UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

PROFESSOR: DR. THIAGO IACHILEY ARAUJO DE SOUZA

JOÃO DAVI LIBERATO ALVES

ATIVIDADE 3: LABORATÓRIO DE TÉCNICAS DE PROGRAMAÇÃO

SOBRAL

2024

1 QUESTÃO 1

O problema proposto envolve o desenvolvimento de um programa que recebe um número do usuário e realiza algumas verificações. O objetivo é determinar se o número é par ou ímpar e se está entre os valores de 10 e 20, inclusive.

1.1 Descrição do problema

O objetivo desta questão é desenvolver um programa que:

a) verifique se um número inserido pelo usuário é par ou ímpar;
b) cheque se o número está no intervalo entre 10 e 20 (não incluindo os limites).

O programa deve imprimir uma mensagem apropriada com base nas condições avaliadas.

1.2 Implementação da Solução

A entrada do programa é feita através da classe Scanner, que permite ler o número digitado pelo usuário. A verificação de paridade é realizada utilizando o operador de módulo (%), onde um número é considerado par se o seu resto da divisão por 2 for igual a zero. Caso contrário, ele é ímpar.

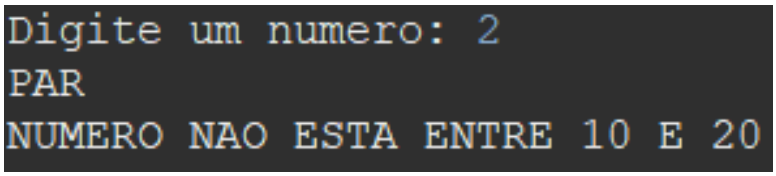
Adicionalmente, utiliza-se uma estrutura condicional para verificar se o número está no intervalo entre 10 e 20. Para isso, é empregada a expressão condicional num > 10 && num < 20, que retorna verdadeiro se o número estiver dentro do intervalo especificado e falso caso contrário.

A função close() é utilizada ao final do programa para fechar o objeto Scanner, liberando assim os recursos alocados para a entrada de dados.

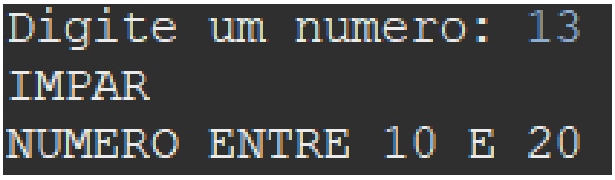
1.3 Resultados

Os resultados obtidos a partir da execução do programa são apresentados na forma de mensagens impressas ao usuário, que indicam se o número é par ou

ímpar e se está no intervalo entre 10 e 20. Abaixo, seguem exemplos de entradas e suas respectivas saídas:



Fonte: Terminal do autor.



Fonte: Terminal do autor.

2 QUESTÃO 2

Neste exercício, o objetivo é desenvolver um programa para realizar a soma de números inseridos pelo usuário até que o valor total seja superior ou igual a 100. Essa prática envolve o uso de estruturas de repetição, permitindo que o programa se repita de acordo com uma condição predeterminada. A questão é relevante para o aprendizado de controle de fluxo e iteração em programação.

2.1 Descrição do Problema

O problema consiste em solicitar ao usuário a inserção de números de forma contínua e calcular a soma acumulada desses números. O programa deve interromper a entrada de dados quando o valor da soma atingir ou ultrapassar 100. O objetivo principal é demonstrar a utilização de laços de repetição, neste caso, o laço do-while, que garante a execução do bloco de código pelo menos uma vez, independentemente da condição inicial.

2.2 Implementação da Solução

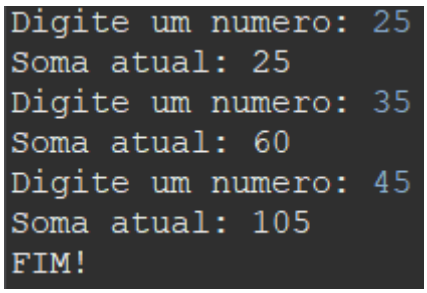
A implementação foi realizada, utilizando um laço de repetição do-while, que garante que o bloco de código seja executado ao menos uma vez antes de verificar a condição de parada. A cada iteração, o programa solicita ao usuário que insira um número, adicionando esse valor à variável soma, que acumula os valores inseridos.

O uso do laço do-while é apropriado nesse contexto, pois a condição de término depende do valor acumulado em soma. O programa continua solicitando números enquanto a soma total for menor que 100. A classe Scanner é novamente utilizada para realizar a entrada de dados do usuário.

As variáveis num e soma são utilizadas para armazenar, respectivamente, o número inserido e a soma acumulada dos valores fornecidos. O laço continua até que a soma seja igual ou superior a 100.

2.3 Resultados

Os resultados do programa são as somas intermediárias e a mensagem final, que indica o término da execução. A cada novo número inserido, o programa exibe a soma atual. Quando a soma total atinge ou excede 100, o laço é encerrado e o programa imprime "FIM!".



