UNIP
EDUCAÇÃO, POR SUA ESCOLHA

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 2

CQA - COMISSÃO DE QUALIFICAÇÃO E AVALIAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 2

Christiane Mazur Doi

Doutora em Engenharia Metalúrgica e de Materiais, Mestra em Ciências - Tecnologia Nuclear, Especialista em Cálculo e Matemática Aplicada, Especialista em Língua Portuguesa e Literatura, Especialista em Recursos Gramaticais para Revisão Textual, Engenheira Química e Licenciada em Matemática, com Aperfeiçoamento em Estatística e MBA em Engenharia Econômica. Professora titular da Universidade Paulista.

Larissa Rodrigues Damiani

Doutora e Mestra em Ciências pelo programa de Engenharia Elétrica – Microeletrônica e Engenheira Elétrica – modalidade Eletrônica (ênfase em Telemática). Professora titular da Universidade Paulista.

Tiago Guglielmeti Correale

Doutor e Mestre em Engenharia Elétrica e Engenheiro Elétrico - ênfase em Telecomunicações. Professor titular da Universidade Paulista.

Questão 1

Questão 1.[1]

Durante as eleições, o eleitor deve comparecer à sua seção e zona munido de um documento válido. Ao chegar ao local, apresenta o documento ao mesário, que verifica se o eleitor está apto a votar. Caso afirmativo, o mesário informa ao sistema o número do título de eleitor. O sistema valida o título e habilita o voto eletrônico para o eleitor. O eleitor informa os números de seus candidatos, podendo anular ou confirmar seu voto. Ao final do dia, termina o processo eleitoral da seção, o mesário finaliza o sistema, que gera os dados em tela ou em papel do resultado da urna, listando os votos para cada candidato. A totalização das urnas ocorre em um processo distinto em que o resultado final da eleição é apresentado à população.

Partindo dessa descrição, assinale a opção correta que corresponde à modelagem conceitual, utilizando diagrama de caso de uso com UML.

A. Verificar o Documento do eleitor e Habilitar o Voto Eletrônico são casos de uso.
B. No processo eleitoral da seção, os atores são: Eleitor, Mesário e População.
C. O caso de uso Informar Título tem uma associação do tipo <<extends>> com o caso de uso Validar Título.
D. O caso de uso Informar Número Candidato tem uma associação do tipo <<extends>> com os casos de uso Anular Voto e Confirmar Voto.
E. Gerar Dados em Tela e Gerar Dados em Papel têm uma associação do tipo <<implements>> com o caso de uso Gerar Dados.

1. Introdução teórica

Caso de uso

No contexto de modelagem de sistemas, um **caso de uso** é uma sequência de ações realizada pelo sistema que gera um resultado observável de valor para determinado ator. Por sua vez, um **ator** é um papel que solicita ações ou eventos do sistema e recebe resultados. Cada ator pode participar de vários casos de uso. Cada caso de uso é um objetivo de um ator, e não corresponde às tarefas do sistema.

Logo, um caso de uso pode ser definido como a descrição de uma sequência de eventos de um ator que usa o sistema para completar um processo. É uma técnica utilizada

[1]Questão 25 – Enade 2008.

para descrever o que um sistema deve fazer. O caso de uso mapeia o problema, não a solução. Trata-se, portanto, da visão do usuário a respeito do sistema.

Os casos de uso têm por objetivos:

- decidir e descrever os requisitos funcionais do sistema;
- fornecer uma descrição clara e consistente do que o sistema deve fazer.

Um exemplo de atores e de casos de usos está mostrado na figura 1. Na parte superior, temos o caso de uso *Cadastrar Usuário*. O ator *Administrador* pode cadastrar novos usuários no sistema, que podem ser *Gerentes* ou *Vendedores*. Na parte inferior, vemos o caso de uso *Cadastrar Cliente*. O ator *Funcionário* pode cadastrar clientes no sistema.

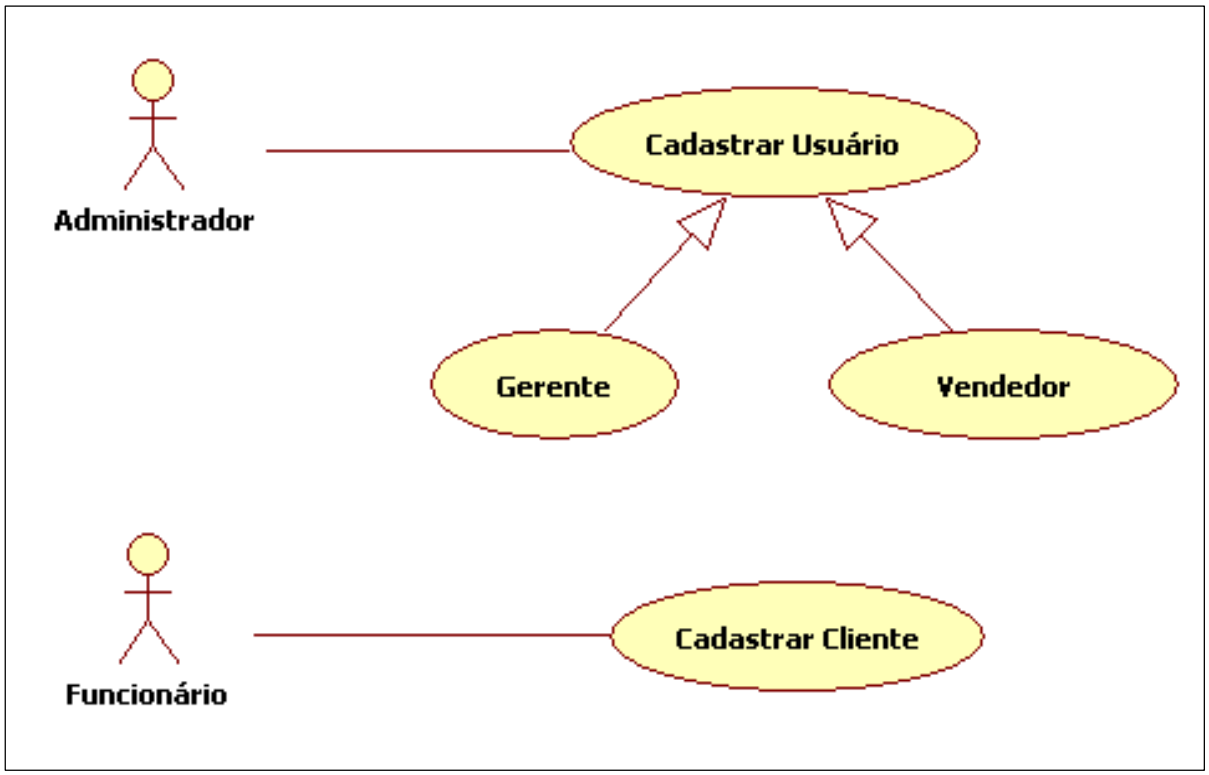


Figura 1. Exemplo de atores e de casos de usos.

