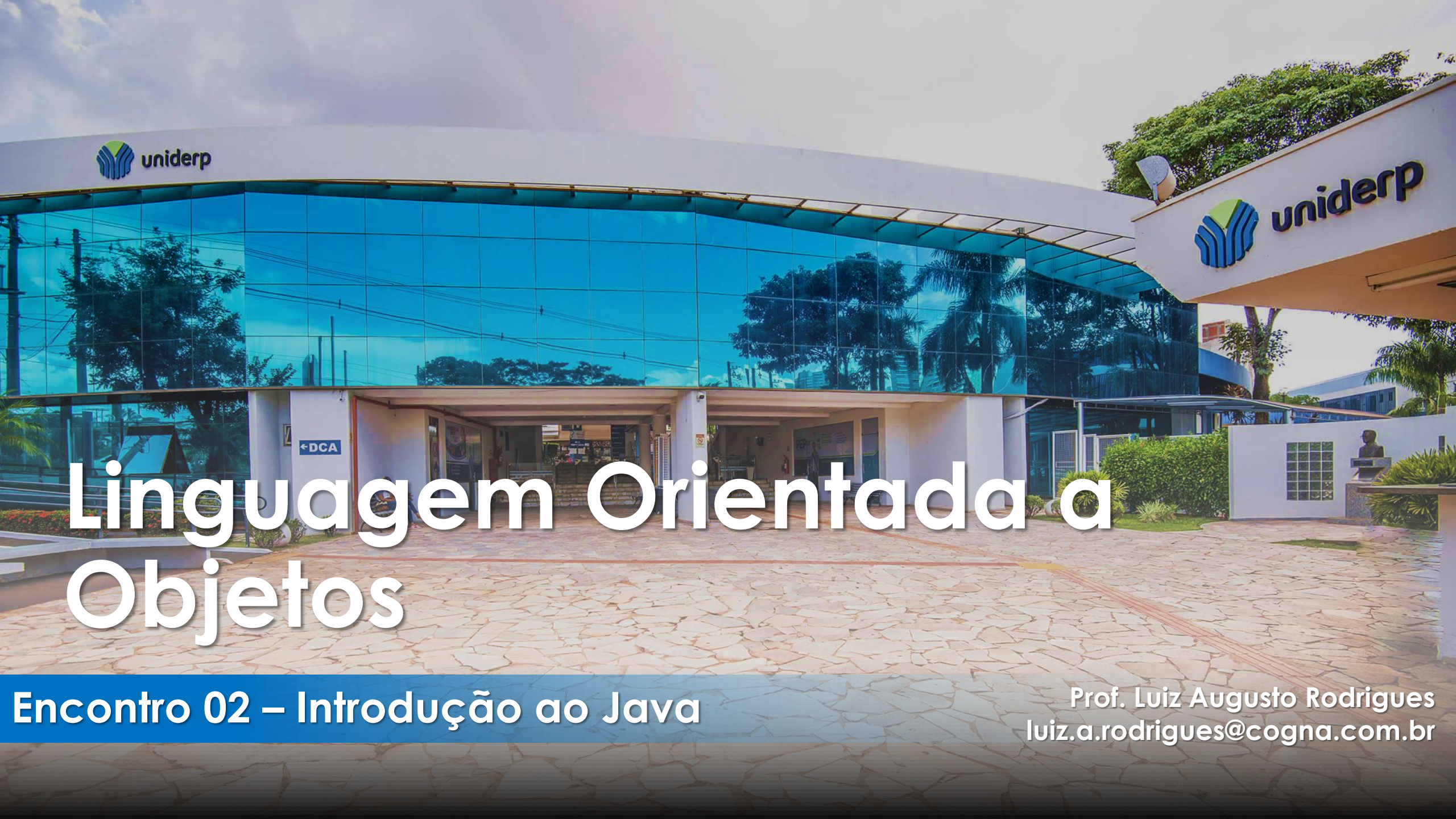

Linguagem Orientada a Objetos

Encontro 02 – Introdução ao Java

Prof. Luiz Augusto Rodrigues
luiz.a.rodrigues@cogna.com.br

Avisos Importantes

Nenhum material será enviado via e-mail. Os materiais serão disponibilizados no **AVA** e no google drive:
https://bit.ly/47qzfvl

Dúvidas, questionamentos, entre outros deverão ser realizados pelo **e-mail** e pelo **Whatsapp** da disciplina**.**

Para ingressar no grupo do **Whatsapp** da disciplina acesse o link a seguir e selecione sua disciplina.
linklist.bio/profluizao_2023-2

A Disciplina de Programação Orientada a Objetos, a partir deste slide, será referenciada pela sigla **LOO**.