Fonte: Terminal do autor.

3 QUESTÃO 3

O objetivo deste exercício é implementar um programa que permita ao usuário inserir uma série de números e armazená-los em uma lista. A tarefa é relevante para demonstrar a utilização de coleções na linguagem Java, mais

especificamente a classe ArrayList, que oferece funcionalidades dinâmicas para a manipulação de listas.

3.1 Descrição do Problema

O problema consiste em solicitar ao usuário que insira 10 números inteiros, os quais serão armazenados em uma lista. Ao final das entradas, o programa deve exibir o conteúdo da lista. O objetivo principal é demonstrar o uso da coleção ArrayList e como é possível interagir com ela para armazenar e exibir dados inseridos pelo usuário.

3.2 Implementação da Solução

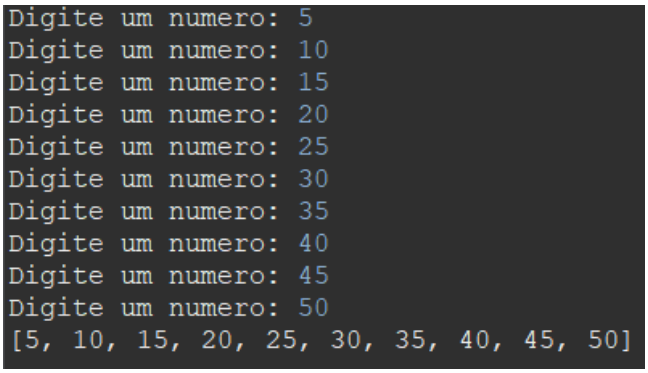
A solução foi implementada utilizando a classe ArrayList do pacote java.util, que oferece uma estrutura de dados flexível para armazenar valores de forma dinâmica. A coleta de dados é realizada através da classe Scanner, que permite a entrada de números pelo usuário.

A classe ArrayList é utilizada para armazenar de forma dinâmica os números inteiros fornecidos pelo usuário. A estrutura de repetição for foi empregada para solicitar a entrada de 10 números consecutivos. O método add() da classe ArrayList é usado para inserir cada número fornecido pelo usuário na lista.

Após a inserção dos números, o programa exibe a lista completa com os números fornecidos.

3.3 Resultados

O programa exibe a lista de números inseridos pelo usuário. Abaixo estão exemplos de execução do código:



Fonte: Terminal do autor.

4 QUESTÃO 4

Este exercício aborda a utilização de estruturas de dados dinâmicas e manipulação de entradas de usuário, especificamente o uso da classe ArrayList para armazenar e ordenar uma lista de strings (nomes) fornecidos pelo usuário.

4.1 Descrição do Problema

O objetivo da questão é solicitar ao usuário que insira cinco nomes, armazená-los em uma lista dinâmica e, posteriormente, exibir essa lista ordenada em ordem alfabética.

4.2 Implementação da Solução

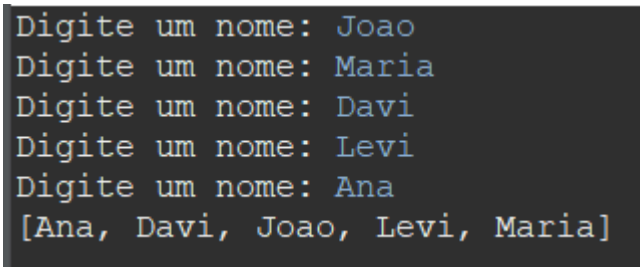
A implementação foi feita utilizando a classe ArrayList para armazenar os nomes inseridos pelo usuário. A estrutura de repetição for percorre cinco iterações, pedindo ao usuário que insira um nome em cada uma delas. O método add() da classe ArrayList é utilizado para adicionar os nomes na lista.

Após a coleta dos nomes, o método sort(null) da classe Collections é chamado para ordenar os elementos da lista em ordem alfabética. Finalmente, a lista ordenada é exibida. As principais funções são:

a) nextLine(): Lê a entrada do usuário como string.
b) add(): Adiciona cada nome fornecido à lista.
c) sort(null): Ordena a lista em ordem alfabética.

4.3 Resultados

O programa armazena e exibe os cinco nomes fornecidos pelo usuário, ordenados alfabeticamente. Ao final, é exibida a lista ordenada.



Fonte: Terminal do autor.

5 QUESTÃO 5

Este exercício foca na implementação de uma função que realiza o cálculo da média aritmética entre dois números. Através da leitura de valores fornecidos pelo usuário, a função calcula e retorna a média desses números.

5.1 Descrição do Problema

O objetivo desta questão é criar um programa que calcula a média aritmética de dois números. Para isso, é utilizada uma função chamada calcularMedia, que recebe dois parâmetros do tipo double e retorna a média entre eles.

