Universidade Federal do Rural do Semi-Árido
Centro Multidisciplinar de Pau dos Ferros
Departamento de Engenharias e Tecnologia
PEX0130 - Programação Orientada a Objetos
Docente: Ítalo Assis
Monitor: Heitor Claudino

Lista de Exercícios

Sumário

1 Criação de classes - parte 1

1. Escreva uma classe para representar um time de um esporte qualquer em um campeonato desse esporte. Que atributos devem ser representados nessa classe? Quais métodos ela deve conter? Escreva um aplicativo de teste que demonstre as capacidades da classe criada.

2. Crie uma classe chamada $Fatura$ para que uma loja de suprimentos de informática a utilize para representar uma fatura de um item vendido. Uma Fatura deve incluir as seguintes informações sobre o item vendido: o número de identificação, a descrição, a quantidade comprada e o preço unitário. Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço por item não for positivo, ele deve ser configurado como 0.0. Forneça também um método chamado $calculaTotal$ que calcula e retorna o valor de fatura (isto é, multiplica a quantidade pelo preço por item). Escreva um aplicativo de teste que demonstre as capacidades da classe criada.

3. Escreva uma classe cujos objetos representam alunos matriculados em uma disciplina da Ufersa. Cada objeto dessa classe deve guardar os seguintes dados do aluno: matrícula, nome, 2 notas de prova ($P1$ e $P2$) e 1 nota de trabalho (T). Escreva os seguintes métodos para esta classe:

   - $media$: calcula a média parcial do aluno (MP)
     - cada prova tem peso $2,5$ e o trabalho tem peso 2 ($MP = \frac{2,5 \times P1 + 2,5 \times P2 + 2 \times T}{7}$)
   - $provaFinal$: calcula quanto o aluno precisa para o exame final (EF)
     - retorna zero se o aluno não necessita realizar o exame final ($MP < 3$ ou $MP >= 7$)
     - média final MF = $(MP \times 6 + EF \times 4)/10$
     - é necessário que MF seja maior ou igual a 5 para que o aluno seja aprovado após realizar o exame final

   Escreva um aplicativo de teste que demonstre as capacidades da classe criada.

4. [FAB - EEAR - 2023 - Sargento da Aeronáutica - Informática] Relacione as colunas quanto aos conceitos de POO. Em seguida, assinale a alternativa com a sequência correta.

   1 – Classe 2 – Objeto 3 – Construtor

   ( ) É uma estrutura dinâmica que encapsula estado e comportamento.

   ( ) É executado automaticamente quando um objeto é instanciado utilizando o operador “new”.

   ( ) Trata-se de modelo ou template que permite a criação de idênticas estruturas dinâmicas.

   (A) 1 - 2 - 3

   (B) 2 - 3 - 1

   (C) 3 - 1 - 2

   (D) 3 - 2 - 1

5. [UFES - 2023 - Técnico de Tecnologia da Informação] Considere o seguinte programa escrito em Java:

```
class Conta {
    double saldo;
    void sacar(double qtd) { saldo -= qtd; }
    void depositar(double qtd) { saldo += qtd; }
}

public class Programa {
    public static void main(String[] args) {
        Conta c1 = new Conta();
        c1.depositar(1000);
        Conta c2 = c1;
        c1.sacar(500);
        System.out.println(c1.saldo);
    }
}
```

Ao compilar e executar esse programa, o valor impresso na tela é:

(A) 1000.0

(B) 500.0

(C) 500

(D) 1000

(E) Indeterminado, pois o saldo não foi inicializado.

2 Criação de classes - parte 2

6. Crie uma classe denominada *Elevador* para armazenar as informações de um elevador dentro de um prédio. A classe deve armazenar o número do andar atual, o total de andares no prédio, a capacidade do elevador e quantas pessoas estão presentes nele. Outras classes não devem ter acesso direto aos atributos de *Elevador*. A classe deve também disponibilizar os seguintes métodos:

   - construtor : que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (um elevador sempre começa no térreo e vazio);
   - *entra* : para acrescentar uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
   - *sai* : para remover uma pessoa do elevador (só deve remover se houver alguém dentro dele);
   - *sobe* : para subir um andar (não deve subir se já estiver no último andar);
   - *desce* : para descer um andar (não deve descer se já estiver no térreo).

   Escreva um aplicativo de teste que demonstre as capacidades da classe criada, ou seja, execute todos os métodos da classe.