Assuntos Abordados

Neste encontro, debateremos os seguintes tópicos:

- Visão Geral
  - Conceitos
  - Tecnologia
  - Linguagem
  - API
  - Máquina Virtual
  - Ambiente
- Desenvolvimento
  - Package
- Classe, Objeto e Instância
- Palavras-chave
- Prática
  - Exercícios

LOO

Encontro 02

Instruções Importantíssimas!!!

Instruções Importantíssimas!!!

Para este semestre, na disciplina de POO, utilizaremos OBRIGATORIAMENTE as seguintes ferramentas:

- Sistema Operacional Windows 10, Linux ou MacOS.
- Linguagem de Programação Java.
- JDK 17 ou superior
  - https://download.oracle.com/java/17/archive/jdk-17.0.6_windows-x64_bin.exe
- VS Code.

Instruções Importantíssimas!!!

- Git 2.39.2 ou superior
  - https://github.com/git-for-windows/git/releases/download/v2.39.2.windows.1/Git-2.39.2-64-bit.exe
- Github
  - https://github.com/
  - https://desktop.github.com/
- Tutorial em Git e Github
  - Tutorial Git e Github 2022 – Introdução prática para iniciantes
  - Canal DevSuperior
  - https://youtu.be/_hZf1teRFNg

Quem for utilizar Linux ou MacOSx, fale comigo após a aula, ou pelo grupo do WhatsApp, para instruções de instalação.

Instruções Importantíssimas!!!

- Os relatórios não serão utilizados nessa disciplina, por se tratar de uma disciplina 90% prática.
- No lugar, teremos atividades de pós-aula, que deverão ser respondidas e justificadas, para fixação do conteúdo discutido na aula.
- As atividades de aula estarão disponíveis apenas no Github da turma.
- O tutorial do Git e Github é **obrigatório**.

LOO

Encontro 02

Pilares de POO

Princípios de POO

Para uma linguagem ser considerada no paradigma do POO, existem quatro características principais que precisam fazer parte de sua aplicação:

- Abstração,
- Encapsulamento,
- Herança
- Polimorfismo.

Princípios de POO

LOO

Encontro 02

Visão Geral

Conceitos

Java é a base de praticamente todos os tipos de aplicativo em rede e é o padrão global para desenvolvimento e fornecimento de aplicativos para celular, jogos, conteúdo on-line e software corporativo.

Foi projetada com alguns objetivos como orientação a objetos, portabilidade (pode ser executado em qualquer ambiente), recursos de redes (grande quantidade de biblioteca cooperam com os protocolocos de rede), segurança e a sintaxe similar a C/C++.

Conceitos

O código criado pelo programador não é interpretado pelo ambiente onde está sendo executado, é gerado um código intermediário, denominado bytecode, que será interpretado e executado pela máquina virtual (JVM), dessa forma o programa pode ser executado em uma grande quantidade de plataforma.

Tecnologia

A tecnologia Java é formada pela a linguagem de programação e a plataforma. Programação Java é uma linguagem de alto nível que tem as características abaixo:

- Orientada a Objetos
- Distribuída
- Multitread
- Portável
- Alta Perfomance
- Robusta
- Segura

Tecnologia

Em Java os códigos-fonte são escritos em arquivos com a extensão .java, que depois são compilados e transformados em arquivos .class pelo compilador javac. Esses arquivos contém os ***bytecodes*** que serão interpretados e executados pela Máquina virtual (JVM).

A plataforma Java é formada por dois componentes:

- Java Virtual Machine (JVM): é a máquina virtual java, responsável em interpretar e executar os programas escritos na linguagem java.
- Java Application Programming Interface (API): é uma coleção de componentes de software, agrupada em bibliotecas de classes e interfaces que são conhecidas como packages. Elas disponibilizam funções prontas para serem usadas pelos programadores.

Linguagem

A linguagem Java é orientada a objetos e é parecida com a sintaxe do C/C++, simples e robusta, minimizando assim bugs e aumentando a produtividade, dá suporte a threads nativo, acoplamento em tempo de execução, tem um ***garbage colletor*** (coletor de lixo) para remover valores da memória, não depende de plataforma.

O código intermediário é interpretado por uma máquina virtual, disponibilizando assim uma compilação rápida, a sintaxe é uniforme e rigorosa a quanto a tipos.

API

Java disponibiliza uma grande coleção de APIS organizadas em pacotes(java.*, javax.* e extensões) usadas pelos ambientes de execução(JRE) e de desenvolvimento (SDK). Principais Apis são distribuídas juntamente com os produtos para desenvolvimento (J2SE, J2EE, J2ME).

Máquina Virtual

Quando um programa Java é compilado um código intermediário é gerado, chamado de ***bytecode***.

Este bytecode é interpretado pelas máquinas virtuais java (JVMs) para a maioria dos sistemas operacionais.