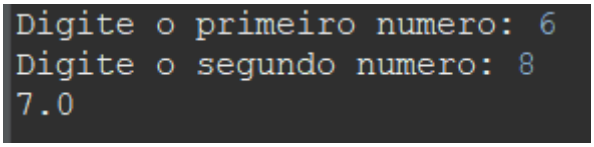
5.2 Implementação da Solução

A solução foi implementada com a função calcularMedia, que aceita dois argumentos (x e y) e retorna a média dos dois valores. No método principal (main), a função é chamada com dois números pré-definidos (3.5 e 6.5), e o resultado é exibido ao usuário por meio do método System.out.println(). As principais funções são:

a) calcularMedia(double x, double y): Essa função recebe dois números do tipo double, realiza a soma deles, e divide o resultado por 2, retornando a média aritmética.
b) println(): Exibe o resultado da média.

5.3 Resultados

O programa solicita ao usuário que insira dois números, calcula a média desses números e exibe o resultado.



Fonte: Terminal do autor.

6 QUESTÃO 6

O exercício tem como objetivo desenvolver um programa que permite ao usuário inserir uma série de números inteiros, armazena esses números em uma lista e calcula a soma dos números ímpares presentes na lista. O foco é na utilização da classe ArrayList e na manipulação de listas para realizar operações de filtragem e cálculo.

6.1 Descrição do Problema

O programa deve solicitar ao usuário que insira 10 números inteiros, armazenar esses números em uma lista e calcular a soma dos números ímpares. A solução requer o uso da classe ArrayList para armazenar os números e a implementação de uma função para calcular a soma dos números ímpares.

6.2 Implementação da Solução

O programa é estruturado em duas partes principais:

1. leitura e armazenamento dos Números**:** A classe Scanner é utilizada para capturar a entrada do usuário. Cada número inserido é adicionado à lista ArrayList utilizando o método add();
2. cálculo da Soma dos Números Ímpares: A função somaImpares percorre a lista e verifica quais números são

ímpares. A soma desses números ímpares é então calculada e retornada pela função.

6.3 Resultados

O programa solicita ao usuário a inserção de 10 números inteiros e os armazena em uma lista. Após a inserção, o programa calcula a soma dos números ímpares presentes na lista e exibe o resultado. O usuário pode visualizar a soma total dos números ímpares.

```
Digite um numero: 1
Digite um numero: 2
Digite um numero: 3
Digite um numero: 4
Digite um numero: 5
Digite um numero: 6
Digite um numero: 7
Digite um numero: 8
Digite um numero: 9
Digite um numero: 10
A soma dos numeros impares eh: 25
```

Fonte: Terminal do autor.

7 QUESTÃO 7

O programa tem como objetivo ler uma sequência de 10 números inteiros fornecidos pelo usuário, categorizar esses números em positivos, negativos e zeros, e verificar se há números primos na lista. O uso de laços de repetição, condicionais e uma função para identificar números primos são elementos fundamentais da solução.

7.1 Descrição do Problema

A questão exige que o programa conte quantos números positivos, negativos e zeros foram inseridos pelo usuário, e ainda verifique se algum dos números é primo. O programa deve então exibir o resultado dessas contagens e informar se existe ou não um número primo entre os valores fornecidos.

7.2 Implementação da Solução

O código utiliza a classe Scanner para obter a entrada do usuário e uma ArrayList para armazenar os números inseridos. A lógica principal é dividida da seguinte forma:

1. classificação dos números: Para cada número, o programa verifica se ele é positivo, negativo ou zero e incrementa os respectivos contadores;
2. verificação de números primos: A função numeroPrimo recebe um número inteiro e retorna true se o número for primo. A função utiliza um loop que vai de 2 até a raiz quadrada do número para verificar se ele possui divisores além de 1 e ele mesmo;
3. exibição dos resultados: Após processar os números, o programa imprime as quantidades de números positivos, negativos e zeros, e informa se há um número primo na lista.

7.3 Resultados

Após a execução do programa, o usuário é solicitado a inserir 10 números inteiros. Após a entrada desses valores, o programa realiza a categorização dos números fornecidos, identificando a quantidade de números positivos, negativos e zeros. Além disso, o programa verifica se há algum número primo na lista inserida pelo usuário e informa o resultado.

```
Digite um numero: 4
Digite um numero: 6
Digite um numero: 20
Digite um numero: 0
Digite um numero: -3
Digite um numero: 5
Digite um numero: 0
Digite um numero: -1
Digite um numero: -7
Digite um numero: 24
Quantidade de positivos: 5
Quantidade de negativos: 3
Quantidade de zeros: 2
Tem numero primo na lista
```

Fonte: Terminal do autor.

8 QUESTÃO 8

Este programa tem como objetivo calcular o fatorial de um número inteiro fornecido pelo usuário. No entanto, quando um número negativo é inserido, o programa solicita ao usuário que insira um número positivo, visto que não é possível calcular o fatorial de números negativos.

