INSTITUTO INFNET

ESCOLA SUPERIOR DE TECNOLOGIA

GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

AT1

Aluno: Enzo Furtini

1 de abr. de 2025

https://github.com/G-itch/Enzo_Furtini_DR1_AT

Exercícios Java

Este repositório contém uma coleção de exercícios práticos em Java, desenvolvidos para demonstrar diferentes conceitos da linguagem.

Primeiro Programa

Enunciado: Crie um programa que imprima uma mensagem de saudação personalizada.

Implementação:

```
public class OlaMundo {
    public static void main(String[] args) {
        System.out.println("Olá, meu nome é [Seu Nome] e estou aprendendo
Java!");
    }
}
```

Validador de Senha

Enunciado: Desenvolva um sistema que valide senhas fortes com os seguintes requisitos:

- Mínimo de 8 caracteres
- Pelo menos uma letra maiúscula
- Pelo menos um número

- Pelo menos um caractere especial

Implementação:

```
public class ValidadorSenha {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String nome, senha;
        boolean senhaValida = false;

        System.out.print("Digite seu nome: ");
        nome = scanner.nextLine();

        while (!senhaValida) {
            System.out.print("Digite uma senha: ");
            senha = scanner.nextLine();

            // Validação da senha
            if (senha.length() < 8) {
                System.out.println("Erro: A senha deve ter no mínimo 8
caracteres.");
                continue;
            }

            // Verifica maiúsculas, números e caracteres especiais
            boolean temMaiuscula = false;
            boolean temNumero = false;
            boolean temEspecial = false;

            for (char c : senha.toCharArray()) {
                if (Character.isUpperCase(c)) temMaiuscula = true;
                else if (Character.isDigit(c)) temNumero = true;
                else if (!Character.isLetterOrDigit(c)) temEspecial = true;
            }

            if (!temMaiuscula || !temNumero || !temEspecial) {
                System.out.println("Erro: A senha deve conter maiúsculas,
números e caracteres especiais.");
                continue;
            }

            senhaValida = true;
            System.out.println("Senha válida! Bem-vindo, " + nome + "!");
        }
    }
}
```

Calculadora de Impostos

Enunciado: Crie um programa que calcule o imposto de renda baseado na seguinte tabela:

- Até R$ 22.847,76 → Isento
- De R$ 22.847,77 a R$ 33.919,80 → 7,5%
- De R$ 33.919,81 a R$ 45.012,60 → 15%
- Acima de R$ 45.012,61 → 27,5%

Implementação:

```
public class CalculadoraImpostos {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Digite seu nome: ");
        String nome = scanner.nextLine();

        System.out.print("Digite seu salário mensal: R$ ");
        double salarioMensal = scanner.nextDouble();

        double salarioAnual = salarioMensal * 12;
        double imposto = 0;
        double aliquota = 0;

        if (salarioAnual <= 22847.76) {
            imposto = 0;
            aliquota = 0;
        } else if (salarioAnual <= 33919.80) {
            imposto = salarioAnual * 0.075;
            aliquota = 7.5;
        } else if (salarioAnual <= 45012.60) {
            imposto = salarioAnual * 0.15;
            aliquota = 15.0;
        } else {
            imposto = salarioAnual * 0.275;
            aliquota = 27.5;
        }

        System.out.printf("\nResultado para %s:%n", nome);
        System.out.printf("Salário Anual: R$ %.2f%n", salarioAnual);
        System.out.printf("Alíquota: %.1f%%%n", aliquota);
        System.out.printf("Imposto a pagar: R$ %.2f%n", imposto);
    }
}
```

Simulador de Empréstimo

Enunciado: Desenvolva um simulador de empréstimo que:

- Solicite o valor do empréstimo
- Permita escolher o número de parcelas (6 a 48)
- Calcule juros mensais de 3%

- Mostre o valor total e o valor da parcela

Implementação:

```
public class SimuladorEmprestimo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Digite o nome do cliente: ");
        String nome = scanner.nextLine();

        System.out.print("Digite o valor do empréstimo: R$ ");
        double valorEmprestimo = scanner.nextDouble();

        int parcelas;
        do {
            System.out.print("Digite o número de parcelas (6 a 48): ");
            parcelas = scanner.nextInt();
        } while (parcelas < 6 || parcelas > 48);

        double taxaJuros = 0.03;
        double valorParcela = calcularParcela(valorEmprestimo, taxaJuros,
parcelas);
        double valorTotal = valorParcela * parcelas;

        System.out.printf("\nSimulação para %s:%n", nome);
        System.out.printf("Valor do Empréstimo: R$ %.2f%n", valorEmprestimo);
        System.out.printf("Número de Parcelas: %d%n", parcelas);
        System.out.printf("Taxa de Juros Mensal: %.1f%%%n", taxaJuros * 100);
        System.out.printf("Valor da Parcela: R$ %.2f%n", valorParcela);
        System.out.printf("Valor Total: R$ %.2f%n", valorTotal);
    }
}
```

Simulador CGI

Enunciado: Crie um programa que simule a saída de um script CGI, gerando uma resposta HTTP válida com headers e conteúdo HTML.