A máquina virtual é a responsável por criar um ambiente multiplataforma.

A execução de um programa depende da origem do código a ser executado e da forma como foi implementada a JVM.

Ambiente

- Java 2 System Development Kit: é uma coleção de ferramentas de linha de comando para compilar, executar e depurar aplicações Java.

- Java Runtime Environment: conjunto de arquivo que é necessário para executar aplicações Java.

- Java Micro Edition: utilizados para rodar aplicações java em dispositivos móveis.

Ambiente

Algumas ambiente de desenvolvimento integrado(IDE) para desenvolver Java:

- JCreator: editor de sintaxe para a linguagem de programação Java leve e muito elegante.
- NetBeans: é um ambiente de desenvolvimento integrado (IDE) gratuito e de código aberto para desenvolvedores de software nas linguagens Java, C, C++, PHP, Groovy, Ruby, entre outras
- Eclipse: é uma IDE para o desenvolvimento aplicações nas linguagens Java, C, C++, PHP, Groovy, Ruby, entre outras.
- VS Code: editor de código avançado da MS, que permite através das extensões, a codificação e compilação de código Java.

LOO

Encontro 02

Desenvolvimento

Package

Um Package (Pacote) é uma forma de organizar as classes dentro dos namespaces similar nos outros módulos.

Um pacote fornece um namespace exclusivo para os tipos que ele contém.

Exemplos:

br.com.uniderp.poo.2022.atacado.dominio

br.com.uniderp.apd.2022.sistema

br.com.empresa.financeiro.controle

br.com.empresa.admin.rh.cadastro

Classe, Objeto e Instância

- Classe e um conjunto de objetos que possuem estados semelhantes, comportamento comum e relacionamentos comuns com outros objetos.
- Objeto é um elemento do munto real, contém comportamentos (forma com que reage a estímulos) e atributos (características).
- Instância é criar fisicamente uma representação concreta da classe.
- O método main() é o primeiro método que vai ser chamado pelo interpretador Java, o argumento é um vetor formado por textos passados na linha de comando.

Palavras-chave

Palavras-chave são requisitos básicos para a prova de certificação, são conhecidas também como palavras reservadas.

Essa palavras não podem ser usadas para representar variáveis, classes ou nomes de métodos.

A seguir listamos as palavras reservadas mais comum da linguagem.

Palavras-chave

- **abstract**: um método marcado como abstract informa que o mesmo não possui uma implementação ainda, mas que uma classe que o estiver herdando precisará implementá-lo.
- **assert**: utilizado para fazer debug de lógica em tempo de execução.
- **boolean**: para variáveis de valor lógico: true e false.
- **break**: normalmente utilizado para interromper execuções de estruturas de repetição.
- **byte**: para variáveis numéricas de precisão -128 até 127.
- **case**: indica uma opção entre várias em blocos catch.

Palavras-chave

- **catch**: é utilizado juntamente com try, seu bloco é executado somente em caso de o programa lançar uma exceção do tipo indicado no seu parâmetro.
- **char**: para variáveis de caracteres, onde a sua representação interna equivale a um tipo numérico.
- **class**: para definir o início de um arquivo Java, todas as classes possuem pelo menos essa palavra-chave.
- **const**: essa palavra não tem uso específico em Java mas mesmo assim é uma palavra-chave.
- **continue**: para pular a iteração atual de uma estrutura de repetição.
- **default**: normalmente utilizado para o final de uma ou mais opções case´s de um bloco catch.
- **do**: estrutura de repetição que garante que o bloco será executado pelo menos uma vez durante a execução do programa.

Palavras-chave

- **double**: para variáveis numéricas e de pontos flutuantes com precisão de 64 bits.
- **else**: complemento de estrutura de condição.
- **enum**: palavra-chave adicionada na versão 5 do Java, é um tipo especifico de dados, que assemelha-se com uma classe, que tem operações e dados internos.
- **extends**: utilizado para aplicar o conceito de herança para uma classe, onde uma classe receberá os métodos e variáveis de instância da classe chamada de pai.
- **final**: marca uma variável, classe ou método para que não seja possível modificar o seu valor ou comportamento no decorrer da execução do programa.

Palavras-chave

- **finally**: compõe o início de um bloco que sempre é executado para um bloco de tratamento de erros, mais utilizado para limpar recursos que foram abertos no bloco de tratamento.
- **float**: variáveis numéricas e de pontos flutuantes com precisão de 32 bits.
- **for**: estrutura de repetição que declara, testa e incrementa variável para uso local.
- **goto**: não tem uso específico na linguagem.
- **if**: estrutura de condição mais comum da linguagem.
- **implements**: informa que uma determinada classe irá implementar uma determinada interface.
- **import**: para relacionar classes externas à atual, permitindo o uso de nomes mais curtos para recursos da classe externa.

