Aula 06

Tópicos adicionais de programação

Rogério Pereira Junior

rogeriopereirajunior@gmail.com

15 de maio de 2024

SENAC
Santa Catarina
Florianópolis

Agenda

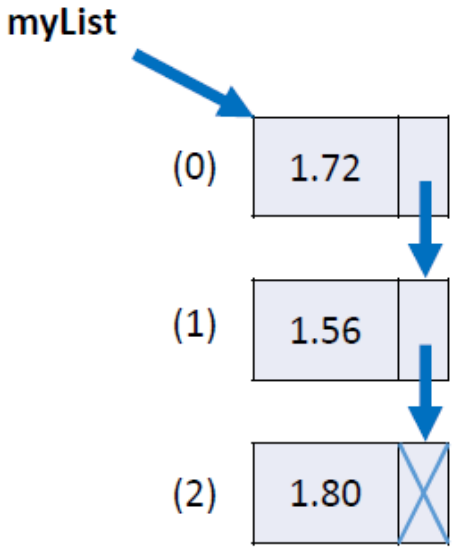
1. Listas

2. Trabalhando com datas

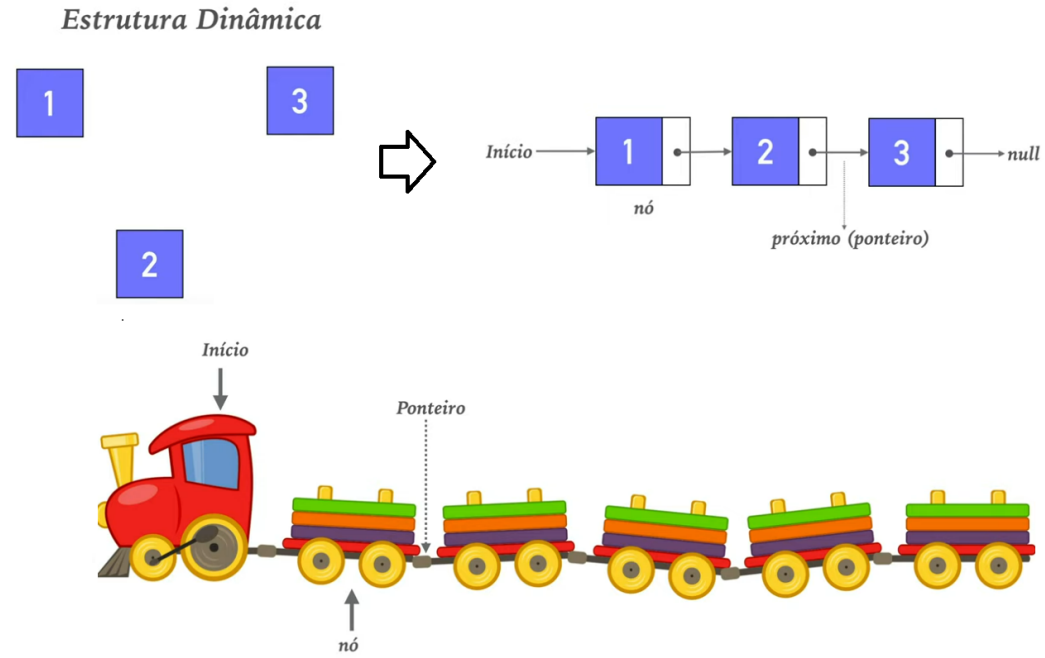
Listas

Lista encadeada

- Lista é uma estrutura de dados:
  - Homogênea (dados do mesmo tipo)
  - Ordenada (elementos acessados por meio de posições)
  - Inicia vazia, e seus elementos são alocados sob demanda
  - Cada elemento ocupa um "nó"(ou nodo) da lista
  - Classe que implementa: **ArrayList**.
  - Vantagens: Tamanho variável e facilidade para se realizar inserções e deleções



Lista encadeada



Lista encadeada

```java
// Inicializando uma lista vazia
List<String> lista = new ArrayList<>();

// Inicializando uma lista com elementos
List<Integer> listaComElementos = new ArrayList<>(List.of(1, 2, 3, 4, 5));

// Inicializando uma lista com capacidade inicial
List<Double> listaComCapacidade = new ArrayList<>(10); // Capacidade inicial de 10 elementos

// Adicionando valores
lista.add("Maria");
lista.add("Joao");
lista.add("Fabio");

// Obtendo o conteudo da posição 1
String nome = lista.get(1);
System.out.println("Elemento na posição 1: " + nome);

for (int i = 0; i < lista.size() ; i++) {
      System.out.println(lista.get(i));
    }

// For each
for (String nome : lista) {
      System.out.println(nome);
    }

// Inserindo todos os elementos da lista1 na lista2
lista2.addAll(lista1);
```

