## linha horizontal

Avaliação A3

02/12/2023

**─**

Pedro Henrique Meireles da Silva

UNIFACS

Análise e Desenvolvimento de Sistemas

Sistemas computacionais e segurança

github: pedromeirelessz

# 

# 

# 

# 

# Visão geral

A avaliação envolve a análise de um código disponibilizado pelo professor. Achar vulnerabilidades e desenvolver melhorias.

# Especificações

Linguagem utilizada: Java

IDE utilizada: Eclipse

Versão do Java: JDK 17

# Análise do código:

O código tem como finalidade a realização de operações entre dois números binários, utilizando a classe Scanner para entrada e Integer.parseInt para converter os valores binários de string para inteiro. Em termos de estrutura, é bastante simples, envolvendo apenas uma classe e um pacote.

# Identificando vulnerabilidades:

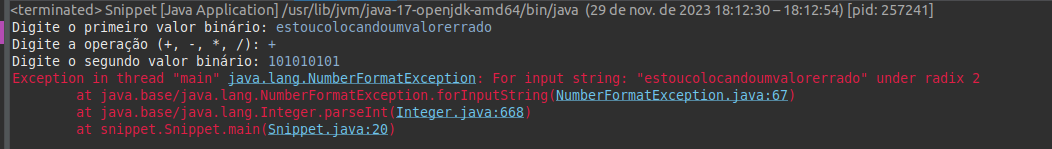
-Injeção de Código: A ausência de validação na entrada torna a aplicação mais suscetível a injeções de código.

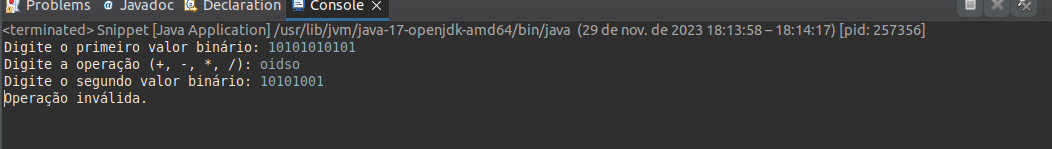
-Estouro de Buffer: Não existem verificações para ter a garantia que operações aritméticas não resultem no estouro de buffer.

-Falta de classes: A falta de classes, deixar todo o controle de operações na classe Main, deixando o código menos coeso e delegando menos as funções.

-Mensagem de erro: A mensagem de erro pode informar ao usuário informações delicadas sobre a aplicação, então é necessário assegurar que ela seja genérica e não forneça detalhes caso ocorra.

-Na prática:





Nota-se que não há verificação para garantir que os números binários sejam realmente binários, além de expor o programa podendo mostrar informações específicas sobre a aplicação. Existe uma verificação apenas nas operações, porém elas não são tratadas de forma adequada, apesar de ter uma mensagem de erro genérica, o usuário ainda tem que fechar a aplicação e abrir novamente.

# Melhorias gerais:

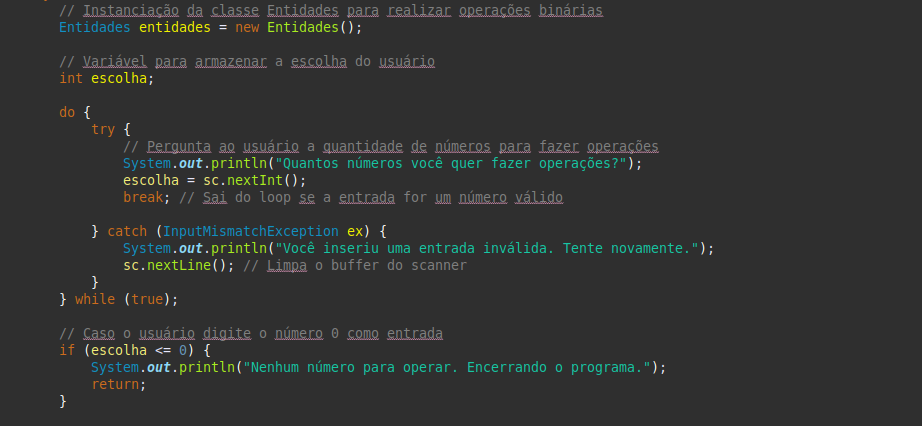
-Encapsulamento da aplicação: O código original estava em uma única classe, agora com a classe entidades no pacote model.entities é feita a reutilização e organização do código, deixando o código mais coeso e delegando melhor as funções.

