PROVA H1 -

Questão teórica 1

* **Classe Anêmica:** Classes que contêm principalmente dados (atributos) e possuem pouca lógica ou métodos associados a elas. É uma abordagem mais passiva em termos de comportamento dentro das classes.
* **Classe Rica:** Classes que não só contêm dados (atributos) mas também métodos (comportamento) associados a esses dados. É uma abordagem mais ativa, onde as classes possuem a capacidade de manipular seus próprios dados e executar lógica de negócios associada a eles.

Em termos de boas práticas de programação, a escolha entre uma abordagem "anêmica" ou "rica" de design de classes depende do contexto e das necessidades específicas do projeto ou aplicação em questão.  
  
Questão 2 teórica  
  
  
O princípio SOLID de Responsabilidade Única é um dos cinco princípios do design de software orientado a objetos. Ele estabelece que uma classe deve ter apenas uma razão para mudar, ou seja, deve ter uma única responsabilidade.  
  
using System;

// Classe Employee para armazenar dados do funcionário

public class Employee

{

public string Name { get; set; }

public double Salary { get; set; }

public DateTime HireDate { get; set; }

}

// Classe separada para cálculos relacionados a salário

public class SalaryCalculator

{

public double CalculateSalary(Employee employee)

{

// Lógica para calcular o salário (exemplo simplificado)

double bonus = 0.1 \* employee.Salary;

return employee.Salary + bonus;

}

}

// Classe separada para geração de relatórios

public class EmployeeReportGenerator

{

public void GenerateEmployeeReport(Employee employee)

{

// Lógica para gerar o relatório do funcionário

Console.WriteLine($"Employee Name: {employee.Name}");

Console.WriteLine($"Salary: {employee.Salary:C}");

Console.WriteLine($"Hire Date: {employee.HireDate:d}");

// ... mais informações no relatório

}

}

Questão 3 teórica  
  
  
  
A validação de dados é essencial no desenvolvimento de software por três razões principais:

1. **Integridade dos Dados:** Garante que os dados inseridos estejam corretos e dentro dos parâmetros esperados, prevenindo inconsistências que possam corromper as informações.
2. **Segurança do Sistema:** Evita vulnerabilidades como injeção de SQL e ataques de script, protegendo contra potenciais brechas de segurança.
3. **Experiência do Usuário:** Contribui para uma melhor experiência do usuário, oferecendo feedback imediato sobre dados inválidos e facilitando a correção de erros, melhorando assim a usabilidade do sistema.

Parte superior do formulário