Apresentação.java

ListaSequencial.java

```
1
2
3
4
5  public class ListaSequencial {
6      public static void main(String[] args) {
7          System.out.println("Bem-vindo à implementação de Lista
           Sequencial em Java!");
8      }
9  }
10
11
12
13
14
```

Java

ListaSequencial.java

Main.java

```java
// 1. Declaração da classe
public class ListaSequencial {

    // Lista para armazenar elementos
    ArrayList<Integer> lista = new ArrayList<Integer>();

    /*
     * Criamos uma classe chamada ListaSequencial
     * que encapsula uma lista sequencial.
     * A variável 'lista' é uma lista genérica
     * usada para armazenar os elementos.
     */
}
```

Java

ListaSequencial.java

Main.java

```java
// 2. Métodos de inserção
// 2.1 Inserção no início
public void inserirNoInicio(int elemento) {
    lista.add(0, elemento);
}

/*
 * Insere um elemento na posição 0,
 * deslocando os outros elementos para a direita.
 * Usamos o método add de ArrayList com o índice 0.
 */
```

Java

ListaSequencial.java

Main.java

```java
// 2.2 Inserção no final
public void inserirNoFinal(int elemento) {
  lista.add(elemento);
}
/*
 * Insere um elemento no final da lista.
 * Esse é o comportamento padrão do método add de ArrayList.
 */
```

Java

ListaSequencial.java | Main.java

```java
// 2.3 Inserção em uma posição específica
public void inserirNaPosicao(int posicao, int elemento) {
  if (posicao < 0 || posicao > lista.size()) {
    throw new IndexOutOfBoundsException("Posição inválida");
  }
  lista.add(posicao, elemento);
}
/*
 * Insere o elemento na posição especificada pelo índice posicao.
 * Antes da inserção, verificamos se a posição é válida:
 * - Posição inválida: Fora dos limites [0, tamanho da lista].
 * - Se inválida, lançamos uma exceção
(IndexOutOfBoundsException).
 */


```

Java

ListaSequencial.java

Main.java

```java
// 3. Métodos de remoção
// 3.1 Remoção do início
public void removerDoInicio() {
  if (lista.isEmpty()) {
    throw new IllegalStateException("A lista está vazia");
  }
  lista.remove(0);
}
/*
 * Remove o elemento na posição 0.
 * Antes de remover, verificamos se a lista está vazia:
 * - Vazia: Lançamos uma exceção (IllegalStateException).
 */
```

Java

ListaSequencial.java | Main.java

```java
// 3.2 Remoção do final
public void removerDoFinal() {
  if (lista.isEmpty()) {
    throw new IllegalStateException("A lista está vazia");
  }
  lista.remove(lista.size() - 1);
}
/*
 * Remove o último elemento (índice = tamanho da lista - 1).
 * Também verificamos se a lista está vazia antes de remover.
 */
```

Java

ListaSequencial.java | Main.java

```java
// 3.3 Remoção de posição específica
public void removerNaPosicao(int posicao) {
  if (posicao < 0 || posicao >= lista.size()) {
    throw new IndexOutOfBoundsException("Posição inválida");
  }
  lista.remove(posicao);
}
/*
 * Remove o elemento na posição especificada.
 * Verificamos se a posição é válida:
 * - Fora dos limites [0, tamanho da lista - 1] gera uma exceção.
 */
```

Java

ListaSequencial.java

Main.java

```java
  // 4. Método de pesquisa
public int buscarElemento(int elemento) {
    for (int i = 0; i < lista.size(); i++) {
        if (lista.get(i) == elemento) {
            return i;
        }
    }
    return -1;
}
/*
 * Realiza uma busca sequencial:
 * - Percorre a lista do início ao fim.
 * - Retorna o índice do elemento se for encontrado.
 * - Retorna -1 se o elemento não estiver na lista.
 */

```

Java

ListaSequencial.java | Main.java

```java
// 5. Método de atualização
public void atualizarElemento(int posicao, int novoValor) {
  if (posicao < 0 || posicao >= lista.size()) {
    throw new IndexOutOfBoundsException("Posição inválida");
  }
  lista.set(posicao, novoValor);
}
/*
 * Atualiza o valor de um elemento na posição especificada.
 * Verificamos se a posição é válida antes de alterar.
 * O método set do ArrayList substitui o valor no índice
fornecido.
 */
```

Java

ListaSequencial.java

Main.java

```java
// 6. Método de exibição
public void exibirLista() {
  System.out.println(lista);
}
/*
 * Exibe os elementos da lista no console usando o toString do
ArrayList.
 */
```

Java

ListaSequencial.java

Main.java

```java
// 7. Método principal (main)
ListaSequencial lista = new ListaSequencial();
Scanner scanner = new Scanner(System.in);
int opcao;
/*
 * Criação de uma instância de ListaSequencial para manipular os
métodos da lista.
 * O Scanner é usado para ler entradas do usuário.
 * A variável opcao armazena a escolha do menu.
 */
```

Java

ListaSequencial.java | Main.java

```java
// 8. Menu interativo
do {
  System.out.println("\nMenu de Opções:");
  System.out.println("1. Inserir no início");...
  ...System.out.println("0. Sair");
  System.out.print("Escolha uma opção: ");
  opcao = scanner.nextInt();
} while (opcao != 0);
scanner.close();
System.out.println("Encerrando o programa...");
/*
 * O menu é exibido ao usuário, com opções numeradas para realizar
diferentes operações.
 * O número digitado pelo usuário é armazenado em opcao.
 */

```

Java

ListaSequencial.java

Main.java

```java
// 9. Uso do try-catch
try {
  switch (opcao) {
    case 1 -> {
      System.out.print("Digite o elemento a ser inserido no início: ");
      int elemento = scanner.nextInt();
      lista.inserirNoInicio(elemento);
    }...
    ...}
    // Outras opções seguem a mesma lógica
  }
} catch (Exception e) {
  System.out.println("Erro: " + e.getMessage());
}

```

Java

ListaSequencial.java

Main.java

```
1  // 9. Uso do try-catch
2  /*
3   * O bloco try envolve a execução das operações com base na
   escolha do usuário.
4   * Garante que o programa não seja interrompido caso ocorra um
5  erro, como:
6   * - O usuário digitar um índice inválido.
7   * - Tentar remover ou acessar elementos em uma lista vazia.
8   * - Inserir dados incorretos (exemplo: digitar uma letra em vez
   de um número).
9   * O catch captura a exceção e exibe uma mensagem amigável usando
10 e.getMessage().
11  */
12
13
14
```

Java

ListaSequencial.java

Main.java

```
// 9. Uso do try-catch
//Exemplo Prático
/*
 * Se o usuário escolhe remover de uma posição específica (opção 6) e digita uma posição inválida:
 * - O método removerNaPosicao lança uma IndexOutOfBoundsException.
 * - O catch captura essa exceção e exibe:
 * - makefile
 * - Copiar código
 * - Erro: Posição inválida
 */
```

Java

ListaSequencial.java | Main.java

```java
// 10. Fechamento do programa
while (opcao != 0);
scanner.close();
System.out.println("Encerrando o programa...");
/*
 * O loop do-while garante que o menu seja exibido até que o
usuário escolha a opção 0.
 * O programa encerra corretamente ao fechar o scanner e exibir
uma mensagem de
saída.
 */
```

Java