7. Escreva a classe *Contador* que encapsule um valor usado para contagem de eventos. Esta classe deve esconder o valor encapsulado de programadores-usuários, fazendo com que o acesso ao valor seja feito através de métodos que devem zerar, incrementar e imprimir o valor do contador. Escreva um aplicativo de teste que demonstre as capacidades da classe criada.

8. Escreva uma classe *Lampada* que possui o atributo *estadoDaLampada* (ligado ou desligado) e os métodos *acende*, *apaga* e *mostraEstado* (ex.: *A lampada esta acesa*). Inclua um método *estaLigada* que retorne verdadeiro se a lâmpada estiver ligada e falso caso contrário. Adicione um campo que indique quantas vezes a lâmpada foi acesa. Para isso, utilize uma instância da classe *Contador* criada anteriormente e implemente a lógica necessária para atualizar seu valor. Escreva um aplicativo de teste que demonstre as capacidades da classe criada.

9. Escreva uma classe para representar um número complexo. Essa classe deve conter três construtores. Um construtor deverá receber os dois valores (parte real e parte imaginária) como argumentos, o outro somente o valor real, considerando o imaginário como sendo zero, e o terceiro construtor não recebe argumentos, considerando as partes real e imaginária do número complexo como sendo iguais a zero. Escreva um método *toString* que apresente o número complexo em notação apropriada (ex.: $2 + 4i$) e o use em um aplicativo de teste que demonstre as capacidades da classe criada.

10. Escreva uma classe que contenha métodos estáticos para retornar o maior de dois, três, quatro e cinco valores, considerando que os argumentos e o retorno dos métodos podem ser dos tipos int e double. Em outra classe, escreva um aplicativo de teste que, sem criar objetos, demonstre as capacidades da classe criada.

11. Escreva uma versão da classe *RegistroAcademico* (vista em aula) que tenha o campo *numeroDeMatriculas* declarado como *static*, e que incremente o valor deste campo cada vez que uma instância da classe for criada. Use o atributo *numeroDeMatriculas* para definir o atributo matricula automaticamente com um valor diferente para cada instância. Escreva também uma aplicação que crie algumas instâncias da classe para demonstrar seu funcionamento.

12. Escreva uma classe genérica com 3 atributos de um mesmo tipo genérico. Além do construtor, essa classe deve ter:

    - Um método que diz quantos dos 3 atributos são iguais;
    - Um método que imprime os 3 atributos.

    Escreva também uma classe executável para demonstrar o uso da classe criada com diferentes tipos de dados.

13. [FGV - 2024 - CVM - Analista CVM - Perfil 8 - TI / Sistemas e Desenvolvimento] Maria está desenvolvendo um aplicativo desktop, com base em um ambiente de janelas, e precisa que alguns processos sejam disponibilizados de forma global no aplicativo, sem a necessidade de instanciar um objeto específico, e de forma que qualquer entidade possa acessar.

    Por estar trabalhando dentro da metodologia orientada a objetos, Maria precisará adicionar aos métodos globais os modificadores:

    (A) privado e estático;
    (B) protegido e sobrecarregado;
    (C) público e abstrato;
    (D) protegido e sobrescrito;
    (E) público e estático.

14. [IV - UFG - 2024 - Prefeitura de Rio Branco - AC - Analista de Tecnologia da Informação] Qual recurso da Análise e Projeto Orientado a Objetos separa os aspectos externos de um objeto dos detalhes internos da implementação, possibilitando alterar a implementação de um objeto sem afetar as aplicações que o utilizam?

    (A) Polimorfismo
    (B) Encapsulamento
    (C) Herança
    (D) Especilização

15. [FAB - CIAAR - 2023 - Oficial de Apoio da Aeronáutica - Análise de Sistema] As classes são fundamentais na orientação a objeto. Sobre esse assunto, marque a opção que apresenta a composição de uma classe de forma correta.

    (A) Nome da classe, operações e armazenamento de dados.
    (B) Nome da classe, método construtor, armazenamento na sessão.

(C) Nome da classe, atributos, métodos e visibilidade.

(D) Nome da classe, configuração e implementação.

16. [PR-4 UFRJ - 2023 - UFRJ - Técnico de Tecnologia da Informação - Desenvolvimento] Considere o código em Java a seguir:

```
public class ContaCorrenteSobrecarga {
    int conta;
    int agencia;
    double saldo;

    void efetuarSaque(double valor) {
        this.saldo = this.saldo - valor;
    }

    void efetuarDeposito(double valor) {
        this.saldo = this.saldo + valor;
    }

    void imprimirAtributos(int a) {
        System.out.println("Imprime Atributos!");
    }

    void imprimirAtributos(char a) {
        System.out.println("Imprime Atributos!");
    }

    int imprimirAtributos(char a) {
        System.out.println("Imprime Atributos!");
    }
}
```

Baseado no código acima, assinale a alternativa correta.

