Linguagem orientada a objetos (LOO)
HAMI - Nível 02

Encontro 10 - Criando Menus com Scanner

uniderp

Materiais e dúvidas

Nenhum material será enviado via e-mail.

Os materiais serão disponibilizados no **AVA** e no drivel: **https://bit.ly/uniderploo**

Dúvidas, questionamentos, entre outros deverão ser realizados **APENAS** pelo **e-mail** e **AVA.**

Whatsapp da disciplina acesse o link https://linklist.bio/profmurilo e selecione sua disciplina.

Tópicos da aula de hoje

Neste encontro, debateremos os seguintes tópicos:

- Método remover utilizando ArrayList
- Menus
- Scanner
- Aplicação prática
- Criação de novos métodos para o menu

Material para aula de hoje

Acesse o link
https://linklist.bio/profmurilo

Clique em:
Código para aulas práticas - LOO

Faça o download da pasta **aula10Exemplo**

Faça a **descompactação** dos arquivos

Abra o VsCode e selecione a pasta **aula10Exemplo**

Abra o arquivo **App.java** e execute o teste

O método para deletar um aluno é funcional?

Adaptar o código para ArrayList

8. Crie um novo método para excluirAluno utilizando como parâmetro um objeto do tipo **Aluno**

```
public void removerAluno(Aluno aluno) {
    if (alunos.remove(aluno)) {
        System.out.println("Aluno removido com sucesso.");
    } else {
        System.out.println("Aluno não encontrado.");
    }
}
```

Método remove do ArrayList

```
public boolean remove(Object o) {
    final Object[] es = elementData;
    final int size = this.size;
    int i = 0;
    found: {
        if (o == null) {
            for (; i < size; i++)
                if (es[i] == null)
                    break found;
        } else {
            for (; i < size; i++)
                if (o.equals(es[i]))
                    break found;
        }
        return false;
    }
    fastRemove(es, i);
    return true;
}
```

```
private void fastRemove(Object[] es, int i) {
    modCount++;
    final int newSize;
    if ((newSize = size - 1) > i)
        System.arraycopy(es, i + 1, es, i,
newSize - i);
    es[size = newSize] = null;
}
```

Adaptar o código para ArrayList

9. Faça os ajustes na classe Aluno, para que a comparação do ArrayList funcione corretamente

```
@Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Aluno aluno = (Aluno) obj;
        return rga.equals(aluno.rga);
    }

@Override
    public int hashCode() {
        return Objects.hash(rga); /
    }
```

Adaptar o código para ArrayList

10. Faça novos testes no método Main

- Crie novos alunos
- Adicione os alunos na turma
- Remova um aluno utilizando o novo método
- Liste os alunos da turma e seu tamanho

Trabalhando com Menus

Criar menus em ambiente de console é uma das tarefas mais repetitivas possíveis, pois o laço de repetição é perpétuo, até que o usuário finalize o programa, selecionando uma opção do menu.

Assim, é um excelente exemplo para o uso de funções em Linguagem Java.

Para isso, precisamos aprender a utilizar o Scanner.

Scanner

O que é o Scanner?

Scanner é uma classe da biblioteca java.util usada para ler a entrada de dados a partir de diferentes fontes, como o teclado (padrão System.in), arquivos, e até strings.

Principais funções:

Facilita a leitura de diferentes tipos de dados (inteiros, floats, strings, etc.).

É amplamente utilizada em programas que interagem com o usuário.

Scanner

Importando o Scanner

Para usar o Scanner no seu programa, você precisa importar a classe:

import java.util.Scanner;

Instanciando um objeto Scanner

Para capturar entradas do teclado, instanciamos um objeto Scanner associado ao fluxo de entrada padrão (System.in):

Scanner scanner = new Scanner(System.in);

Lendo diferentes tipos de dados

O Scanner tem métodos específicos para ler diferentes tipos de dados.

Ler uma string:

Método: nextLine()

```
System.out.print("Digite seu nome: ");
String nome = scanner.nextLine();
System.out.println("Seu nome é: " + nome);
```

O método nextLine() lê uma linha inteira, incluindo espaços.

Ler um número inteiro:

Método: nextInt()

```
System.out.print("Digite sua idade: ");
int idade = scanner.nextInt();
System.out.println("Sua idade é: " + idade);
```

O método nextInt() lê apenas números inteiros e espera uma entrada válida.

Dicas para Usar Scanner

Limpar o buffer: Quando alternar entre métodos como nextInt() ou nextDouble() e nextLine(), use scanner.nextLine() após ler um número para evitar que a quebra de linha seja capturada.

Validação de entrada: O Scanner pode lançar exceções se a entrada não for do tipo esperado. É uma boa prática tratar erros com **try-catch** ou **verificar a entrada** antes de ler.