8.1 Descrição do Problema

O fatorial de um número é obtido multiplicando-se o número pelos seus antecessores até 1. Matematicamente, o fatorial de um número n é representado por n! e definido como $n!=n\times(n-1)\times...\times1$. O problema aborda a impossibilidade de calcular o fatorial de números negativos e propõe uma solução que permite ao usuário corrigir a entrada.

8.2 Implementação da Solução

O programa utiliza uma estrutura de controle que solicita a entrada de um número e calcula o fatorial se for um número positivo. Caso o número seja negativo, ele solicita repetidamente a entrada de um número válido. A função principal utilizada é a calcularFatorial, que realiza o cálculo de forma iterativa. A função realiza as seguintes etapas:

1. Verifica se o número inserido é negativo.
2. Solicita a correção da entrada se o número for negativo.
3. Calcula o fatorial caso o número seja positivo ou zero.

As principais funções são**:**

a) calcularFatorial(int num): Realiza o cálculo do fatorial de um número positivo e, caso o número inserido seja negativo, solicita uma nova entrada até que um número válido seja inserido;
b) sc.nextInt(): Realiza a leitura da entrada do usuário;
c) System.out.println(): Exibe o resultado final do cálculo do fatorial.

8.3 Resultados

Após a execução do programa, caso o usuário insira um número negativo, o programa solicita repetidamente que ele insira um número positivo. Quando um número positivo ou zero é inserido, o fatorial é calculado corretamente e exibido para o usuário.

```
Digite um numero inteiro nao negativo para calcular seu fatorial: -4
Nao existe fatorial de numero negativo. Tente novamente.
Digite um numero positivo: 4
24
```

Fonte: Terminal do autor.

9 QUESTÃO 9

O objetivo desta atividade é implementar um programa que receba 10 números inteiros fornecidos pelo usuário, calcule o cubo de cada um deles e exiba o resultado em uma nova lista. A implementação é realizada, utilizando estruturas de lista e laços de repetição para processar os números.

9.1 Descrição do Problema

O problema consiste em solicitar 10 números inteiros do usuário, calcular o cubo de cada número e exibir uma lista contendo os resultados. O cubo de um número é obtido elevando-o à terceira potência. O programa deve processar esses números de forma eficiente e exibir o resultado em uma estrutura de lista.

9.2 Implementação da Solução

A implementação do programa utiliza as seguintes principais funções:

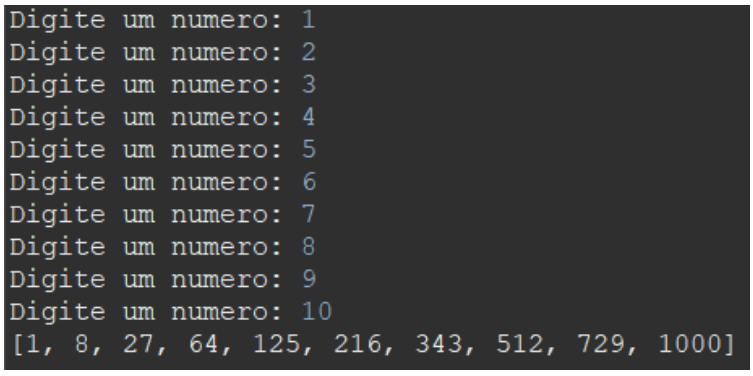
a) listaCubo(List<Integer> lista): Função que percorre a lista original, calcula o cubo de cada elemento e retorna uma nova lista contendo os resultados;

b) Math.pow(): Função da biblioteca Java utilizada para elevar os números à potência de três;

c) o programa utiliza a classe Scanner para capturar a entrada do usuário, que insere os números inteiros. Em seguida, os números são armazenados em uma lista, e o método listaCubo() é responsável por gerar uma nova lista contendo os cubos dos números.

9.3 Resultados

O programa solicita ao usuário a inserção de 10 números inteiros e, ao final, exibe uma lista com o valor do cubo de cada número inserido.



Fonte: Terminal do autor.

10 QUESTÃO 10

O problema proposto consiste em implementar um programa que leia três números inteiros fornecidos pelo usuário, identifique o maior entre eles e exiba o resultado. A implementação lida com a possibilidade de números iguais, solicitando ao usuário a reinserção dos valores até que todos os números sejam distintos.

10.1 Descrição do Problema

O objetivo é comparar três números inteiros inseridos pelo usuário e identificar o maior valor entre eles. Além disso, caso haja números iguais, o programa deve solicitar que o usuário insira novamente os números até que todos sejam diferentes, garantindo assim uma comparação válida.

10.2 Implementação da Solução

As principais funções utilizadas na implementação são:

a) maiorNumero(int x, int y, int z): Função responsável por verificar qual dos três números é o maior. Caso sejam inseridos números iguais, essa função solicita ao usuário que insira novos valores até que os números sejam distintos;
b) nextInt(): Utilizada para capturar a entrada de números inteiros fornecida pelo usuário.