Um relacionamento <<extends>> entre os Casos de Uso *A* e *B* significa que o caso de uso *B* pode ser acrescentado para complementar a funcionalidade do caso de uso *A*, mas essa adição não é obrigatória. Quando o caso de uso *A* for invocado, ele verificará se suas extensões devem ou não ser invocadas.

Na figura 2, temos um exemplo de caso de uso *Editar Documento*, que pode utilizar-se ou não das funcionalidades *Substituir Texto* ou *Corrigir Ortografia*, as quais são casos de uso independentes do primeiro.

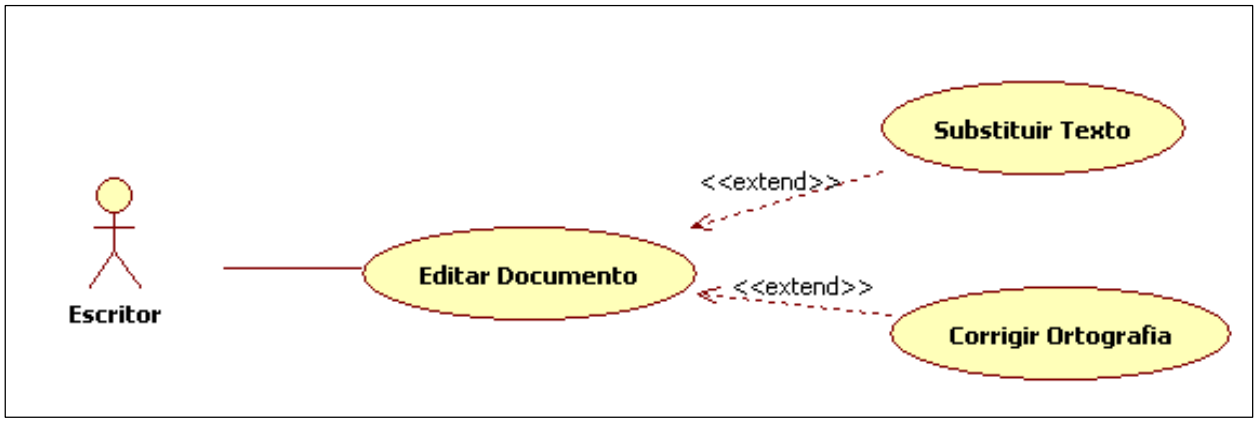


Figura 2. Exemplo de caso de uso *Editar Documento*.

No contexto de programação orientada a objetos, uma **interface** é um "contrato" que especifica quais métodos uma classe deve implementar. A interface não pode ser diretamente instanciada, apenas pode ser implementada por uma classe.

A palavra-chave <<implements>> é utilizada para indicar que uma classe implementa determinada interface, conforme o exemplo dado na figura 3, que traz um diagrama de classes UML. A interface *Pessoa* define um contrato comum para todos os tipos de pessoa no sistema. Assim, qualquer classe que implemente essa interface será obrigada a fornecer comportamentos definidos, garantindo padronização e polimorfismo.

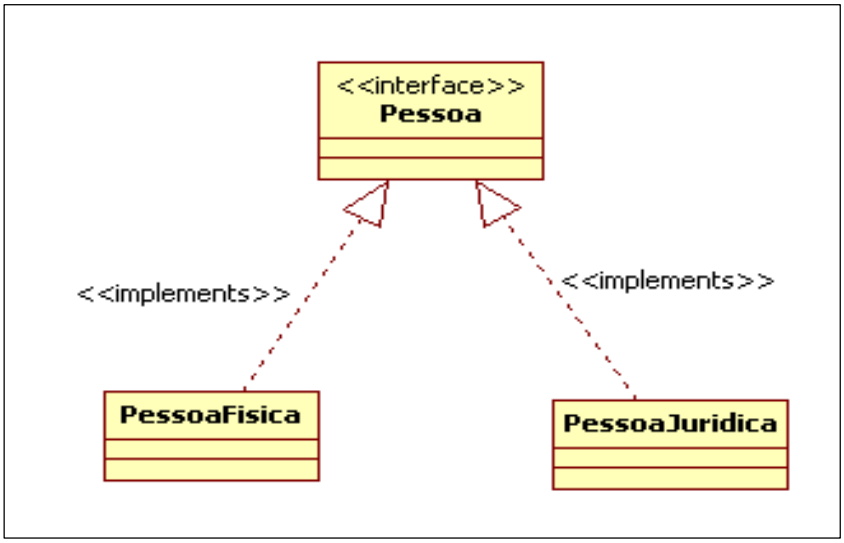


Figura 3. Palavra-chave *implements*.

2. Análise das alternativas

A – Alternativa incorreta.

JUSTIFICATIVA. *Habilitar o voto eletrônico* é um caso de uso porque é uma ação desenvolvida pelo sistema para um retorno de valor para o Ator. No entanto, *Verificar o documento do eleitor* não é um caso de uso porque nada acontece no sistema com relação a essa ação.

B – Alternativa incorreta.
JUSTIFICATIVA. O mesário informa ao sistema o número do título de eleitor, portanto é um ator. O eleitor informa os números de seus candidatos e confirma ou anula o voto no sistema, portanto também é um ator. Porém a população é apenas uma generalização que, no sistema apresentado, não tem significado, não sendo, portanto, ator.

C – Alternativa incorreta.
JUSTIFICATIVA. *Informar Título* e *Validar Título* são passos do caso de uso *Habilitar o Voto Eletrônico*, e não casos de uso separados.

D – Alternativa correta.
JUSTIFICATIVA. O eleitor, ao informar o número do candidato, pode anular ou confirmar seu voto. Como são ações opcionais, a associação <<extends>> é a adequada.

E – Alternativa incorreta.
JUSTIFICATIVA. A associação do tipo <<implements>> refere-se a classes. Os casos de uso não são implementados, são realizados. Uma realização de casos de uso (não implementação) descreve como determinado caso de uso é realizado no modelo de projeto em termos de objetos de colaboração.

3. Indicações bibliográficas

- FUGERI, S. *Programação orientada a objetos: conceitos e técnicas*. São Paulo: Érica, 2014.
- GILLEANES, T. A. G. *UML 2: uma abordagem prática*. São Paulo: Novatec, 2018.
- RANGEL, P. CARVALHO JUNIOR, J. G. de. *Sistemas orientados a objetos: teoria e prática com UML e Java*. Rio de Janeiro: Brasport, 2021.