Lista - Outros métodos importantes de uma lista

- Remoção

```
//Removendo pelo elemento
lista.remove("Maria");

//Removendo pela posição (indice)
lista.remove(0);

// Removendo o elemento com valor 20 (Caso a lista seja de inteiros!)
lista.remove(Integer.valueOf(20));

// Remove o elemento com valor 10
lista.removeIf(i -> i.charAt(0) == 'M'); // Remove da lista os elementos cujo primeira letra é M

// Remover primeiro ou ultime elemento
lista.removeFirst();
lista.removeLast();
```

- Adição em uma posição especifica

```
lista.add(0, "Maria"); // Adicionar (append) em uma posição especifica (posicao 0)
// Adicionando no começo da lista
lista.addFirst("Julio");
```

- Tamanho da lista: size()

```
lista.size()
```

Lista encadeada - Outros métodos

- Modificar um elemento

```
// Modificando o elemento na posição 1 (segundo elemento)
lista.set(1, "Novo Elemento 2");
```

- Limpando todos os elementos da lista

```
lista.clear();
```

- Ordenação

```
lista.sort(null); // Ordena a lista em ordem crescente
System.out.println(  lista.reversed()); // Reverte os elementos
```

- Verificar posição

```
boolean contem = lista.contains(10);  verifica se a lista contém o elemento 10
int indice = lista.indexOf(20); // encontrar a posição de um elemento
int ultimoIndice = lista.lastIndexOf(20); // Retorna a posiçãoda última ocorrência de um elemento
```

- Verificando se a lista está vazia

```
boolean vazia = lista.isEmpty();
System.out.println("A lista está vazia? " + vazia);
```

Lista encadeada

- Criar uma lista com elementos de uma outra lista

```
List<String> lista = new ArrayList<>();
lista.add("Maçã");
lista.add("Banana");
lista.add("Abacaxi");
lista.add("Laranja");
lista.add("Pera");

// Obtendo uma sublista da lista original
List<String> sublista = lista.subList(1, 4); // Elementos do índice 1 ao 3

// Exibindo a sublista
System.out.println("Sublista: " + sublista);
```

Exemplo

Vamos criar uma lista de números e executar algumas operações, como adicionar elementos, remover elementos, calcular a média e encontrar o maior valor.

Lista encadeada

```
List<Integer> lista = new ArrayList<>();
// Adicionar elementos
lista.add(1);
lista.add(2);
lista.add(3);
lista.add(3,55); // Adiciona o elemento 55 na posição 3
for (int i = 10; i < 16; i++) {
    lista.add(i);
}
System.out.println(lista);
// Remover elementos
lista.remove(2); // Remove o elemento da posição 2
lista.remove(Integer.valueOf(2));  // Remove o elemento de valor 2
lista.removeIf(i -> i > 50); // Removendo os valores maiores que 50
System.out.println(lista);
// Calcular a média
double soma = 0;
for (int numero : lista) {
    soma += numero;
}
double media = soma/lista.size();
System.out.println("Média é = " + media);
int maiorValor = lista.get(0);
for (int numero : lista) {
    if (numero > maiorValor ) {
        maiorValor = numero;
    }
}
System.out.println("Maior valor da lista é = " + maiorValor);
```

Exercícios

- Faça um programa que leia duas listas e que gere uma terceira com os elementos das duas primeiras
- Faça um programa que percorra duas listas e gera uma terceira sem elementos repetidos.

Resumo de métodos

1. Adição de elementos:
   - add(E element): Adiciona o elemento especificado no final da lista.
   - add(int index, E element): Adiciona o elemento especificado na posição index.
2. Acesso a elementos:
   - get(int index): Retorna o elemento na posição index.
3. Remoção de elementos:
   - remove(int index): Remove o elemento na posição index e reorganiza a lista.
   - remove(Object obj): Remove a primeira ocorrência do objeto especificado.
4. Tamanho da lista:
   - size(): Retorna o número de elementos na lista.
5. Verificação de existência:
   - contains(Object obj): Verifica se a lista contém o objeto especificado.
6. Índice de um elemento:
   - indexOf(Object obj): Retorna o índice da primeira ocorrência do objeto especificado.
   - lastIndexOf(Object obj): Retorna o índice da última ocorrência do objeto especificado.
7. Sublista:
   - subList(int fromIndex, int toIndex): Retorna uma sublista dos elementos da posição fromIndex até toIndex.