Palavras-chave

- **instanceof**: serve para fazer o teste "É-UM" com duas referências.
- **int**: para variáveis numéricas de precisão -2.147.483.648 até 2.147.483.647.
- **interface**: informa que o modelo não é uma classe, mas sim um protótipo de classe sem implementação para os métodos, obrigando as classes que a implementarão siga as regras de retorno, assinatura de métodos, etc...
- **long**: para variáveis numéricas de precisão de 64 bits.
- **native**: métodos marcados como native dizem que sua implementação é feita em uma outra linguagem (por exemplo C), para que se possa acessar recursos específicos do sistema operacional.
- **new**: utilizada para se criar novas instâncias de objetos.

Palavras-chave

- **package**: informa em que estrutura de diretórios a classe está localizada.
- **private**: marca a visibilidade de um método ou variável de instância para que apenas a própria classe acesse.
- **protected**: marca a visibilidade de um método ou variável de instância para que a própria classe ou suas filhas acessem.
- **public**: marca a visibilidade de uma classe, método ou variável de instância para que todas as classes em todos os pacotes tenham acesso.
- **return**: devolve para o método chamador um valor que é do mesmo tipo declarado na assinatura do método.
- **short**: para variáveis numéricas de precisão de -32.768 até 32.767.

Palavras-chave

- **static**: marca um método ou variável para que se tenha apenas uma cópia da memória desse membro.
- **strictfp**: serve para aumentar a precisão em operações com pontos flutuantes.
- **super**: chama membros da classe pai.
- **switch**: representa blocos de decisões de fluxos semelhantes ao if, mas com mais organização em determinadas situações.
- **synchronized**: um método com essa marcação será controlado para que não se possa ter duas threads acessando o mesmo objeto.
- **this**: representa a instância que está atualmente sendo executada.
- **throw**: é utilizado para lançar uma exceção.

Palavras-chave

- **throws**: é utilizado para se declarar que um método pode lançar uma exceção.
- **transient**: indica que uma determinada variável de instância não será serializada junto com o objeto da classe.
- **try**: para executar métodos que têm chances de lançar exceções, mas que serão tratados em blocos catch´s que o seguirão.
- **void**: representa um retorno vazio, ou seja, nenhum retorno para esse método.
- **volatile**: indica que uma determinada variável de instância pode ser modificada em duas threads distintas ao mesmo tempo.
- **while**: bloco de repetição que será executado enquanto seu parâmetro estiver retornando verdadeiro (true).

LOO

Encontro 02

Prática

Exemplo 01

```
public class Program {
    public static void main(String[] args) {
        System.out.println("Hello World.");
        System.out.println("Primeiro Exemplo Java.");
        System.out.println("Introdução a Java.");
  }
}
```

Exemplo 02

```
public class Program {
  public static void main(String[] args) {
    int value1 = 11;
    int value2 = 6;
    System.out.println("Value 1 = 11 / Value 2 = 6");
    System.out.println("Soma = " + (value1 + value2));
    System.out.println("Subtração = " + (value1 - value2));
    System.out.println("Divisão = " + ((float)value1 / (float)value2));
    System.out.println("Multiplicação = " + (value1 * value2));
    System.out.println("Resto = " + (value1 % value2));
  }
}
```

Exemplo 03

```
import java.util.Scanner;
public class Program {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.println("Digite sua idade: ");
    int idade = s.nextInt();
    System.out.println("Vc tem " + idade + " anos.");
  }
}
```

Exemplo 04

```
import java.util.Scanner;
public class Program {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.println("Digite seu nome: ");
    String nome = s.nextLine();
    System.out.println("\nDigite sua altura: ");
    double altura = s.nextDouble();
    System.out.println(nome + " tem " + altura + " de altura.");
  }
}
```

Exemplo 05

```
import java.util.Scanner;
public class Program {
  public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
       System.out.println("Digite um número inteiro.");
       int num = s.nextInt();
       for(int index = 1;index <= 10;index++)
       {
             System.out.println(num + " X " + index + " = " + (index * num));
       }
  }
}
```

LOO

Encontro 02

Atividade de Aula

Atividade de Aula

1. Responder a atividade, denominada “LOO – Atividade 02.pdf”, disponível apenas no Github e no Google Drive. VALE NOTA!!!

2. Enviar as respostas justificadas no e-mail do professor, em formato PDF, com o seguinte assunto:

[LOO] ATIVIDADE 02

3. Data de entrega: 21/08/2023, até às 23h00.

Encerramento

Dúvidas e sugestões, entre em contato pelo whatsapp da disciplina, ou mande um e-mail para luiz.a.rodrigues@cogna.com.br