Questão 2

Questão 2.[2]

Os métodos de projeto e análise de algoritmos são necessários para o desenvolvimento de algoritmos eficientes, pois eles permitem que se resolvam problemas computacionais, reduzindo complexidade e tempo de execução. Acerca desses métodos, assinale a opção incorreta.

A. A abordagem de Divisão e Conquista propõe dividir o problema em vários subproblemas, resolvendo-os e combinando suas soluções para criar a solução final do problema original.
B. A Programação Dinâmica é uma técnica tipicamente aplicada a problemas de otimização em que pode haver várias soluções possíveis.
C. O método Guloso nem sempre encontra a solução ótima, mas faz sempre a melhor escolha momentânea.
D. A Programação Dinâmica e o Método Guloso têm em comum o fato de que se aplicam a problemas em que se observa a existência de sobreposição de subproblemas, ou seja, subproblemas que se repetem.
E. Os métodos de Divisão e Conquista e Programação Dinâmica apresentam a mesma eficiência quando resolvem problemas combinando soluções de subproblemas dependentes uns dos outros.

1. Introdução teórica

Métodos de projeto

Métodos de projeto são estratégias gerais usadas para construir algoritmos eficientes para resolver problemas. São, portanto, abordagens sistemáticas para pensar em como realizar uma tarefa computacional por meio de um algoritmo.

O método de **divisão e conquista** divide o problema em duas ou mais partes, criando subproblemas menores. Os subproblemas são resolvidos recursivamente, usando novas divisões e conquistas. Caso os subproblemas sejam suficientemente pequenos, serão resolvidos de forma direta.

A **programação dinâmica** é uma técnica computacional apoiada no princípio de otimização. As soluções são obtidas frequentemente por um processo regressivo, trabalhando

[2]Questão 26 – Enade 2008.

no problema do final para o começo. A dificuldade é reduzida ao decompor o problema numa sequência de problemas inter-relacionados mais simples. Essa técnica baseia-se no princípio de que o problema completo pode ser resolvido se os valores das melhores soluções de certos subproblemas forem determinados.

O método **guloso** baseia-se em decisões tomadas de forma isolada em cada passo. A estratégia é pegar a solução ótima local. Ao término do algoritmo, espera-se que a solução ótima tenha sido encontrada. É a técnica mais simples que pode ser aplicada a uma grande variedade de problemas. O problema possui n entradas e é requerido que satisfaça uma restrição chamada de solução viável. Uma solução viável que satisfaça a função objetivo é chamada de solução ótima.

É fácil determinar uma solução viável, mas ela não é necessariamente uma solução ótima. O método guloso sugere que podemos construir um algoritmo que trabalhe em estágios, considerando uma entrada por vez. Em cada estágio, uma decisão é tomada considerando se uma entrada particular é uma solução ótima. Isso é conseguido considerando as entradas em uma ordem determinada por algum processo de seleção. Se a inclusão da próxima entrada, na solução ótima construída parcialmente, resultará em uma solução inviável, então essa entrada não será adicionada à solução parcial. O processo de seleção em si é baseado em alguma medida de otimização, que pode ou não ser a função objetivo. Na verdade, várias medidas de otimização diferentes podem ser aplicáveis para determinado problema. Muitas delas, entretanto, resultarão em algoritmos que gerarão solução subótimas.

2. Análise das alternativas

A – Alternativa incorreta.
JUSTIFICATIVA. Trata-se da definição clássica do método *Divisão e Conquista.*

B – Alternativa incorreta.
JUSTIFICATIVA. A *Programação Dinâmica* aplica-se, principalmente, a problemas de otimização combinatória.

C – Alternativa correta.
JUSTIFICATIVA. O método *Guloso* nem sempre faz a melhor opção momentânea. Ele escolhe um subconjunto do problema que satisfaça determinada restrição. Essa função viável é chamada de *Solução Ótima*, mas o algoritmo nem sempre tende para uma solução ótima.

Somente alguns problemas, que possuem a propriedade de escolha gulosa, têm uma solução global ótima, alcançada a partir de um conjunto de soluções locais ótimas.

D – Alternativa incorreta.
JUSTIFICATIVA. Os dois métodos trabalham em subproblemas que se repetem.

E – Alternativa correta.
JUSTIFICATIVA. Embora o método *Divisão e Conquista* resolva, recursivamente, problemas combinando subproblemas independentes, a *Programação Dinâmica* trata de subproblemas que não são necessariamente diferentes. Portanto, eles não apresentam a mesma eficiência.

Como há mais de uma alternativa correta e mais de uma alternativa incorreta, a questão foi **anulada**.

3. Indicações bibliográficas

- BHARGAVA, A. Y. *Entendendo algoritmos*. São Paulo: Novatec, 2017.
- CORMEN, T. H. *Algoritmos – teoria e prática.* São Paulo: LTC, 2012.
- DASGUPTA, S.; PAPADIMITRIOU, C. H.; VAZIRANI, U. *Algoritmos.* São Paulo: McGraw-Hill, 2006.
- MEDINA, M.; FERTIG, C. *Algoritmos e programação – teoria e prática.* São Paulo: Novatec, 2005.

Questão 3

Questão 3.[3]

Os termos da sequência de Fibonacci são definidos por:

```
Fibonacci(0) = 0
Fibonacci(1) = 1
Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)
```

Uma solução recursiva para o cálculo do *i*-ésimo termo da sequência é dada pela função a seguir.

```
01. funcao fibonacci(inteiro longo n)
02.     se((n=0) OU (n=1)) entao
03.         retorne n
04.     senao
05.         retorne fibonacci(n-1) + fibonacci(n-2)
06.     fim se
07. fim
```

Acerca da execução recursiva dessa função, assinale a opção incorreta.

A. À medida que o valor de `n` cresce, há um aumento no número de chamadas recursivas.
B. Na linha 4, a ordem de execução é calcular o valor para `fibonacci(n-1)` e somente depois calcular o valor para `fibonacci(n-2)`.
C. O método recursivo é o mais eficiente para o cálculo do i-ésimo termo da sequência de Fibonacci, pois realiza duas chamadas por passo da recursão, cada uma mais simples do que a chamada original.
D. As condições de parada da recursão são: o valor de `n` é 0 ou o valor de `n` é 1.
E. O uso da recursão para o problema da série de Fibonacci não é indicado, pois ele gera rapidamente uma explosão de chamadas do método.

[3]Questão 27 – Enade 2008.

1. Introdução teórica

Solução recursiva de problemas

Uma solução recursiva de problemas é uma estratégia que utiliza a técnica de dividir para conquistar. A ideia fundamental é dividir um problema em um conjunto de subproblemas menores, resolvidos independentemente, para depois serem combinados, a fim de gerar a solução final.

