🐴 Desafio Java:
Projeto **S.O.S. Jumento Nordestino** - Uma Solução Modular ## 🚨 Regras da Avaliação (Estrutura e Conduta) Este é um exame prático individual com regras estritas de entrega e conduta. | Conduta | Status | Detalhes | | :--- | :--- | :--- | | **Consulta a Materiais/IA** | ❌ **PROIBIDA** | Proibido usar qualquer material externo, internet ou ferramentas de IA. | | **Comunicação com Colegas** | ❌ **PROIBIDA** | Qualquer forma de comunicação com colegas. | | **Entrega do Código (`git push`)** | ❌ **ANULADA** | **Se feito fora do horário de aula.** | | **Local do Commit** | ❌ **ANULADA** | **Se o commit for detectado fora da rede/localização da Universidade.** | --- ## ⏰ Regras de *Commit* e Presença (ESTRITO) **Atenção: Estas regras são auditáveis e o não cumprimento anulará a sua avaliação.** 1. **Horário de Aula:** A prova deve ser realizada e finalizada **estritamente** no horario de aula. 2. **Registro de Presença:** Para validar sua participação, você deve assinar a lista de presença física ao iniciar a prova. ### Validação por *Commit* (Obrigatório) * **Validação de Tempo:** O **último *commit*** no seu repositório Git, contendo o código final, deve ter o *timestamp* **dentro do horário de aula**. * *Exemplo:* Se o horário final é 10h00, um *commit* às 10h01 anula a prova. --- **Esta é uma prova individual e sem consulta.** ## Leia a noticia a seguir: A invenção brasileira que promete salvar os jumentos da extinção no Nordeste Os principais pesquisadores de equinos de universidades federais do Brasil têm se unido para enfrentar o risco de extinção de jumentos no Nordeste. A crise é consequência de uma demanda bilionária da China que compra as peles destes animais para fazer ejiao, um elixir que promete vitalidade, entre outros benefícios para saúde, segundo os preceitos milenares da Medicina Tradicional Chinesa (MTC). Fonte: https://www.bbc.com/portuguese/articles/c5yj905ep4yo
Acesso em 2025/10/18 ## 💡 Contexto do Problema A população de jumentos (*Equus asinus*) no Nordeste brasileiro tem sofrido uma redução drástica devido à demanda pelo *ejiao*, um produto de colágeno extraído de sua pele. Para combater essa ameaça de extinção, cientistas buscam soluções que permitam o monitoramento e a valorização da espécie, promovendo uma gestão sustentável do rebanho restante e buscando alternativas para a produção de colágeno. Você foi contratado(a) como desenvolvedor(a) para criar o protótipo de um sistema de monitoramento de rebanhos, utilizando o conceito de **funções (métodos)** em Java para modularizar e organizar o código. ## 🎯 Objetivo da Prova Demonstrar o domínio na criação e utilização de **funções (métodos)** em Java, com e sem parâmetros, com e sem retorno (`void`), para resolver um problema prático e modular. ## 💻 Requisitos Técnicos Seu código deve estar em um único arquivo `.java` (`MonitoramentoJumentos.java`) e precisa implementar **pelo menos 4 métodos estáticos** (funções) além do `main`, cada um com responsabilidades únicas, conforme detalhado abaixo: ### Requisito 1: Cálculo e Gestão do Risco de Extinção (Função com Retorno) Crie uma função chamada `calcularTaxaReducao` que: 1. **Recebe como Parâmetros:** * `populacaoInicial` (tipo `int`): População de jumentos nas últimas décadas (Utilize o valor fictício de **1.400.000** para simulação, baseado no contexto de antes do declínio). * `populacaoAtual` (tipo `int`): População atual de jumentos (Utilize o valor fictício de **84.000** para simulação). 2. **Processamento:** Calcula a taxa percentual de redução da população. $$\text{Taxa de Redução} = \left(1 - \frac{\text{População Atual}}{\text{População Inicial}}\right) \times 100$$ 3. **Retorna:** A taxa de redução calculada (tipo `double`). ### Requisito 2: Determinação da Categoria de Risco (Função com Parâmetro e Retorno Booleano) Crie uma função chamada `estaEmRiscoCritico` que: 1. **Recebe como Parâmetro:** * `taxaReducao` (tipo `double`): A taxa calculada no Requisito 1. 2. **Processamento:** Verifica se a taxa de redução é **igual ou superior a 90%** (critério de risco extremo simulado). 