(A) Um erro de compilação será indicado, pois existem métodos com a mesma assinatura.

(B) Os métodos efetuarSaque e efetuarDeposito não podem ter o mesmo parâmetro.

(C) A palavra this está sendo usada de forma incorreta.

(D) Um erro de compilação será indicado, pois não há um método construtor.

(E) Métodos de mesmo nome não podem ter parâmetros diferentes.

3 Arrays e ArrayLists

17. Crie uma classe *EntradaEmAgenda* que contenha:

    - os dados necessários para armazenar uma entrada de agenda (hora, dia, mês, ano e assunto);
    - um construtor;
    - um método *toString*;
    - um método *ehNoDia* que recebe valores de dia, mês e ano e retorna *true* se o dia, mês e ano daquela instância da classe forem iguais aos argumentos passados.

    Crie também uma classe *Agenda* que:

    - encapsule uma agenda de compromissos representada por um *ArrayList* de instâncias da classe *EntradaEmAgenda*;
      - este item é obrigatório. Caso não seja contemplado, toda a questão será desconsiderada
    - implemente um método construtor;
    - possua um método para adicionar um novo compromisso à lista de compromissos;
    - tenha um método *listaDia* que recebe valores de dia, mês e ano e lista todas as instâncias de *EntradaEmAgenda* que caem naquele dia, mês e ano.

    Por fim, escreva uma classe executável que crie uma *Agenda*, adicione a ela 2 compromissos e, usando o método *listaDia*, liste as entradas da agenda que tem a mesma da data do seu aniversário.

18. Escreva uma classe que encapsule uma matriz de tamanho 2×2 de valores do tipo *float* usando um *array* de duas dimensões. Nesta classe, além do construtor, escreva um método que calcule o determinante da matriz encapsulada e um método que permita a impressão em formato matricial dos seus valores. Escreva um aplicativo de teste que demonstre as capacidades da classe criada.

19. Crie um objeto de uma classe chamada *Cliente* com os atributos *id*, *nome*, *idade*, *telefone*. Faça um programa para solicitar os dados de vários clientes e armazenar em um *ArrayList* até que se digite um número de *id* negativo. Em seguida, exiba os dados de todos os clientes.

20. Escreva um aplicativo que calcula o produto de uma série de inteiros que são passados para um método *produto* utilizando uma lista de argumentos de comprimento variável. Escreva também uma classe executável que teste seu método com várias chamadas, cada uma com um número diferente de argumentos.

21. Escreva um aplicativo para simular o lançamento de dois dados. O aplicativo deve utilizar um objeto de classe *Random*, uma vez para lançar o primeiro dado e novamente para lançar o segundo dado. A soma dos dois valores deve então ser calculada. Cada dado pode mostrar um valor inteiro de 1 a 6, portanto a soma dos valores irá variar de 2 a

12, com 7 sendo a soma mais frequente e 2 e 12, as somas menos frequentes. Seu aplicativo deve lançar o dado 36.000.000 vezes. Utilize um *array* unidimensional para contar o número de vezes que cada possível soma aparece. Exiba os resultados.

22. [FCC - 2022 - Analista Judiciário - Tecnologia da Informação] Considere o código Java abaixo.

```
public class Main {
    public static void main(String args[]) {
        int dados[][] = new int[5][];
        for(int i = 0; i < 5; i++) {
            dados[i] = new dados[5];
            for(int j = 0; j < 5; j++) {
                dados[i][j] = i + j;
                System.out.println(dados[i][j]);
            }
        }
    }
}
```

Sobre o código

(A) Ocorrerá uma exceção do tipo NullPointerException, pois na terceira linha não foi definido o número de linhas da array dados.

(B) Ao tirar o comando `dados[i] = new int[5];` o programa executará normalmente.

(C) `i < 5` na quarta linha pode ser substituído por `i < dados.length` sem afetar a lógica de execução.

(D) O comando `int[][] dados = new int[5][];` precisa obrigatoriamente ser substituído por `int[][] dados = new int[5][5];`.

(E) Ocorrerá uma exceção do tipo NullPointerException no comando dados `[i][j] = i + j;`.

23. [UFMA - 2023 - Analista de Tecnologia da Informação] Na linguagem de programação JAVA, qual das seguintes alternativas representa uma declaração válida?

