeGude elemento){
    return bolinhas.remove(elemento);
}

public int tamanho(){
    return bolinhas.size();
}
```

HashMap

Implementação padrão de mapas, bastante eficiente para trabalhar com chaves.

Use quando quiser associar um objeto a outro, i.e. a identificação de um objeto por outro.

- **Operações Básicas:**

Método	Descrição
put(chave, objeto)	Adiciona a chave e o elemento
get(chave)	Retorna o elemento de acordo com a chave. Se a chave não existir, retorna null. Se a chave existir, e mapear para um objeto null também.
iterator()	Retorna um objeto do tipo iterator

Método	Descrição
size()	Retorna um int (inteiro) com a quantidade de elementos da coleção
containsKey(chave)	Retorna true se a chave já existe
clear()	Elimina todos os elementos do Set
containsValue(objeto)	Retorna true se o objeto já existir no Map

- **Operações com HashMap:**

```
public class Turma {

    private HashMap<String, Aluno> mapaMatriculaAlunos = n

    public Aluno adicionaAluno(String matricula, Aluno alu
        return this.mapaMatriculaAlunos.put(matricula, alu
        // retorna o aluno anteriormente associado a essa
    }

    public boolean existeAluno(String matricula){
        return this.mapaMatriculaAlunos.containsKey(matric
    }

    public boolean existeAluno(Aluno aluno){
        return this.mapaMatriculaAlunos.containsValue(alun
    }

    public Aluno recuperaAluno(String matricula){
        return this.mapaMatriculaAlunos.get(matricula);
    }

    public Aluno remove(String matricula){
        return this.mapaMatriculaAlunos.remove(matricula);
    }

    public int numeroDeAlunos(){
```

```
        return this.mapaMatriculaAlunos.size();]
    }
```