3. **Retorna:** Um valor booleano (`boolean`): `true` se o risco for crítico, `false` caso contrário. ### Requisito 3: Simulação de Abates (Função com Parâmetros e Retorno) Crie uma função chamada `simularImpactoAbate` que: 1. **Recebe como Parâmetros:** * `populacaoAtual` (tipo `int`): População de jumentos. * `abatesAnuais` (tipo `int`): O número de abates por ano (Utilize o valor real médio de **50.000** para simulação, baseado na notícia). * `anosSimulados` (tipo `int`): O número de anos futuros para simular (Peça esta entrada ao usuário dentro do `main`). 2. **Processamento:** Calcula a população restante após o período simulado. *Considere o pior cenário, onde a população não se reproduz, apenas diminui.* $$\text{População Restante} = \text{População Atual} - (\text{Abates Anuais} \times \text{Anos Simulados})$$ 3. **Retorna:**

A população restante (tipo `int`). ### Requisito 4: Exibição da Declaração Final (Função sem Retorno - `void`) Crie uma função chamada `exibirDeclaracaoFinal` que: 1. **Recebe como Parâmetros:** * `populacaoRestante` (tipo `int`): Resultado do Requisito 3. * `anos` (tipo `int`): O número de anos simulados. 2. **Processamento:** * Imprime uma mensagem de **alerta** se a `populacaoRestante` for **menor ou igual a zero**. * Imprime um resumo da situação, informando a população restante após os `anos` simulados. ### Método Principal (`main`) O método `main` deve funcionar como o orquestrador do sistema: 1. Declarar as variáveis de população inicial e atual. 2. Solicitar ao usuário o número de anos para a simulação de abate (Requisito 3), utilizando a classe `Scanner`. 3. **Chamar sequencialmente** as funções criadas, utilizando seus retornos como parâmetros para as chamadas subsequentes, garantindo a modularidade do código. 4. O `main` **não deve conter a lógica de cálculo**, apenas a lógica de entrada/saída e as chamadas de método. ## Massa de Teste Para te apoiar no teste do algoritmo, um analista de sistemas levantou os requisitos para voce: ~~~ === S.O.S. JUMENTO NORDESTINO - SISTEMA DE MONITORAMENTO === População inicial de referência: 1400000 População atual estimada: 84000 Abates anuais (simulados): 50000 [Análise Inicial] Taxa de Redução Histórica: 94,00% [Status] Espécie em Risco CRÍTICO (Redução > 90%)! Quantos anos futuros você deseja simular o impacto dos abates (Ex: 1, 3, 5)? 1 --- RESULTADO DA SIMULAÇÃO --- População restante após 1 anos: 34000 jumentos. Ainda há esperança! Iniciativas de conservação são vitais. Programa de Monitoramento Finalizado. ~~~ ## 🌟 Critérios de Avaliação 1. **Uso Correto de Funções (Métodos):** Definição correta de tipos de retorno, parâmetros e modificadores (`public static`). 2. **Modularidade:** A lógica principal do cálculo deve estar isolada nos métodos, e o `main` deve apenas coordenar as chamadas. 3. **Clareza e Legibilidade:** Utilização de nomes de métodos e variáveis descritivos, seguindo as convenções de Java (camelCase). 4. **Resultados Corretos:** Os cálculos devem estar matematicamente corretos, e as verificações lógicas (`if/else`) devem ser aplicadas conforme as regras de negócio (Critério de Risco e Declaração Final). ## 📝 Entrega 1. Crie um novo repositorio chamado: una-psc-prova-a1-matricula-impar-202502. (2 pts) 2. Adicione um arquivo .gitignore, um arquivo licença e suba este arquivo readme. (3 pts) 3. Crie um arquivo chamado `MonitoramentoJumentos.java`. (5 pts) 4. Insira todo o seu código Java neste arquivo. (5 pts) 5. Inclua seu nome completo e número de matrícula como comentário no cabeçalho do arquivo. (1 pts) 6. Submeta o arquivo `MonitoramentoJumentos.java` no seu repositorio. (5 pts) 7. Teste seu algoritmo e envie uma imagem dele em funcionamento. (9 pts) Qualquer descumprimento de regras ou de alguma solicitacao da prova deve acarretar em perda total ou parcial dos pontos. Entregue o link do seu repositorio na plataforma. **Lembre-se: A clareza do seu código e a facilidade de compreensão da saída são cruciais. Confie no seu conhecimento!** **Bom trabalho e sucesso!** ***