Uma definição recursiva é constituída geralmente por uma parte base e por uma parte recursiva. Muitas linguagens de programação permitem que funções chamem a si mesmas. As principais vantagens do uso da recursão são a clareza na interpretação do código e a simplicidade e a elegância na implementação. Suas principais desvantagens são a dificuldade para depurar erros e a ineficiência.

A chamada de uma função requer espaço para parâmetros, variáveis locais e endereço de retorno. Com as chamadas recursivas, todas essas informações são armazenadas em uma pilha de execução e, depois, são retiradas. A quantidade de informação armazenada pode facilmente se tornar desproporcional ao número de chamadas.

O processo de alocar e liberar memória e de copiar informações envolve tempo computacional. Quanto mais chamadas, mais tempo gasto.

Por fim, é fundamental definir corretamente os critérios de parada, pois, se forem definidos de maneira equivocada, podem provocar uma infinidade de chamadas e consumir toda a memória disponível.

2. Análise das alternativas

A – Alternativa incorreta.
JUSTIFICATIVA. À medida que o valor de `n` cresce, há efetivamente aumento no número de chamadas recursivas (`n-1` e `n-2`).

B – Alternativa incorreta.
JUSTIFICATIVA. A execução de `fibonacci(n-1)` também é recursiva e, somente após concluída, chama-se `fibonacci(n-2)`.

C – Alternativa correta.

JUSTIFICATIVA. O método recursivo torna-se mais ineficiente com o aumento de `n`. Dependendo do valor de `n`, pode causar um estouro de pilha (StackOverflow), devido ao número excessivo de chamadas recursivas.

D – Alternativa incorreta.

JUSTIFICATIVA. A condição de parada é sempre `n=0` ou `n=1`.

E – Alternativa incorreta.

JUSTIFICATIVA. Se `n` for muito grande, haverá uma explosão de chamadas de métodos e um erro do tipo StackOverflow. Esse erro ocorre quando a pilha de execução de um programa fica sem espaço, geralmente por recursão infinita ou por uso exagerado de memória local.

Um exemplo do cálculo de sequência de Fibonacci para qualquer `n` dado, sem recursão, está mostrado abaixo, implementado em Java. Esse programa é muito mais eficiente (embora menos elegante) do que o proposto na questão. Para testes em casa, programe as duas soluções, execute para n=30.000 e verifique o tempo gasto por cada um dos programas.

```
01. public long fibonacci(long n) {
02.     long n1, n2, total;
03.
04.     total = 1;
05.     n1 = 0;
06.     n2 = 1;
07.
08.     for (int i = 2; i <= n; i++) {
09.         total = (n1 + n2);
10.         n1 = n2;
11.         n2 = total;
12.     }
13.
14.     return total;
15. }
```

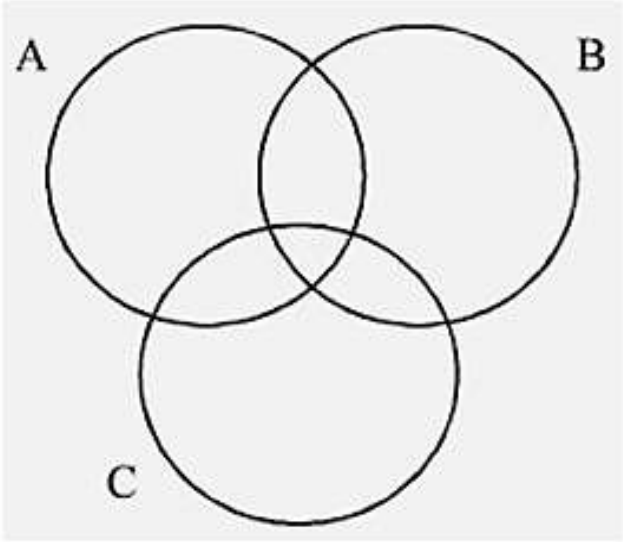
3. Indicações bibliográficas

- BHARGAVA, A. Y. *Entendendo algoritmos*. São Paulo: Novatec, 2017.
- CORMEN, T. H. *Algoritmos – teoria e prática.* São Paulo: LTC, 2012.
- DASGUPTA, S.; PAPADIMITRIOU, C. H.; VAZIRANI, U. *Algoritmos.* São Paulo: McGraw-Hill, 2006.
- MEDINA, M.; FERTIG, C. *Algoritmos e programação – teoria e prática.* São Paulo: Novatec, 2005.

Questão 4

Questão 4.[4]

Leia o diagrama a seguir.



A figura acima mostra 3 conjuntos — A, B e C — em que cada conjunto é representado, no diagrama de Venn, por um círculo no plano. Com relação aos conjuntos A, B e C, julgue os seguintes itens.

I. $A \cup B = \emptyset$

II. $A - (B \cap C) = (A - B) \cap (A - C)$

III. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

IV. $A \cap A = \emptyset$

Assinale a opção correta.

A. Apenas um item está certo.

B. Apenas os itens I e II estão certos.

C. Apenas os itens II e III estão certos.

D. Apenas os itens III e IV estão certos.

E. Apenas os itens II, III e IV estão certos.

[4]Questão 28 – Enade 2008.

1. Introdução teórica

Operações entre conjuntos

Na matemática, conjuntos são coleções definidas de elementos. Os conjuntos são geralmente denotados por letras maiúsculas. Como exemplos, temos os conjuntos V e P, cujas listas de elementos é exposta a seguir.

- $V = \{a, e, i, o, u\}$ (conjunto das vogais)
- $P = \{2,3,5,7,11\}$ (conjunto dos números primos menores do que 13)

Podemos realizar operações entre conjuntos. Veremos, a seguir, três operações: união, interseção e diferença.

Dados os conjuntos A e B, a sua **união** ($A \cup B$) é formada por todos os elementos pertencentes a A e a B, conforme mostrado na figura 1.

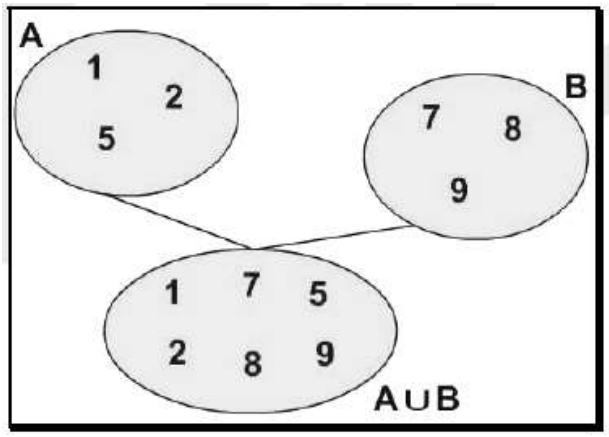


Figura 1. União de conjuntos.