A lógica implementada primeiro verifica se os números inseridos são distintos. Se houver números iguais, o programa entra em um laço de repetição para solicitar novas entradas. Após garantir que todos os números são diferentes, a função realiza a comparação e retorna o maior valor.

10.3 Resultados

O programa solicita ao usuário três números inteiros e exibe o maior número entre eles. Caso o usuário insira números iguais, o programa pede que ele forneça novos valores, garantindo que a comparação seja feita apenas entre números distintos. O resultado final é a exibição do maior número.

```
Digite o primeiro numero:
3
Digite o segundo numero:
3
Digite o terceiro numero:
5
Ha numeros iguais. Digite novamente.
Digite o primeiro numero: 3
Digite o segundo numero: -3
Digite o terceiro numero: 0
O maior numero eh: 3
```

Fonte: Terminal do autor.

11 QUESTÃO 11

O problema proposto consiste em criar um programa que simule o cadastro de um funcionário, capturando seu nome e salário bruto. O programa também permite que o usuário aplique um aumento percentual sobre o salário e, em seguida, exiba as informações atualizadas.

11.1 Descrição do Problema

O objetivo é capturar o nome e o salário bruto de um funcionário e, em seguida, permitir que o usuário insira uma porcentagem de aumento. Após o aumento ser aplicado, o programa deve exibir os dados atualizados do funcionário, incluindo o novo salário.

11.2 Implementação da Solução

As principais funções utilizadas na implementação são:

a) nextLine(): Utilizada para capturar o nome do funcionário, permitindo que o programa leia uma sequência de caracteres inserida pelo usuário;
b) nextDouble(): Utilizada para capturar o salário bruto do funcionário e a porcentagem de aumento fornecida pelo usuário.

O salário é ajustado aplicando a fórmula de aumento percentual, onde o valor inserido pelo usuário é transformado em uma fração e somado ao valor original do salário.

O programa é estruturado para primeiro capturar o nome e o salário bruto do funcionário. Após isso, o usuário insere a porcentagem de aumento, que é aplicada diretamente ao salário. Em seguida, as informações atualizadas são exibidas, incluindo o novo valor do salário bruto.

11.3 Resultados

O programa exibe as informações do funcionário antes e após a aplicação do aumento salarial.

```
====CADASTRO DE FUNCIONARIO====

Digite seu nome: joao
Digite seu salario bruto: 1234

====DADOS DO FUNCIONARIO====
Nome: joao
Salario bruto: 1234.0

Quanto voce deseja aumentar o seu salario? (em porcentagem) 100

====DADOS ATUALIZADOS DO FUNCIONARIO====
Nome: joao
Salario bruto: 2468.0
```

Fonte: Terminal do autor.

12 QUESTÃO 12

Este progama descreve a implementação de um programa que simula o registro de produtos em uma loja e realiza o cálculo do valor total dos itens em estoque. O programa permite cadastrar produtos, exibir suas informações e calcular o valor total dos produtos disponíveis.

12.1 Descrição do Problema

O objetivo do programa é gerenciar o estoque de uma loja. Para isso, é necessário cadastrar produtos com seus respectivos nomes e preços, exibir os produtos registrados e calcular o valor total dos itens em estoque.

12.1 Implementação da Solução

A implementação utiliza duas classes principais: Produto e Loja. A classe produto representa cada item no estoque. Ela contém os atributos nome e preco, além de métodos para exibir as informações do produto e retornar seus valores. As principais funções:

a) getPreco(): Retorna o preço do produto;
b) exibirInfo(): Exibe o nome e o preço do produto no formato adequado.

A classe Loja representa a loja, que contém uma lista de produtos. Ela possui métodos para registrar novos produtos, exibir os produtos cadastrados e calcular o valor total do estoque. As principais funções são:

a) registrarProduto(Produto produto): Adiciona um produto à lista de produtos da loja;
b) exibirProdutos(): Exibe a lista completa de produtos cadastrados, utilizando o método exibirInfo() de cada produto;
c) calcularEstoque(): Soma o preço de todos os produtos cadastrados na lista e retorna o valor total do estoque.

12.3 Resultados

Após cadastrar quatro produtos, o programa exibe os detalhes de cada um, incluindo nome e preço. Em seguida, ele calcula e exibe o valor total do estoque, somando os preços de todos os produtos registrados.

```
====== PRODUTOS EM ESTOQUE ======
Nome: Geladeira
Preco: 1499.99

Nome: Fogao
Preco: 799.99

Nome: Microondas
Preco: 399.99

Nome: Televis□o
Preco: 2599.99

Total em estoque: 5299.96
```

Fonte: Terminal do autor.

13 QUESTÃO 13

Este programa aborda a implementação de um sistema simples para registrar e exibir informações de reservas de viagem. O programa desenvolvido permite criar reservas associadas a um nome, um destino e um número de reserva, exibindo esses dados de forma clara para o usuário.