(A) char[ ] ch = new char[5]

(B) char[ ] ch = new char(5)

(C) char[ ] ch = new char( )

(D) char[ ] ch = new char[ ]

(E) char[] ch = new [5]

24. [CONSULPLAN - 2017 - TRF - 2ª REGIÃO - Analista Judiciário - Informática Desenvolvimento] "Um array em Java é uma coleção ordenada que ocupa uma porção fixa e sequencial da memória. Além disso, é definido como uma estrutura homogênea, pois armazena um determinado tipo de dado. Esse, por sua vez, faz referências para objetos, valores de um tipo primitivo ou para outros arrays." Considere que o usuário digitou os valores: 14, 40, 16, 22 e 60 para o array. Assinale a alternativa que contém o valor que será exibido quando executado o código Java a seguir.

```
for (i = 0; i < 5; i++) {
    for (j = 0; j < 4; j++) {
        if (vetor[j] < vetor[j + 1]) {
            aux = vetor[j];
            vetor[j] = vetor[j + 1];
            vetor[j + 1] = aux;
        }
    }
}
```

(A) 14, 16, 22, 40 e 60.

(B) 28, 32, 44, 80 e 120.

(C) 60, 40, 22, 16 e 14.

(D) 120, 80, 44, 32 e 28.

25. [CS-UFG - 2024 - Câmara de Anápolis - GO - Analista Administrativo - Analista de Sistemas] Considere o trecho abaixo de código escrito na linguagem Java

```
int index, aux, i, j;
for (i = 0; i <= 3; i++) {
    index = i;
    for (j = i + 1; j <= 4; j++) {
        if (v[j] < v[index]) {
            index = j;
        }
    }
    if (index != i) {
        aux = v[index];
        v[index] = v[i];
        v[i] = aux;
    }
}
```

Suponha um vetor inteiro v de tamanho 5. Ao entrar com valores v = [3, 7, 2, 3, 10] a saída deste vetor após passar pelo código será

(A) v = [10, 7, 3, 3, 2].

(B) v = [2, 3, 10, 3, 7].

(C) v = [3, 3, 2, 10, 7].

(D) v = [2, 3, 3, 7, 10].

26. [FADE - UFPE - 2023 - UFPE - Analista de Tecnologia da Informação - Área: Sistemas] Considere o seguinte código em Java:

```
    public class Main {
        public static void main(String[] args) {
```

```
        Integer[] A = new Integer[3];
        A[0] = 2;
        A[1] = 3;
        A[2] = 9;
        proc(A, A[2]);
        System.out.println(A[0] + A[1] + A[2]);
    }

    private static void proc(Integer[] B, Integer c) {
        B[0] = c;
        B[2] = B[0] + B[1];
    }
}
```

A execução desse código resulta na impressão do seguinte valor:

(A) 14

(B) 17

(C) 24

(D) 11

(E) 21

27. [FADE - UFPE - 2023 - UFPE - Analista de Tecnologia da Informação - Área: Sistemas] Considere o seguinte código em Java:

```
public class Main {
    public static void main(String[] args) {
        int[] A = {12, 25, 16, 11, 9, 31, 23, 27, 22};
        System.out.println(A[A[2] - A[4]] - A[6]);
    }
}
```

A execução desse código resulta na impressão do seguinte valor:

(A) 0

(B) 1

(C) 2

(D) 3

(E) 4

4 Reúso de Classes

28. Crie as classes *Equipamento* e *Computador*, cada uma com dois atributos privados à sua escolha. Além disso, a classe *Computador* deverá herdar os métodos e atributos da classe *Equipamento*. Escreva métodos de acesso, *get*'s e *set*'s, para os atributos definidos em ambas as classes. Cada classe também deve ter um método *toString*. Lembre-se que o método *toString* de *Computador* também deve representar os atributos herdados. Por fim, crie uma classe executável, *TestaEquipamento*, para instanciar um objeto de cada classe, inicializar seus atributos e imprimí-los.

29. Crie uma classe para representar uma data e um horário (*DataHora*).

    - Escreva uma classe *EventoDelegacao* que seja baseada na classe *DataHora* e que contenha um campo para indicar qual o evento que ela representa (use uma *String* para isto). Use o mecanismo de delegação para criar a classe *EventoDelegacao*;
    - Escreva uma classe *EventoHeranca* que seja baseada na classe *DataHora* e que contenha um campo para indicar qual o evento que ela representa (use uma *String* para isto). Use o mecanismo de herança para criar a classe *EventoHeranca*;
    - Escreva um aplicativo de teste que demonstre o uso das classes criadas.