Dados os conjuntos A e B, a sua **interseção** ($A \cap A$) é formada por todos os elementos pertencentes a A e a B simultaneamente, conforme mostrado na figura 2.

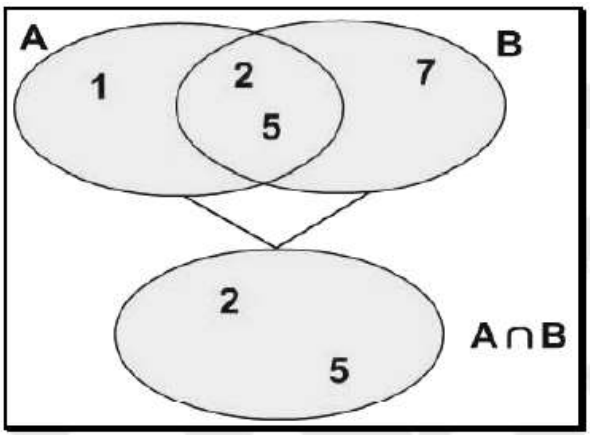


Figura 2. Interseção de conjuntos.

Dados os conjuntos A e B, a sua **diferença** ($A - B$) é formada pelos elementos que pertencem a A e não pertencem a B, conforme mostrado na figura 3.

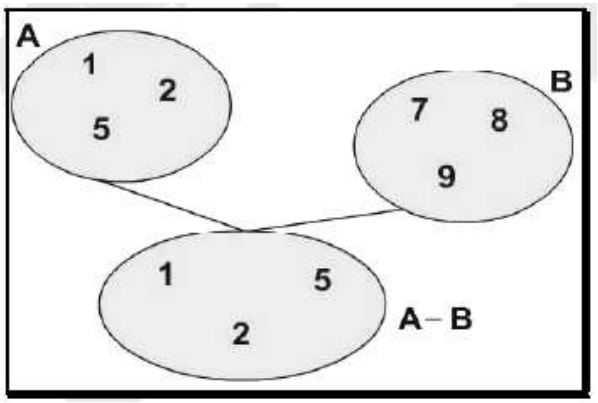


Figura 3. Diferença de conjuntos.

Veja os exemplos a seguir.

1. Se $A = \{1,2,5\}$ e $B = \{7,8,9\}$, então:

   $A \cup B = \{1,2,5,7,8,9\}$

   $A \cap B = \emptyset$ (conjunto vazio)

   $A - B = \{1,2,5\}$

2. Se $A = \{1,2,5\}$ e $B = \{2,5,7,8\}$, então:

   $A \cup B = \{1,2,5,7,8\}$

   $A \cap B = \{2,5\}$

   $A - B = \{1\}$

O **conjunto vazio** ($\emptyset$) é o conjunto sem elementos. Todo conjunto tem o conjunto vazio como subconjunto. A união de um conjunto A e de um conjunto vazio resulta sempre em A. A interseção do conjunto vazio com qualquer conjunto é o conjunto vazio.

2. Análise das afirmativas

I - Afirmativa incorreta.

JUSTIFICATIVA. Para facilitar a visualização e o entendimento, colocamos números em cada um dos envolvidos nas interações entre os conjuntos, conforme mostrado na figura 4.

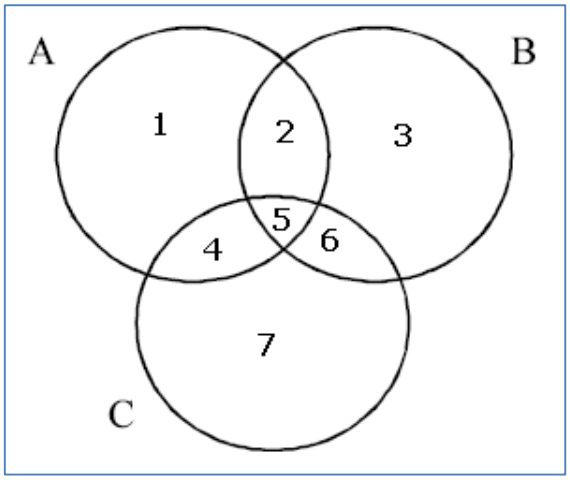


Figura 4. União dos conjuntos.

A união de A com B somente daria um conjunto vazio se tanto A quanto B fossem conjuntos vazios. No exemplo, temos que $A \cup B = \{1,2,3,4,5,6\}$.

II – Afirmativa incorreta.

JUSTIFICATIVA. Vamos resolver o item "passo a passo", considerando o diagrama da figura 4.

- $B \cap C = \{5,6\}$
- $A - (B \cap C) = \{1,2,4,5\} - \{5,6\} = \{1,2,4\}$
- $A - B = \{1,2,4,5\} - \{2,3,5,6\} = \{1,4\}$
- $A - C = \{1,2,4,5\} - \{4,5,6,7\} = \{1,2\}$
- $(A - B) \cap (A - C) = \{1,4\} \cap \{1,2\} = \{1\}$

Como $\{1,2,4\}$ é diferente de $\{1\}$, a afirmativa é incorreta.

III – Afirmativa correta.

JUSTIFICATIVA. Vamos resolver o item "passo a passo", considerando o diagrama da figura 4.

- $B \cup C = \{2,3,4,5,6,7\}$
- $A \cap (B \cup C) = \{1,2,4,5\} \cap \{2,3,4,5,6,7\} = \{2,4,5\}$
- $A \cap B = \{2,5\}$
- $A \cap C = \{4,5\}$
- $(A \cap B) \cup (A \cap C) = \{2,4,5\}$

Podemos perceber que as operações tiveram o mesmo resultado, {2, 4, 5}. A igualdade proposta na afirmativa, $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$, é conhecida na matemática e na lógica como Lei Distributiva da Interseção sobre a União.

IV – Afirmativa incorreta.

JUSTIFICATIVA. A interseção de A com A somente resultaria em um conjunto vazio se A fosse um conjunto vazio. A regra é: $A \cap A = A$ e $A \cap \emptyset = \emptyset$.

Alternativa correta: A.

3. Indicações bibliográficas

- GERSTING, J. L. *Fundamentos matemáticos para a Ciência da Computação*. São Paulo: LTC, 2016.
- HAZZAN, S. *Fundamentos de Matemática Elementar – conjuntos e funções*. São Paulo: Atual, 2019.
- KNUTH, D. E. *Matemática Concreta: fundamentos para a Ciência da Computação.* 2. ed. São Paulo: LTC, 1995.

Questão 5

Questão 5.[5]

Considere a sentença a seguir.