13.1 Descrição do Problema

O objetivo é permitir que o usuário cadastre reservas de viagem, fornecendo um nome, destino e número de reserva. Após o registro, as informações de cada reserva devem ser exibidas. O foco do programa é garantir que os dados das reservas sejam corretamente armazenados e visualizados.

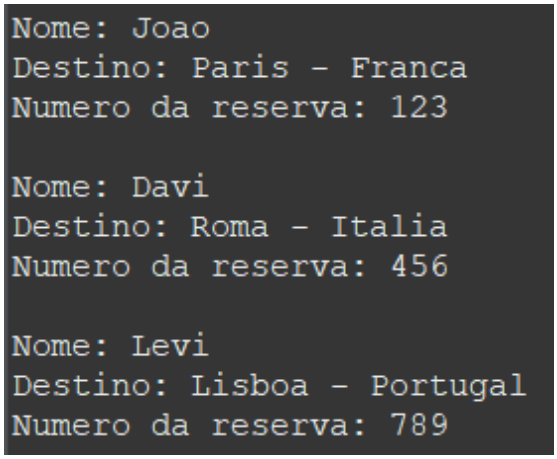
13.2 Implementação da Solução

A implementação utiliza a classe Reserva, que representa cada reserva de viagem. Esta classe contém três atributos principais: nome, destino e numeroReserva, que armazenam os dados da reserva. A classe também possui um método para exibir as informações da reserva.

O método exibirReserva() **e**xibe as informações da reserva, como o nome do titular, o destino e o número da reserva.

13.3 Resultados

O programa cria três reservas com diferentes destinos e números de reserva. Após o cadastro, as informações de cada reserva são exibidas individualmente.



Fonte: Terminal do autor.

14 QUESTÃO 14

Este programa descreve a implementação de um sistema simples para gerenciar empréstimos de livros. O sistema desenvolvido tem como objetivo controlar o estado dos livros (disponível ou emprestado) e exibir informações sobre eles.

14.1 Descrição do Problema

O sistema deve permitir que livros sejam cadastrados com um nome e autor. Ele deve controlar o status de cada livro, indicando se está disponível ou emprestado. Além disso, o programa deve possibilitar a exibição das informações do livro e a mudança de status em caso de empréstimo ou devolução.

14.2 Implementação da Solução

A implementação é feita por meio da classe Livro, que contém atributos e métodos para gerenciar o status e as ações relacionadas aos livros. As principais funções são:

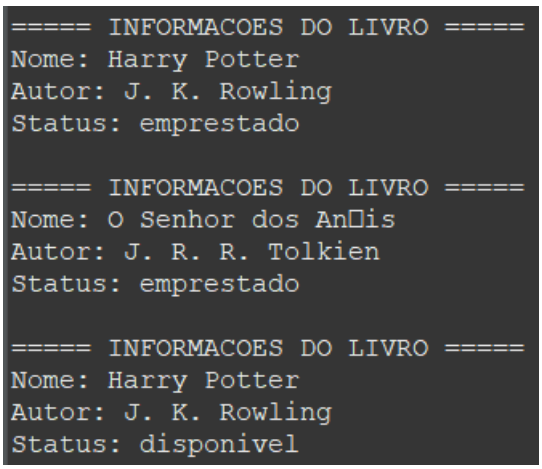
a) emprestarLivro(): Altera o status do livro para "emprestado" caso o livro esteja disponível. Caso contrário, exibe uma mensagem indicando que o livro não está disponível para empréstimo;

b) devolverLivro(): Altera o status do livro para "disponível" se ele estiver emprestado. Caso contrário, informa que o livro já está disponível;

c) exibirLivro(): Exibe as informações do livro, como o nome, autor e status atual.

14.3 Resultados

O sistema permite o controle adequado do status dos livros. Ao tentar emprestar um livro, o programa verifica se ele está disponível, alterando o status quando apropriado. Da mesma forma, ao devolver um livro, o status é atualizado corretamente.



Fonte: Terminal do autor.

15 QUESTÃO 15

Este programa descreve a implementação de um sistema básico de gestão de pedidos. O sistema visa armazenar informações sobre os pedidos de clientes, como nome, item solicitado e a quantidade de cada item. Além disso, permite a exibição das informações de cada pedido de forma estruturada.

15.1 Descrição do Problema

O objetivo do sistema é registrar e exibir pedidos feitos por clientes. Cada pedido contém o nome do cliente, o item solicitado e a quantidade desejada. O sistema deve ser capaz de armazenar e exibir essas informações de maneira clara e organizada.

15.2 Implementação da Solução

A implementação é feita através da classe Pedido, que contém os atributos e métodos necessários para armazenar e gerenciar as informações de cada pedido. O método exibirPedido() exibe as informações do pedido, incluindo o nome do cliente, o item solicitado e a quantidade. Ele é responsável por organizar e formatar essas informações para serem exibidas na saída do sistema.