Vetor de lista

```java
// Criando um vetor de listas
List<Integer>[] vetorDeListas = new ArrayList[5]; // Vetor com 5 listas

// Inicializando as listas
for (int i = 0; i < vetorDeListas.length; i++) {
    vetorDeListas[i] = new ArrayList<>();
}
// Adicionando elementos às listas
vetorDeListas[0].add(1);
vetorDeListas[0].add(2);
vetorDeListas[1].add(3);
vetorDeListas[1].add(4);
vetorDeListas[1].add(5);
vetorDeListas[2].add(6);
vetorDeListas[3].add(7);
vetorDeListas[3].add(8);
vetorDeListas[3].add(9);
vetorDeListas[4].add(10);

// Exibindo os elementos das listas
for (int i = 0; i < vetorDeListas.length; i++) {
    System.out.print("Lista " + i + ": ");
    for (int j = 0; j < vetorDeListas[i].size(); j++) {
        System.out.print(vetorDeListas[i].get(j) + " ");
    }
    System.out.println();
}
```

Trabalhando com datas

Trabalhando com datas

```
LocalDate hoje = LocalDate.now(); // Data atual
LocalDate dataEspecifica = LocalDate.of(2023, 12, 31); // Data específica: 31 de dezembro de 2023

System.out.println(hoje);

System.out.println(dataEspecifica);


LocalTime agora = LocalTime.now(); // Horário atual
LocalTime horaEspecifica = LocalTime.of(14, 30, 0); // Horário específico: 14:30:00
System.out.println(agora);
System.out.println(horaEspecifica);


LocalDateTime agora2 = LocalDateTime.now(); // Data e hora atual
LocalDateTime dataHoraEspecifica = LocalDateTime.of(2023, 12, 31, 14, 30, 0); // 31 de dezembro de
     2023, 14:30:00

System.out.println(agora2);
System.out.println(dataHoraEspecifica);
```

Trabalhando com datas

String para LocalDate

```
// String que representa a data
String dataString = "2024-01-18";

// Define o formato da String
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

// Faz o parsing da String para um objeto LocalDate usando o formatter
LocalDate data2 = LocalDate.parse(dataString, formatter);

// Imprime a data formatada
System.out.println("Data formatada: " + data2);
```

Trabalhando com datas

Transformando Data para String

```
// Obtém a data e hora atual
LocalDateTime agora = LocalDateTime.now();

// Define o formato desejado
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");

// Formata a data e hora como uma string
String dataHoraFormatada = agora.format(formatter);

// Imprime a data e hora formatada
System.out.println("Data e hora formatadas: " + dataHoraFormatada);
```

Trabalhando com datas

pacote sql

```
String dataString = "2024-04-21"; // String representando a data
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
java.util.Date utilDate = sdf.parse(dataString); // Convertendo a string para java.util.Date
java.sql.Date sqlDate = new java.sql.Date(utilDate.getTime()); // Convertendo para java.sql.Date
```

```
 // Obtendo a data atual como LocalDate
LocalDate localDate = LocalDate.now();

// Convertendo LocalDate para java.sql.Date
Date sqlDate = Date.valueOf(localDate);
```

```
// Obtendo a hora atual como LocalTime
LocalTime localTime = LocalTime.now();

// Convertendo LocalTime para java.sql.Time
Time sqlTime = Time.valueOf(localTime);

System.out.println("LocalTime: " + localTime);
System.out.println("java.sql.Time: " + sqlTime);
```

Trabalhando com datas

string para localtime

```
String horaString = "10:30"; // String representando a hora

// Convertendo a string para LocalTime
LocalTime localTime = LocalTime.parse(horaString);
```

localtime para string

```
// Criando um objeto LocalTime
LocalTime localTime = LocalTime.of(10, 30, 0); // Exemplo de hora: 10:30

// Convertendo para string usando toString()
String horaString = localTime.toString();

System.out.println("Hora em string: " + horaString);


// USANDO FORMATER
// Criando um objeto LocalTime
LocalTime localTime = LocalTime.of(10, 30, 0); // Exemplo de hora: 10:30

// Definindo o formato desejado
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm");

// Convertendo para string usando o formatter
String horaString = localTime.format(formatter);
```