Se Maria for ao aniversário, João irá e ficará feliz, mas Maria ficará infeliz, ou, se João não for ao aniversário, Maria irá e ficará feliz, mas João ficará infeliz.

Considere as proposições a seguir.

- P: João vai ao aniversário.
- Q: Maria vai ao aniversário.
- R: João feliz.
- S: Maria feliz.

Assinale a opção que contém fórmula de lógica proposicional com uma representação válida para a sentença proposta. Quanto à notação dos operadores, considere o seguinte: conjunção = Λ; disjunção = V; negação = ¬; implicação = →.

A. $\Big(\big(Q \to (P \wedge R)\big) \wedge \neg S\Big) \vee \Big(\big(\neg P \to (Q \wedge S)\big) \wedge R\Big)$

B. $\Big(\big(\neg Q \to (P \wedge R)\big) \wedge S\Big) \wedge \Big(\big(P \to (Q \wedge S)\big) \wedge \neg R\Big)$

C. $\Big(Q \to \big((P \wedge R) \wedge \neg S\big)\Big) \vee \Big(\neg P \to \big((Q \wedge S) \wedge \neg R\big)\Big)$

D. $\Big(\big(Q \to (P \wedge R)\big) \wedge S\Big) \to \Big(\big(\neg P \to (Q \wedge S)\big) \wedge R\Big)$

E. $\Big(\big(Q \to (P \wedge R)\big) \wedge S\Big) \to \Big(\big(P \to (Q \wedge S)\big) \vee R\Big)$

1. Introdução teórica

Operações lógicas

A lógica estuda a relação entre proposições, que são sentenças declarativas que podem ser classificadas como verdadeiras ou falsas. As operações lógicas básicas são:

- conjunção;
- disjunção;
- negação;
- implicação.

[5]Questão 29 – Enade 2008 (com adaptações).

A **conjunção**, algebricamente representada pelo símbolo Λ, une duas proposições pelo conectivo "E". A conjunção exige que ambas as proposições envolvidas sejam verdadeiras para que o resultado da operação também seja verdadeiro. Se pelo menos uma delas for falsa, o resultado da conjunção será falso. Por exemplo, para a frase "está sol e estou na praia" ser verdadeira, é necessário que realmente esteja fazendo sol e que eu realmente esteja na praia. Em situações em que as sentenças unidas pela conjunção apresentam algum tipo de contraste, é comum usarmos a conjunção adversativa "MAS" como conectivo.

A **disjunção**, algebricamente representada pelo símbolo V, une duas proposições pelo conectivo "OU". A disjunção exige que pelo menos uma das proposições envolvidas seja verdadeira para que o resultado da operação também seja verdadeiro. O resultado só será falso se ambas as proposições componentes forem falsas. Por exemplo, a afirmação "eu vou de ônibus ou de metrô" é verdadeira se eu usar qualquer um dos dois meios de transporte ou ambos.

A **negação** é uma operação diferente das anteriores, pois atua sobre uma única proposição. Algebricamente, ela é representada pelo símbolo ¬, lido como "NÃO". A função da negação é inverter o valor lógico da proposição à qual se aplica. Se uma afirmação é verdadeira, a sua negação será falsa. Por outro lado, se a afirmação original é falsa, a sua negação é verdadeira. Por exemplo, se a proposição "o céu é azul" for considerada verdadeira, então a sua negação, "o céu não é azul", será falsa.

Por fim, temos a **implicação**, representada pelo símbolo →, que une duas proposições pelo conectivo "SE...ENTÃO". A implicação estabelece uma relação de condição entre duas proposições. Ela só produz um resultado falso em uma situação muito específica: quando a primeira proposição (chamada de antecedente) é verdadeira, e a segunda proposição (chamada de consequente) é falsa. Em todas as outras combinações de valores lógicos, a implicação é considerada verdadeira. Um exemplo clássico é a sentença: "se chover, então eu fecharei a janela". A única circunstância em que essa sentença é considerada falsa é se realmente estiver chovendo (antecedente verdadeiro) e eu não fechar a janela (consequente falso).

Além dos operadores, também podemos representar as próprias proposições simbolicamente. É comum que uma proposição seja representada por uma letra. Por exemplo, a disjunção "eu vou de ônibus ou de metrô" pode ser representada como $A \vee B$. Nesse caso, chamamos de A a proposição "eu vou de ônibus" e de B a proposição "eu vou de metrô".

2. Resolução da questão

A questão pede que transformemos a sentença a seguir em uma fórmula proposicional. Isso significa que devemos transformar essa sentença, expressa em linguagem corrente, em uma expressão simbólica.

Se Maria for ao aniversário, João irá e ficará feliz, mas Maria ficará infeliz, ou, se João não for ao aniversário, Maria irá e ficará feliz, mas João ficará infeliz.

Os símbolos que devemos usar para cada proposição simples componente da sentença composta são os que seguem.

- P: João vai ao aniversário.
- Q: Maria vai ao aniversário.
- R: João feliz.
- S: Maria feliz.

Podemos "dividir" essa longa sentença em duas partes. Começaremos pela primeira parte:

Se Maria for ao aniversário, João irá e ficará feliz, mas Maria ficará infeliz.

Nesse trecho, podemos identificar diretamente uma implicação e duas conjunções, cujos conectivos são destacados a seguir.

***Se** Maria for ao aniversário, João irá **e** ficará feliz, **mas** Maria ficará infeliz.*

Pelo contexto e pela própria ordem de precedência dos operadores lógicos, consideramos que a implicação é dominante em relação às operações de conjunção. Desse modo, temos:

***Se** Maria for ao aniversário, ((João irá **e** ficará feliz), **mas** Maria ficará infeliz).*

Além disso, podemos "reler" a sentença "Maria ficará infeliz" como "Maria não ficará feliz". Isso representa a negação da sentença S. Desse modo, a fórmula proposicional que representa essa primeira parte fica conforme exposto a seguir.

$$Q \rightarrow \big((P \wedge R) \wedge \neg S\big)$$

Na segunda parte, temos:

***Se** João não for ao aniversário, Maria irá **e** ficará feliz, **mas** João ficará infeliz.*

Nesse caso, seguindo o mesmo critério, temos a fórmula a seguir.

$$\neg P \rightarrow \big((Q \wedge S) \wedge \neg R\big)$$

Como ambas as partes estão unidas por uma disjunção, temos que a sentença completa é representada pela fórmula proposicional a seguir.

$$\Big(Q \rightarrow \big((P \wedge R) \wedge \neg S\big)\Big) \vee \Big(\neg P \rightarrow \big((Q \wedge S) \wedge \neg R\big)\Big)$$

Observação. O gabarito da questão, originalmente, considerava que o conectivo "MAS" expressa implicação entre sentenças. No entanto, isso vai de encontro ao consenso da literatura de lógica matemática, que afirma que esse conectivo expressa conjunção em situação de contraste entre sentenças. Por isso, as alternativas da questão foram modificadas.