Implementação:

```
public class ProgramaCGI {
    public static void main(String[] args) {
        System.out.println("Content-Type: text/html");
        System.out.println();

        System.out.println("<html>");
        System.out.println("<head><title>Saudação CGI</title></head>");
        System.out.println("<body>");
        System.out.println("<h1>Olá, Terráqueos!</h1>");
```

```
        System.out.println("</body>");
        System.out.println("</html>");
    }
}
```

Gestão de Frota

Enunciado: Desenvolva um sistema para gerenciar veículos com:

- Cadastro de veículos (placa, modelo, ano, quilometragem)
- Registro de viagens
- Exibição de detalhes

Implementação:

```
public class Veiculo {
    private String placa;
    private String modelo;
    private int anoFabricacao;
    private double quilometragem;

    public Veiculo(String placa, String modelo, int anoFabricacao, double
quilometragem) {
        this.placa = placa;
        this.modelo = modelo;
        this.anoFabricacao = anoFabricacao;
        this.quilometragem = quilometragem;
    }

    public void exibirDetalhes() {
        System.out.println("\nDetalhes do Veículo:");
        System.out.println("Placa: " + placa);
        System.out.println("Modelo: " + modelo);
        System.out.println("Ano: " + anoFabricacao);
        System.out.printf("Quilometragem: %.2f km%n", quilometragem);
    }

    public void registrarViagem(double km) {
        quilometragem += km;
        System.out.printf("Viagem registrada! Nova quilometragem: %.2f km%n",
quilometragem);
    }
}
```

Sistema Acadêmico

Enunciado: Crie um sistema para gerenciar notas de alunos com:

- Cadastro de alunos (nome, matrícula, notas)
- Cálculo de média

- Verificação de aprovação

Implementação:

```
public class Aluno {
    private String nome;
    private String matricula;
    private double nota1;
    private double nota2;
    private double nota3;

    public Aluno(String nome, String matricula, double nota1, double nota2,
double nota3) {
        this.nome = nome;
        this.matricula = matricula;
        this.nota1 = nota1;
        this.nota2 = nota2;
        this.nota3 = nota3;
    }

    public double calcularMedia() {
        return (nota1 + nota2 + nota3) / 3;
    }

    public void verificarAprovacao() {
        double media = calcularMedia();
        System.out.printf("\nMédia do aluno %s: %.2f%n", nome, media);
        if (media >= 7) {
            System.out.println("Situação: APROVADO");
        } else {
            System.out.println("Situação: REPROVADO");
        }
    }
}
```

Gestão de RH

Enunciado: Desenvolva um sistema de gestão de funcionários com:

- Classe base Funcionario
- Subclasses Gerente (bônus 20%) e Estagiário (desconto 10%)
- Cálculo de salário final

Implementação:

```
public class Funcionario {
    protected String nome;
    protected double salarioBase;

    public Funcionario(String nome, double salarioBase) {
```

```
        this.nome = nome;
        this.salarioBase = salarioBase;
    }

    public double calcularSalario() {
        return salarioBase;
    }
}

class Gerente extends Funcionario {
    public Gerente(String nome, double salarioBase) {
        super(nome, salarioBase);
    }

    @Override
    public double calcularSalario() {
        return salarioBase * 1.20;
    }
}

class Estagiario extends Funcionario {
    public Estagiario(String nome, double salarioBase) {
        super(nome, salarioBase);
    }

    @Override
    public double calcularSalario() {
        return salarioBase * 0.90;
    }
}
```

Sistema Bancário

Enunciado: Crie um sistema bancário com:

- Conta bancária (titular, saldo)
- Operações de depósito e saque
- Histórico de transações
- Validações de saldo