-Tratamento de exceções e validação para entradas: No código original não se tinha um tratamento de exceções adequado e validações da mesma, fazendo o programa quebrar, foi adicionado blocos de tratamento de exceções específicos, funções e métodos para assegurar que a aplicação lide contra entradas inválidas. Incluindo a verificação do número de entrada e a validação de operações.

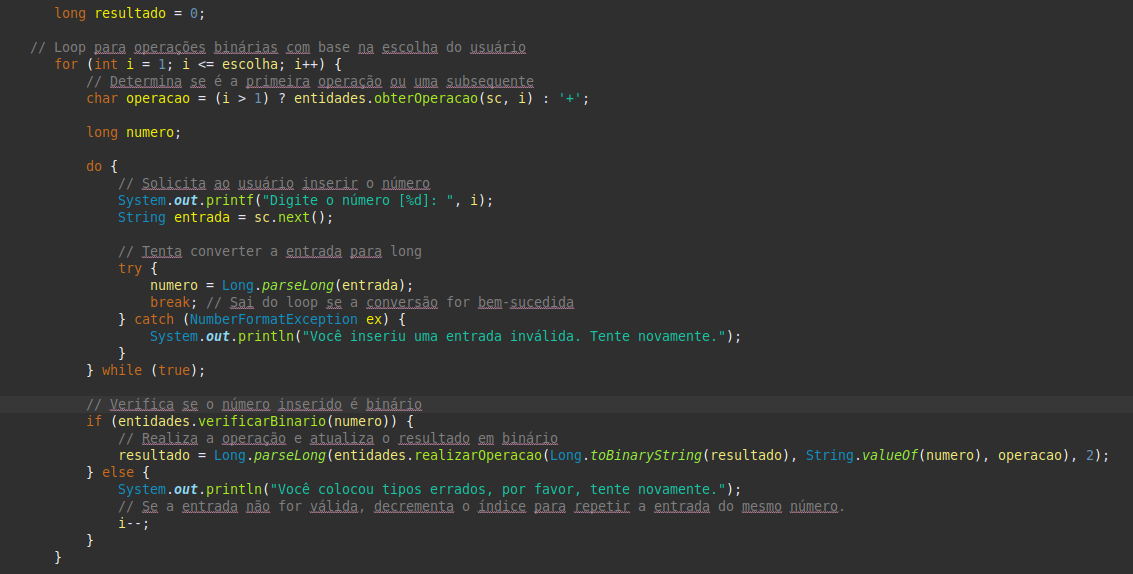
-Aumento de números: No código original era possível fazer operação entre apenas 2 números binários, agora é possível fazer operações entre quantos números o usuário desejar.

# Explicando melhorias detalhadamente:

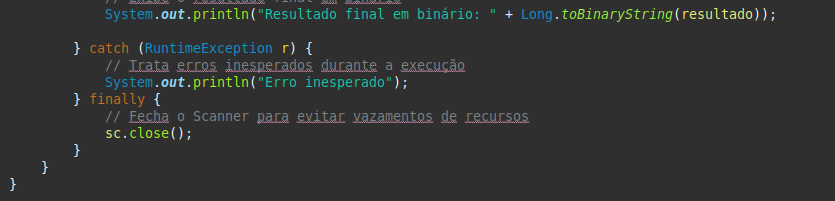
-Pacote application classe Main:



A instância da classe Entidades é criada e pode ser chamada no objeto entidades, podendo agora ser utilizado os métodos criados na classe entidades, deixando o código mais coeso e delegando melhor as funções. Neste escopo em específico, é utilizado um loop do-while para que o usuário digite a quantidade de números com os quais ele quer realizar operações. Utilizamos o (sc.nextInt()) para ler a entrada inteira digitada. Caso seja inserido algo que não seja um valor do tipo int, irá ser capturada a exceção InputMismatchException, então a aplicação irá exibir uma mensagem de erro e limpar o buffer do scanner com a reutilização (sc.nextLine()) ( erro de looping ). Na condição if, caso a quantidade escolhida seja menor ou igual a zero, irá exibir uma mensagem informando que não há números para operar e encerra o programa.