Alternativa correta: C.

3. Indicações bibliográficas

- FAVARO, S.; FILHO, O. K. *Noções de lógica e matemática básica*. São Paulo: Ciência Moderna, 2020.
- GERSTING, J. L. *Fundamentos matemáticos para a Ciência da Computação*. São Paulo: LTC, 2016.
- QUILELLI, P. *Raciocínio lógico-matemático.* São Paulo: Saraiva, 2015.

Questão 6

Questão 6.[6]

O plano de negócios, mais do que um documento de elaboração das ações de implementação de um novo empreendimento, serve como documento que estabelece o relacionamento entre empreendedores e investidores. O conhecimento de características dos atores envolvidos nessa relação interfere diretamente na elaboração do plano de negócios. Considerando os papéis do empreendedor, do investidor e de conceitos de fatores envolvidos na elaboração do plano de negócios, assinale a opção correta.

A. O verdadeiro empreendedor cria um negócio diante de uma oportunidade e procura, o mais breve possível, vendê-lo para um grupo de investidores.
B. Investidores inteligentes consideram, ao analisar onde investir, que projeções financeiras mês a mês para um período maior que um ano constituem um dos fatores que garantem o sucesso de um novo empreendimento.
C. O empreendedor é uma pessoa à procura de riscos, que diante de uma nova oportunidade de empreendimento transfere todos os riscos para si.
D. As pessoas, as oportunidades, o contexto e as possibilidades de riscos e recompensas são quatro fatores fundamentais, que devem ser considerados para o sucesso de um novo empreendimento.
E. Um plano de negócios deve ser criado seguindo uma fórmula de sucesso preestabelecida apresentada em livros da área de administração e implementada em aplicativos.

1. Introdução teórica

Empreendedorismo e plano de negócios

Uma empresa é criada para sobreviver no longo prazo. Para isso, algumas ações são fundamentais, como as citadas a seguir.

- Inovar para crescer.
- Investir em qualificação.
- Fazer um planejamento estratégico.
- Manter a qualidade de produtos e serviços.
- Gerenciar o negócio.

[6]Questão 31 – Enade 2008.

- Controlar os custos.
- Manter a base de clientes.

O plano de negócios é o documento que estrutura as ideias e as opções que o empreendedor analisará para decidir sobre a viabilidade do seu negócio. Deve conter informações sobre o negócio, as previsões e as projeções financeiras, a análise do mercado, a previsão de fluxo de caixa e as necessidades de capital.

O empreendedor é aquele que inicia novos negócios e consegue escolher entre várias alternativas. Ele apresenta determinadas habilidades e competências para criar, abrir e gerir um negócio, gerando resultados positivos.

É impossível afastar o elemento risco do negócio, por isso ele tem que ser gerenciado e acompanhado, com o objetivo de minimizá-lo. Assumir riscos é a principal característica do empreendedor.

Não existem "fórmulas mágicas" para ser um empreendedor de sucesso. Há exemplos de negócios que prometiam sucesso e fracassaram. Igualmente, temos, também, negócios que nasceram por acaso e obtiveram sucesso.

2. Análise das alternativas

A – Alternativa incorreta.
JUSTIFICATIVA. A criação do negócio é voltada para o longo prazo e para a permanência da empresa.

B – Alternativa incorreta.
JUSTIFICATIVA. As projeções financeiras devem abranger vários períodos de análise (mensais, semestrais, anuais e acima de um ano). Quanto mais distante o prazo, menos precisas serão. Ao projetar um investimento, o investidor deve levar em conta o período necessário para que o capital seja remunerado, o que pode levar vários anos.

C – Alternativa incorreta.
JUSTIFICATIVA. Embora seja correto que o empreendedor assume riscos, ele não os procura (os riscos são inerentes ao negócio), muito menos transfere todos para si.

D – Alternativa correta.

JUSTIFICATIVA. Os fatores listados estão corretos. O negócio tem que ser avaliado em várias dimensões para que não sejam cometidos erros graves que o inviabilizem.

E – Alternativa incorreta.

JUSTIFICATIVA. Não existe uma fórmula de sucesso pré-estabelecida. Sucessos passados não garantem a viabilidade futura de um negócio similar. Cada um tem que ser analisado como um evento singular, mesmo que sejam comparados com negócios ou mercados similares.

3. Indicações bibliográficas

- CARVALHO, H. G. *Empreendedorismo*. São Paulo: Ferreira, 2009.
- DORNELAS, J. *Empreendedorismo corporativo: como ser um empreendedor, inovar e se diferenciar na sua empresa*. Barueri: Atlas, 2023.
- PETERS, M. *Empreendedorismo*. São Paulo: Artmed, 2009.

Questão 7

Questão 7.[7]

Com relação a conceitos de orientação a objetos, julgue os itens.

I. As variáveis ou métodos declarados com modificador de acesso *private* só são acessíveis a métodos da classe em que são declarados.

II. Uma classe deve possuir uma única declaração de método construtor.

III. Uma instância de uma classe abstrata herda atributos e métodos de sua superclasse direta.

IV. O polimorfismo permite substituir a lógica condicional múltipla (lógica *switch* ou faça caso).

São corretos apenas os itens

A. I e II.

B. I e III.

C. I e IV.

D. II e III.

E. II e IV.

1. Introdução teórica

1.1. Modificador de acesso `private`

No contexto da orientação a objetos, um membro (método ou variável) declarado com o modificador `private` somente pode ser acessado por uma instância da classe que declara o método ou variável. Veja o exemplo abaixo.

```
// Classe Cliente
class Cliente{
    private String nome = "Osmar";
}

// Classe de teste
public class Estudos{
    public static void main(String args[]){
        Cliente cliente = new Cliente();
        System.out.println(cliente.nome);

        System.exit(0);
    }
}
```

[7]Questão 32 – Enade 2008.

A tentativa de compilação do código anterior dará erro porque a classe Estudos não tem acesso aos membros privados da classe Cliente. Do mesmo modo, uma subclasse não tem acesso aos membros privados da superclasse, mas pode acessar membros com modificadores `protected` e `public`.

1.2. Método construtor

Um **construtor** é um método especial de uma classe que é chamado automaticamente quando um objeto é criado. Portanto, as instruções contidas no construtor serão executadas sempre que um objeto for instanciado. Em Java, um método construtor tem o mesmo nome da classe e não tem tipo de retorno. Veja o exemplo.

```
class TV {
    int tamanho;
    int canal;
    boolean ligada;

    TV(){
        tamanho=21;
        canal=0;
        ligada=false;
    }
}
```

No exemplo anterior, a classe TV é construída com tamanho de 21 polegadas, nenhum canal e desligada.