Trabalhando com datas

```java
import java.time.LocalDate;
import java.time.Period;

public class CalculadoraMesesEntreDatas {

    public static void main(String[] args) {
        // Defina suas duas datas
        LocalDate dataInicio = LocalDate.of(2022, 1, 1);
        LocalDate dataFim = LocalDate.of(2024, 1, 1);

        // Calcule a diferença em meses
        int mesesDeDiferenca = calcularMesesEntreDatas(dataInicio, dataFim);

        // Exiba o resultado
        System.out.println("Número de meses na diferença: " + mesesDeDiferenca);
    }

    private static int calcularMesesEntreDatas(LocalDate dataInicio, LocalDate dataFim) {
        // Calcule a diferença entre as datas
        Period periodo = Period.between(dataInicio, dataFim);

        // Extraia o número de meses da diferença
        int mesesDeDiferenca = periodo.getYears() * 12 + periodo.getMonths();

        return mesesDeDiferenca;
    }
}
```

calendar

```java
// Obtendo uma instância de Calendar
Calendar calendar = Calendar.getInstance();

// Obtendo o ano, mês, dia, hora, minuto e segundo atuais
int ano = calendar.get(Calendar.YEAR);
int mes = calendar.get(Calendar.MONTH) + 1; // Janeiro é 0, então adicionamos 1
int dia = calendar.get(Calendar.DAY_OF_MONTH);
int hora = calendar.get(Calendar.HOUR_OF_DAY);
int minuto = calendar.get(Calendar.MINUTE);
int segundo = calendar.get(Calendar.SECOND);

// Imprimindo a data e hora atuais
System.out.println("Data e hora atuais:");
System.out.printf("%02d/%02d/%d %02d:%02d:%02d%n", dia, mes, ano, hora, minuto, segundo);
```

calendar

```java
// Criando uma instância de Calendar
Calendar calendar = Calendar.getInstance();

// Definindo a data para 21 de abril de 2024
calendar.set(Calendar.YEAR, 2024);
calendar.set(Calendar.MONTH, Calendar.APRIL); // Lembrando que os meses começam de 0, então abril é 3
calendar.set(Calendar.DAY_OF_MONTH, 21);

// Imprimindo a data definida
System.out.println("Data definida: " + calendar.getTime());
```

calendar para string

```java

// Criando uma instância de Calendar
Calendar calendar = Calendar.getInstance();

// Formatando a data
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
String dataFormatada = sdf.format(calendar.getTime());

// Imprimindo a data formatada
System.out.println("Data formatada: " + dataFormatada);
```

calendar

sqldata para calendar

```
// Criando uma instância de java.sql.Date
Date sqlDate = Date.valueOf("2024-04-21");

// Convertendo java.sql.Date para Calendar
Calendar calendar = Calendar.getInstance();
calendar.setTime(sqlDate);

// Imprimindo o Calendar
System.out.println("Calendar: " + calendar.getTime());
```

calendar para sql data

```


// Criando uma instância de Calendar
Calendar calendar = Calendar.getInstance();

// Obtendo o timestamp do Calendar
long timestamp = calendar.getTimeInMillis();

// Criando um objeto java.sql.Date com o timestamp
Date sqlDate = new Date(timestamp);

// Imprimindo o java.sql.Date
System.out.println("Data em java.sql.Date: " + sqlDate);
```

java.sql.timestamp

```java
// Criando um objeto Timestamp para a data e hora atual
Timestamp timestamp = new Timestamp(System.currentTimeMillis());

// Imprimindo o Timestamp
System.out.println("Timestamp atual: " + timestamp);


String dateTimeString = "2024-04-21 12:30:45"; // String representando a data e hora
Timestamp timestamp = Timestamp.valueOf(dateTimeString);
```

timestamp para string

```java
// Criando um objeto Timestamp para a data e hora atual
Timestamp timestamp = new Timestamp(System.currentTimeMillis());

// Formatando o Timestamp
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
//SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
String timestampString = sdf.format(timestamp);

// Imprimindo a string formatada
System.out.println("Timestamp como string: " + timestampString);
```

java.sql.timestamp

Pegando apenas a data

```
// Criando um objeto Timestamp para a data e hora atual
Timestamp timestamp = new Timestamp(System.currentTimeMillis());

// Obtendo a parte da data
Date data = new Date(timestamp.getTime());

// Imprimindo a parte da data
System.out.println("Data: " + data);
```

Pegando apenas o tempo

```
// Criando um objeto Timestamp para a data e hora atual
Timestamp timestamp = new Timestamp(System.currentTimeMillis());

// Convertendo o Timestamp em um LocalTime
LocalTime localTime = timestamp.toLocalDateTime().toLocalTime();
```