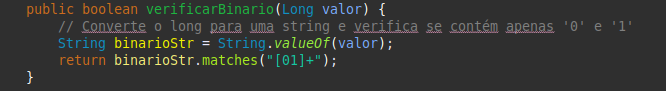


Funcionando a partir da quantidade de números da entrada. Com a variável operação sendo determinada com base no índice ‘i’. Caso o ‘i’ seja maior que 1, o programa irá chamar o método ‘obterOperacao’ da classe Entidades, para que seja feita a soma com o número anterior. Caso esteja ainda no 1 (a primeira operação), a operação padrão é definida como adição (+), e será somada com a variável ‘resultado’ e definimos ela como zero acima. Logo após irá entrar em um loop do-while, solicitando ao usuário que digite o número correspondente ao índice atual, dentro será utilizando um bloco try-catch para converter o número para um tipo long. Caso a conversão for bem-sucedida, o loop será interrompido com break. Então irá entrar a estrutura condicional if, invocando o método ‘verificarBinario’, colocando o número como parâmetro, logo após iremos atribuir a variável long ‘resultado’ que está fora do escopo do for, o resultado da operação do método ‘realizarOperacao’ que retorna um string, então utilizamos o método ‘Long.parseLong’ para converter para um tipo long, e utilizamos o ‘Long.toBinaryString’ para converter a variável ‘resultado’ para uma String binária, e a variável ‘numero’ apenas utilizamos o método ‘String.valueOf’ pois essa variável já passou pelo método ‘verificarBinario’ anteriormente, portanto a função ‘realizarOperacao’’ tem como parâmetros as variáveis ‘resultado’, ‘numero’ e ‘operacao’, caso o número que o usuário digitar não for binário, irá resultar no else, que irá dar um ‘i–’, repetindo o for e voltando no mesmo índice.

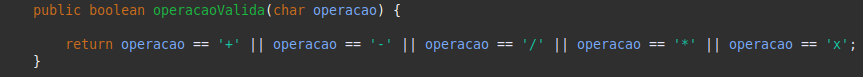


Após isso, o resultado final será exibido em uma variável tipo long em uma representação binária usando ‘Long.toBinaryString(resultado)’. Caso por algum motivo, mesmo assim uma exceção seja lançada, a aplicação irá exibir a mensagem "Erro inesperado" no console, evitando que seja mostrado detalhes fundamentais do código. Então teremos o bloco finally garantindo que o Scanner seja fechado.

-Pacote model.entities classe entidades ( Explicando metódos implementados ):

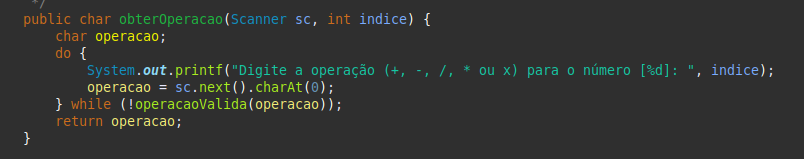


Verifica se o valor digitado pelo usuário é binário, se contém apenas os dígitos '0' e '1'. Convertendo o valor para uma string e usa uma expressão regular ([01]+) para garantir que a string contenha apenas '0' e '1'. Retorna true se for binário e false caso não seja.



Verifica se um caractere de operação digitado pelo usuário é válido. Ele retorna true se o caractere de operação for '+', '-', '/', '\*' ou 'x', e false caso não seja, ele será parte de outros métodos.

Recebe duas variáveis do tipo string representando números binários, converte eles para números decimais, e recebe uma variável do tipo char realizando a operação matemática com base na escolha do usuário (adição, subtração, multiplicação ou divisão), logo após converte o resultado de volta para uma string binária antes de retorná-la.

Solicita ao usuário inserir uma operação válida (+, -, \*, / ou x) para que seja feita a conta. Sendo usado a estrutura condicional de loop do-while e fazendo a chamada do método ‘operacaoValida’ como parâmetro, assim assegurando que o usuário digite uma entrada válida e o programa não quebre. Retornando o caractere de operação inserido pelo usuário.

# Github:

O projeto será postado no github, no seguinte link:

https://github.com/pedromeirelessz/ProjetoFaculdadeA3

# Youtube:

O vídeo do projeto será postado no youtube, no seguinte link:

https://www.youtube.com/watch?v=bGGdeYDyB4M