|  |  |  |
| --- | --- | --- |
|  | **Escola Estadual Professor João Anastácio**  **CURSO TÉCNICO EM INFORMÁTICA**  **AVALIAÇÃO DO 3º BIMESTRE DA DISCIPLINA**  **LABORATÓRIO DE SOFTWARE**  **ALUNO (a):**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **PROFESSOR(a):**  **DATA:     /     /**  **VALOR: 05 pontos     TURMA: \_\_\_\_\_\_                     Nota:** \_\_\_\_\_\_ |  |

"Educação não transforma o mundo. Educação muda as pessoas. Pessoas transformam o mundo."

(Paulo Freire)

1. Qual das alternativas representa corretamente a diferença entre design e arquitetura de software?

a) O design trata de decisões de longo prazo, enquanto a arquitetura foca apenas em detalhes locais.

b) O design analisa conexões locais entre componentes, já a arquitetura avalia impactos globais no sistema.

c) A arquitetura está restrita à programação de classes, enquanto o design abrange apenas a documentação.

d) O design é estático, enquanto a arquitetura nunca muda ao longo do tempo.

1. Por que a programação estruturada, proposta por Dijkstra, foi considerada uma revolução?

a) Porque introduziu o uso irrestrito do goto, tornando o fluxo mais flexível.

b) Porque substituiu os saltos arbitrários por estruturas de controle bem definidas, como sequência, seleção e repetição.

c) Porque criou novas linguagens, como COBOL e Fortran.

d) Porque permitiu que funções continuassem existindo mesmo após sua execução.

1. Qual conceito é fundamental na Programação Orientada a Objetos, segundo o material?

a) Imutabilidade total das variáveis.

b) Uso disciplinado de goto para controlar fluxos indiretos.

c) Persistência de variáveis locais após a execução de funções, levando à criação de objetos.

d) Exclusão do conceito de polimorfismo.

1. Na Programação Funcional, o princípio da imutabilidade significa que:

a) Variáveis podem ser alteradas livremente, desde que dentro da função.  
b) Uma vez inicializada, a variável não pode ter seu valor modificado.  
c) O estado global do programa deve ser constantemente atualizado.  
d) O map e o filter substituem a necessidade de funções puras.

1. O Princípio da Inversão de Dependência (Dependency Inversion Principle – DIP) afirma que:

a) Classes concretas devem sempre depender umas das outras.

b) Módulos de alto nível devem depender de módulos de baixo nível.

c) Tanto módulos de alto quanto de baixo nível devem depender de abstrações.

d) Interfaces nunca devem ser usadas como contratos.