15.3 Resultados

O sistema exibe corretamente os detalhes de cada pedido feito por diferentes clientes. A funcionalidade foi testada com três exemplos de pedidos, incluindo diferentes itens e quantidades.

```
===== INFORMACOES DO PEDIDO =====
Nome: Joao
Item: Cachorro-quente
Quantidade: 3

===== INFORMACOES DO PEDIDO =====
Nome: Maria
Item: Hamburguer
Quantidade: 2

===== INFORMACOES DO PEDIDO =====
Nome: Carlos
Item: Pizza
Quantidade: 1
```

Fonte: Terminal do autor.

16 QUESTÃO 16

Este programa apresenta a implementação de um sistema de gerenciamento de contas bancárias. O sistema permite o cadastro de contas, depósitos, saques e exibição do saldo, simulando operações comuns de uma conta bancária.

16.1 Descrição do Problema

O objetivo do sistema é gerenciar contas bancárias, permitindo que o usuário deposite, saque e visualize o saldo de sua conta. Para garantir a integridade das operações, o sistema deve verificar se há saldo suficiente antes de permitir saques.

16.2 Implementação da Solução

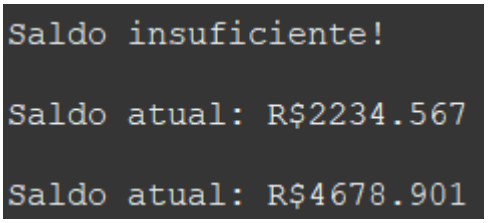
A implementação foi realizada com a criação da classe ContaBancaria, que encapsula os dados e operações bancárias. As principais funções são:

a) depositar(double valor): A função recebe um valor e o adiciona ao saldo da conta;
b) sacar(double valor): Verifica se o saldo é suficiente antes de deduzir o valor do saque, emitindo uma mensagem caso o saldo seja insuficiente;

c) exibirSaldo(): Exibe o saldo atual da conta.

16.3 Resultados

O sistema permite realizar depósitos e saques nas contas bancárias cadastradas. No exemplo, duas contas foram criadas, e suas operações demonstram o funcionamento adequado do sistema. Quando um saque excede o saldo disponível, o sistema informa que o saldo é insuficiente.



Fonte: Terminal do autor.

17 QUESTÃO 17

Este relatório aborda o desenvolvimento de um sistema de aluguel de veículos. O sistema foi projetado para gerenciar a disponibilidade de veículos, permitindo seu aluguel, devolução e a exibição de informações detalhadas sobre os veículos cadastrados.

17.1 Descrição do Problema

O objetivo deste sistema é gerenciar o aluguel de veículos, assegurando que um veículo possa ser alugado apenas quando estiver disponível e possa ser devolvido após o aluguel. Além disso, o sistema deve fornecer ao usuário as informações atualizadas sobre o status dos veículos.

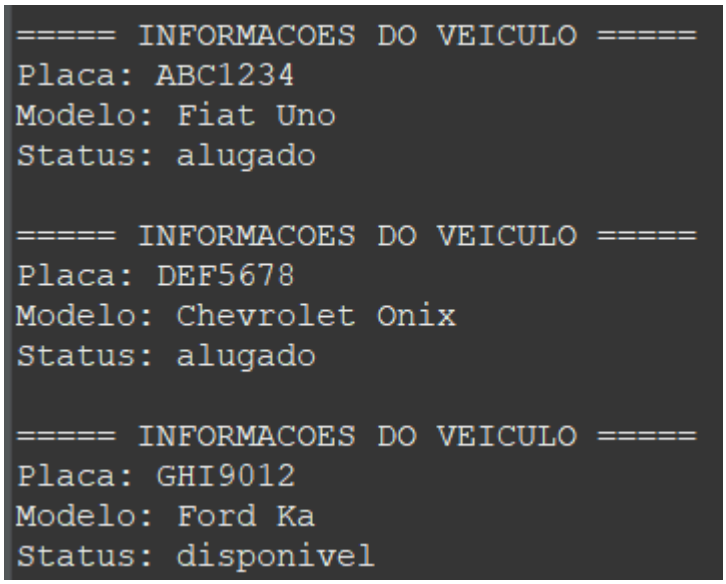
17.2 Implementação da Solução

A solução foi implementada por meio da classe Veiculo, que encapsula os dados e as operações necessárias para o aluguel e devolução dos veículos. As principais funções são:

a) alugar(): Verifica se o veículo está disponível. Se sim, altera o status para "alugado". Caso contrário, informa que o veículo está indisponível;
b) devolver(): Verifica se o veículo está alugado. Se sim, altera o status para "disponível". Caso contrário, informa que o veículo já está disponível;
c) exibirDetalhes(): Exibe as informações do veículo, incluindo a placa, modelo e o status atual (disponível ou alugado).

17.3 Resultados

O sistema permite o gerenciamento de três veículos cadastrados, mostrando o status de cada um após as operações de aluguel e exibição de informações. Quando um veículo é alugado, o status é alterado corretamente para "alugado". Se o veículo já estiver alugado, o sistema informa que ele está indisponível para aluguel. Ao devolver, o status retorna para "disponível".