Implementação:

```
public class ContaBancaria {
    private String titular;
    private double saldo;
    private List<String> historico;

    public ContaBancaria(String titular, double saldoInicial) {
        this.titular = titular;
```

```
        this.saldo = saldoInicial;
        this.historico = new ArrayList<>();
    }

    public void depositar(double valor) {
        if (valor > 0) {
            saldo += valor;
            registrarOperacao("Depósito", valor);
        }
    }

    public void sacar(double valor) {
        if (valor > 0 && valor <= saldo) {
            saldo -= valor;
            registrarOperacao("Saque", valor);
        }
    }

    private void registrarOperacao(String tipo, double valor) {
        String operacao = String.format("[%s] %s: R$ %.2f - Saldo: R$ %.2f",
            LocalDateTime.now(), tipo, valor, saldo);
        historico.add(operacao);
    }
}
```

Controle de Vendas

Enunciado: Desenvolva um sistema para registrar compras com:

- Cadastro de produtos (nome, quantidade, preço)
- Salvamento em arquivo
- Leitura e exibição do histórico

Implementação:

```
public class RegistroCompras {
    public static void main(String[] args) {
        try (FileWriter writer = new FileWriter("compras.txt")) {
            for (int i = 1; i <= 3; i++) {
                System.out.println("\nRegistro da Compra " + i);
                System.out.print("Produto: ");
                String produto = scanner.nextLine();

                System.out.print("Quantidade: ");
                int quantidade = scanner.nextInt();

                System.out.print("Preço unitário: R$ ");
                double precoUnitario = scanner.nextDouble();

                double total = quantidade * precoUnitario;
```

```
                writer.write(String.format("Compra %d:%n", i));
                writer.write("Produto: " + produto + "\n");
                writer.write("Quantidade: " + quantidade + "\n");
                writer.write(String.format("Total: R$ %.2f%n", total));
                writer.write("-----------------------\n");
            }
        } catch (IOException e) {
            System.out.println("Erro ao escrever no arquivo: " +
e.getMessage());
        }
    }
}
```

Simulador de Loteria

Enunciado: Crie um jogo de loteria que:

- Gere 6 números aleatórios
- Permita apostar 6 números
- Compare e mostre os acertos
- Exiba mensagens personalizadas

Implementação:

```
public class SimuladorLoteria {
    public static void main(String[] args) {
        Random random = new Random();
        List<Integer> numerosSorteados = new ArrayList<>();

        while (numerosSorteados.size() < 6) {
            int numero = random.nextInt(60) + 1;
            if (!numerosSorteados.contains(numero)) {
                numerosSorteados.add(numero);
            }
        }

        List<Integer> numerosApostados = new ArrayList<>();
        System.out.println("Digite 6 números diferentes entre 1 e 60:");

        while (numerosApostados.size() < 6) {
            int numero = scanner.nextInt();
            if (numero >= 1 && numero <= 60 &&
!numerosApostados.contains(numero)) {
                numerosApostados.add(numero);
            }
        }

        List<Integer> acertos = new ArrayList<>();
```

```
        for (int numero : numerosApostados) {
            if (numerosSorteados.contains(numero)) {
                acertos.add(numero);
            }
        }

        System.out.println("\nResultados:");
        System.out.println("Números sorteados: " + numerosSorteados);
        System.out.println("Seus números: " + numerosApostados);
        System.out.println("Acertos: " + acertos.size());
    }
}
```

Chat Terminal

Enunciado: Desenvolva um chat simples que:

- Permita dois usuários conversarem
- Armazene mensagens em array
- Exiba histórico da conversa
- Inclua timestamps nas mensagens

Implementação:

```
public class SistemaChat {
    public static void main(String[] args) {
        String[] mensagens = new String[10];
        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("HH:mm:ss");

        System.out.print("Digite o nome do primeiro usuário: ");
        String usuario1 = scanner.nextLine();

        System.out.print("Digite o nome do segundo usuário: ");
        String usuario2 = scanner.nextLine();

        for (int i = 0; i < 10; i++) {
            String usuarioAtual = (i % 2 == 0) ? usuario1 : usuario2;
            String horaAtual = LocalTime.now().format(formatter);

            System.out.printf("[%s] %s: ", horaAtual, usuarioAtual);
            String mensagem = scanner.nextLine();

            if (mensagem.equalsIgnoreCase("sair")) {
                break;
            }

            mensagens[i] = String.format("[%s] %s: %s", horaAtual,
usuarioAtual, mensagem);
```

```
        }

        System.out.println("\nHistórico da conversa:");
        for (String mensagem : mensagens) {
            if (mensagem != null) {
                System.out.println(mensagem);
            }
        }
    }
}
```