Além do construtor padrão, uma classe pode ter inúmeros construtores, o que constitui uma sobrecarga de construtores.

1.3. Polimorfismo

Polimorfismo é o princípio pelo qual duas ou mais classes que implementam a mesma interface podem definir métodos com a mesma assinatura, mas com comportamentos diferentes. A decisão sobre qual método será executado é tomada em tempo de execução, de acordo com o tipo real do objeto.

O polimorfismo pode ser aplicado com interfaces ou com classes abstratas. Usaremos a primeira abordagem. Uma **interface** é um "contrato" que define um conjunto de métodos que uma classe deve implementar, sem fornecer a implementação desses métodos. Diferentes classes podem implementar a mesma interface de maneiras distintas.

Veja o que segue.

- Interface que define o formato de operações aritméticas.

```
public interface OperacaoMatematica {
    double calcular(double x, double y);
}
```

- Classes que implementam operações aritméticas específicas.

```
public class Adicao implements OperacaoMatematica {
    public double calcular(double x, double y) {
        return x + y;
    }
}

public class Subtracao implements OperacaoMatematica {
    public double calcular(double x, double y) {
        return x - y;
    }
}
```

- Classe de teste.

```
public class Contas {
    public static void mostrarCalculo(OperacaoMatematica operacao, double x, double y) {
        System.out.println("O resultado é: " + operacao.calcular(x, y));
    }

    public static void main(String args[]) {
        // Primeiro calculamos uma adição
        Contas.mostrarCalculo(new Adicao(), 5, 5);       // Imprime: O resultado é: 10
        // Depois uma subtração
        Contas.mostrarCalculo(new Subtracao(), 5, 5);  // Imprime: O resultado é: 0
    }
}
```

Observe que o método `calcular` retorna resultados diferentes dependendo da operação. Esse é um exemplo de polimorfismo dinâmico (ou de sobrescrita), no qual o retorno é decidido em tempo de execução.

A sobrecarga de métodos, por sua vez, é um tipo de polimorfismo estático. Ele ocorre quando há vários métodos com o mesmo nome, mas assinaturas diferentes, dentro da mesma classe. Nesse caso, retorno é decidido em tempo de compilação.

A instrução **`switch`** pode ser usada quando precisamos escolher entre múltiplos caminhos no nosso programa, como na seleção de opção em um menu. Vejamos um exemplo.

```
public class Estudos{
    public static void main(String[] args){
        int valor = 4;
```

```
        switch(valor){
            case 1:
                System.out.println("Valor é 1");
                break;
            case 2:
                System.out.println("Valor é 2");
                break;
            case 3:
                System.out.println("Valor é 3");
                break;
            default:
                System.out.println("Valor diferente de 1, 2 e 3");
                break;
        }
    }
}
```

Comumente, podemos substituir comandos `switch` por polimorfismo, o que torna o código mais claro e extensível. O `switch` também poderia ser substituído, em alguns casos, por uma *hash table* (dicionário, mapa etc.).

O exemplo abaixo, para as classes `Adicao` e `Subtracao`, mostra a implementação que fizemos por meio da interface, mas usando o `switch`.

```
public static final int ADICAO = 1;
public static final int SUBTRACAO = 2;

public void mostrarCalculo (int operacao, double x, double y) {
    System.out.print("O resultado é: ");
    switch (operacao) {
        case ADICAO:
            System.out.print(""+(x+y));
            break;
        case SUBTRACAO:
            System.out.print(""+(x-y));
            break;
        //... outras operacoes
        default:
            throw new UnsupportedOperationException();
    }
}
```

1.4. Classes abstratas

O polimorfismo também pode ser aplicado com classes abstratas. Uma classe abstrata é um tipo especial de classe que não pode ser instanciada diretamente. Desse modo, não podemos criar objetos a partir dela.

Ela apenas serve como modelo ou base para outras classes que irão herdar dele e implementar seus métodos. Veja o exemplo.

- Classe abstrata

```
abstract class Forma {
    abstract double calcularArea();
}
```

- Subclasses concretas

```
class Retangulo extends Forma {
    double largura, altura;
    Retangulo(double largura, double altura) {
        this.largura = largura; this.altura = altura;
    }
    @Override
    double calcularArea() {
        return largura * altura;
    }
}

class Circulo extends Forma {
    double raio;
    Circulo(double raio) {
        this.raio = raio;
    }
    @Override
    double calcularArea() {
        return Math.PI * raio * raio;
    }
}
```

- Teste de polimorfismo

```
public class Teste {
    public static void main(String[] args) {
        Forma f1 = new Retangulo(5, 10);
        Forma f2 = new Circulo(3);

        System.out.println("Área do retângulo: " + f1.calcularArea());
        System.out.println("Área do círculo: " + f2.calcularArea());
    }
}
```

No exemplo anterior, `f1` e `f2` são do tipo `Forma` (classe abstrata), mas referenciam objetos de subclasses diferentes (`Retangulo` e `Circulo`, respectivamente). O método `calcularArea()` é chamado de forma polimórfica, e a implementação executada depende da subclasse real (`Retangulo` ou `Circulo`).

2. Análise das afirmativas

I – Afirmativa correta.
JUSTIFICATIVA. As variáveis com acesso `private` somente são acessíveis a métodos da classe em que são declarados. É a definição de variáveis privadas.

II – Afirmativa incorreta.
JUSTIFICATIVA. Uma classe pode ter múltiplos métodos construtores, desde que tenham assinaturas diferentes (número ou tipos distintos de parâmetros). Esse processo se chama sobrecarga de construtores.

III – Afirmativa incorreta.
JUSTIFICATIVA. Classes abstratas não são instanciadas. O que pode acontecer é uma subclasse concreta herdar atributos e métodos da classe abstrata.

IV - Afirmativa correta.
JUSTIFICATIVA. O polimorfismo elimina a necessidade de múltiplos comandos condicionais, como o `switch`, permitindo que chamadas de métodos sejam resolvidas de acordo com o tipo do objeto.

Alternativa correta: C.

3. Indicações bibliográficas

- ARAÚJO, E. C. *Orientação a objetos com Java.* São Paulo: Visual Books, 2008.
- CARDOSO, C. *Orientação a objeto na prática*. São Paulo: Ciência Moderna, 2020.
- WEST, D. *Use a cabeça! Análise e projeto orientado ao objeto*. Porto Alegre: Alta Books, 2008.

ÍNDICE REMISSIVO