Fonte: Terminal do autor.

18 QUESTÃO 18

Este programa apresenta o desenvolvimento de um sistema de reserva de quartos, implementado em Java. O objetivo do sistema é gerenciar a disponibilidade de quartos, permitindo sua reserva, liberação e a exibição de informações detalhadas sobre os quartos cadastrados.

18.1 Descrição do Problema

O sistema tem como propósito garantir que os quartos de um hotel ou similar possam ser reservados quando disponíveis e liberados após o uso, com controle preciso de status e exibição das informações dos quartos.

18.2 Implementação da Solução

A classe Quarto foi criada para encapsular os dados e as operações necessárias para gerenciar a reserva e liberação de quartos. As principais funções são:

a) reservar(): Verifica se o quarto está disponível. Se estiver, altera o status para "reservado". Se já estiver reservado, o sistema informa que o quarto está indisponível para reserva;
b) liberar(): Verifica se o quarto está reservado. Se estiver, altera o status para "disponível". Caso contrário, informa que o quarto já está disponível;
c) exibirDetalhes(): Exibe as informações do quarto, incluindo o número, tipo (simples ou duplo) e o status atual (disponível ou reservado).

18.3 Resultados

Dois quartos foram cadastrados e gerenciados pelo sistema. Após a tentativa de reservar ambos os quartos, o sistema alterou corretamente o status para "reservado" e exibiu as informações detalhadas de cada um.

```
===== INFORMACOES DO QUARTO =====
Numero: 123
Tipo: simples
Status: reservado

===== INFORMACOES DO QUARTO =====
Numero: 456
Tipo: duplo
Status: reservado
```

Fonte: Terminal do autor.

19 QUESTÃO 19

Este programa descreve a implementação de um sistema de gerenciamento de eventos em Java. O objetivo do sistema é permitir a criação de eventos, a alteração do local e a exibição dos detalhes de cada evento.

19.1 Descrição do Problema

O sistema visa auxiliar na organização e controle de eventos, oferecendo funcionalidades para alterar o local de um evento já cadastrado e exibir informações detalhadas, como o nome, local e data do evento.

19.2 Implementação da Solução

A classe Evento foi desenvolvida para representar cada evento, com os atributos principais: nome, local e data. A classe também oferece métodos para alterar o local do evento e exibir suas informações. Os métodos são:

a) alterarLocal(String novoLocal): Permite modificar o local do evento.
b) exibirDetalhes(): Exibe as informações detalhadas do evento, como nome, local e data.

19.3 Resultados

Dois eventos foram cadastrados e gerenciados pelo sistema. Após alterar o local do evento 1 para "Universidade" e do evento 2 para "Sítio", o sistema exibiu corretamente os novos detalhes dos eventos.

```
==== DETALHES DO EVENTO ====
Nome: Formatura
Local: Universidade
Data: 23/12/2024

==== DETALHES DO EVENTO ====
Nome: Casamento
Local: SItio
Data: 25/12/2024
```

Fonte: Terminal do autor

20 QUESTÃO 20

Este programa descreve a implementação de um sistema para gerenciamento de sessões de cinema. O sistema possibilita a venda de ingressos e a exibição dos detalhes das sessões disponíveis.

20.1 Descrição do Problema

O sistema tem como objetivo gerenciar sessões de cinema, controlando a venda de ingressos de acordo com a disponibilidade e exibindo informações detalhadas sobre cada sessão.

20.2 Implementação da Solução

A classe SessaoCinema foi criada para representar uma sessão de cinema, com os atributos principais: filme, horario e ingressosDisponiveis. A classe também inclui métodos para vender ingressos e exibir as informações da sessão. Esses métodos são:

a) venderIngresso(int quantidade): Verifica se há ingressos disponíveis para a quantidade solicitada e realiza a venda, atualizando o número de ingressos restantes. Caso a

quantidade solicitada seja maior que a disponível, uma mensagem de erro é exibida;

b) exibirDetalhes(): Exibe as informações detalhadas da sessão, incluindo o nome do filme, horário e a quantidade de ingressos disponíveis.

20.3 Resultados

Duas sessões de cinema foram cadastradas e gerenciadas pelo sistema. Após realizar a venda de 4 ingressos para a sessão de "Star Wars", o sistema atualizou corretamente a quantidade de ingressos restantes e exibiu as informações detalhadas. No entanto, ao tentar vender 5 ingressos para a sessão de "Vingadores", o sistema identificou que a quantidade disponível era insuficiente e exibiu a mensagem de erro apropriada.

```
Ingresso(s) vendido com sucesso!
===== INFORMACOES DA SESSAO =====
Filme: Star Wars
Horario: 19:00
Ingressos disponiveis: 26

Nao ha ingressos suficientes disponiveis!
===== INFORMACOES DA SESSAO =====
Filme: Vingadores
Horario: 21:00
Ingressos disponiveis: 3
```

Fonte: Terminal do autor.