Benefícios:

- Simples e eficiente para capturar entradas no terminal.
- Facilita a leitura de múltiplos tipos de dados com métodos especializados.
- Ideal para programas que interagem com o usuário, como menus interativos e sistemas de cadastro.

Criação de um Menu Interativo no Terminal

Objetivo: Permitir que o usuário interaja com o sistema, escolhendo opções.

Estrutura básica:

```
int opcao;
do {
    System.out.println("1. Cadastrar Aluno");
    System.out.println("2. Inserir Aluno na Turma");
    System.out.println("3. Sair");
    opcao = scanner.nextInt();
} while (opcao != 3);
```

Conceito: Uso de um laço do-while para manter o menu em execução até que o usuário escolha a opção de sair.

Implementando a Lógica do Menu

Opções comuns no sistema:

- Cadastrar aluno.
- Inserir aluno na turma.
- Remover aluno da turma.
- Listar alunos.

Chamada de métodos:

```
switch (opcao) {
    case 1:
        cadastrarAluno();
        break;
    case 2:
        inserirAlunoNaTurma();
        break;
    default:
        System.out.println("Opção inválida");
}
```

Executando o Sistema de Escola

```
public class App {
    public static void main(String[] args) {
        SistemaEscola sistema = new SistemaEscola();
        sistema.executar();
    }
}
```

Vamos testar o sistema:

- Cadastrar um aluno.
- Inserir o aluno na turma.

Agora é sua vez!

Realize a implementação

Crie novas opções no menu, assim como as funcionalidades para sua execução.

- **3. Remover Aluno da Turma**
  - Essa função deve ter retorno do tipo void
  - Deve realizar a leitura do cpf do aluno a ser removido exibindo a mensagem: "**Digite o CPF do aluno a ser removido:**"
  - Deve utilizar a busca do Aluno através do método buscarAlunoPorCpf
  - Caso encontre o aluno, realize a remoção, caso não encontre o aluno exiba uma mensagem de erro: "**Aluno não encontrado.**"

- **4. Listar alunos da turma**
  - A classe turma possui um método para listagem, então basta chamar o método para essa função

Realize a implementação

- **5. Definir idade de um aluno**
  - Essa função deve ter retorno do tipo void
  - Deve realizar a leitura do cpf do aluno a ser editado exibindo a mensagem: "**Digite o CPF do aluno a ser editado:**"
  - Caso encontre o aluno, deve ser solicitado a idade a ser definida, exibindo a mensagem: "**Digite a idade do aluno: NOME-DO-ALUNO**"
  - Realize a edição da idade
  - Faça a impressão de todos dados do aluno através do método **toString** da classe **Aluno**

- **6. Remover todos alunos de uma turma**
  - Implementar novo método na classe turma para realizar a remoção de todos alunos.
  - Fazer a confirmação da ação, exibindo a mensagem: "**Você tem certeza que deseja remover todos alunos da turma? 1. Sim 2. Não**"
  - Caso digite 1, realize a remoção, caso digite 2 volte para o menu principal

Realize a implementação

- **7. Permitir que mais de uma turma seja criada**
  - Esta funcionalidade é mais extensa, então irá precisar de várias refatorações e implementações.
- **7.1. Criar nova turma**
  - Na classe **Turma**, crie um novo atributo chamado **codigo** do tipo **String**, defina seus métodos de Get e Set e crie um novo **construtor** para a classe que tenha a assinatura **(String nome, String codigo)**
  - Na classe **SistemaEscola,** faça a definição de um novo **arrayList** com o nome **turmasCadastradas**, crie a opção no menu para inserção de uma nova turma, solicitando o nome da turma e o código. (Similar ao cadastro de alunos)
- **7.2. Refatoração dos métodos existentes**
  - Crie o método **buscarTurmaPorCodigo** (similar ao **buscarAlunoPorCpf**)
  - Para as funcionalidades "**Inserir Aluno na Turma**", "**Remover Aluno da Turma**", "**Listar alunos da turma**" e "**Remover todos alunos de uma turma**" insira um passo novo onde é solicitado o código da turma, e realize a ação de acordo com a turma selecionada através do código.

Lista de exercícios

A lista será liberada para ser feita até o dia 18 de outubro - Sexta-Feira às 23:59 via Google Forms

https://forms.gle/iDDuPMULtJASDvoi8

Realize as implementações solicitadas nos slides anteriores, salve em um novo repositório do git e envie o link via google forms.

Caso tenha dúvidas estarei disponível até a quinta-feira para responder.

Encerramento

DÚVIDAS, CRÍTICAS E SUGESTÕES, ENVIAR PARA:

murilo.g.costa@cogna.com.br