30. Escreva as classes *LivroLivraria* e *LivroBiblioteca* que herdam da classe *Livro*. Quais as diferenças entre as duas classes e que campos elas têm em comum? Defina os atributos de cada classe e escreva um aplicativo de teste que demonstre o uso das classes criadas.

31. Crie uma classe *Pessoa* com ao menos 2 atributos a sua escolha. Escreva a classe *Politico* que herda da classe *Pessoa* e tem um campo adicional para representar o partido do político. Escreva também as classes *Prefeito* e *Governador* que herdam da classe *Politico* e que contém campos para representar a cidade ou estado governado. Todos atributos devem ser privados. Cada classe deve ter um construtor e um método *toString*. Lembre-se que cada método *toString* deve representar todos os atributos, inclusive os herdados. Escreva também uma aplicação que demonstre o uso de instâncias destas classes.

32. Implemente a classe *Funcionario* com nome, salário e os métodos:

    - *aumentarSalario* : recebe o valor do aumento e o adiciona ao salário;
    - *ganhoAnual* : computa o valor recebido em 12 meses e o 13^o;
    - e *toString* : retorna uma representação textual de um objeto de *Funcionario*.

    (i) Crie também a classe *Assistente*, que também é um funcionário e que possui um número de matrícula (e seus métodos de acesso), além de um método *toString*.

    (ii) Escreva as classes *Tecnico* e *Administrativo*

    - Ambas as classes são filhas da classe *Assistente*
    - Ambas as classes devem ter um método *ganhoAnual*
    - Assistentes Técnicos possuem um bônus salarial

- Assistentes Administrativos possuem um turno (dia ou noite) e um adicional noturno

(iii) Lembre-se que o 13^o não possui adicional noturno, mas pode possuir bônus salarial (se aplicável).

33. [FGV - 2024 - INPE - Tecnologista Júnior I - Desenvolvimento de Software para Operação de Satélites] (Adaptada) Uma linguagem de programação Orientada a Objetos deve prover suporte aos principais fundamentos do desenvolvimento Orientação a Objetos. Entretanto, cada linguagem apresenta as suas especificidades e formas de implementar esses fundamentos.

    Sobre a Linguagem Java, analise as afirmativas a seguir.

    I A palavra-chave this em Java é uma referência ao próprio objeto da classe e pode ser usada para acessar atributos e métodos da instância atual.

    II Quando uma classe ClasseB estende ClasseA com extends, significa que ClasseB pode acessar membros privados e protegidos de ClasseA.

    III A linguagem Java permite herança múltipla de classes.

    (A) I, apenas.

    (B) I e II, apenas.

    (C) I e III, apenas.

    (D) II e III, apenas.

    (E) I, II e III.

34. [FAB - EEAR - 2023 - Sargento da Aeronáutica - Informática] Herança é um dos conceitos fundamentais da Programação Orientada a Objetos. Analise as afirmativas abaixo sobre esse importante pilar da POO.

    I- As classes inferiores da hierarquia não herdam automaticamente todas as propriedades e os métodos das classes superiores.

    II- A herança permite basear uma nova classe na definição de uma classe previamente existente.

    III- A classe filha é conhecida como superclasse e a classe progenitora como subclasse.

    IV- A herança permite o agrupamento de classes relacionadas.

    Está correto o que se afirma em

    (A) I e II

    (B) I e IV

    (C) II e III

    (D) II e IV

35. [CESGRANRIO - 2024 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação] No contexto de orientação a objeto, para as classes P, Q, R, S, T, U, sendo Q uma classe declarada como abstrata, considere a hierarquia de classes a seguir:

- U e R herdam diretamente de S
- S e T herdam diretamente de Q
- P herda de T

Nesse contexto, é possível criar uma instância de

(A) P e associar a uma variável da classe S
(B) Q e associar a uma variável da classe Q
(C) Q e associar a uma variável da classe P
(D) S e associar a uma variável da classe U
(E) U e associar a uma variável da classe Q

36. [FGV - 2024 - CGM de Belo Horizonte - MG - Auditor Interno - Ciência da Computação - Manhã] Com relação à programação orientada a objetos usando Java, avalie o código a seguir.

```
public class Veiculo {

    private String marca;

    private String modelo;

    public Veiculo(String marca, String modelo) {

        this.marca = marca;

        this.modelo = modelo;

    }

    public String getMarca() {

        return marca;

    }

    public String getModelo() {

        return modelo;

    }

}

public class Mobi extends Veiculo {

    private int portas;
```

```
        public Mobi(String marca, String modelo, int portas) {

            super(marca, modelo);

            this.portas = portas;

        }

        public int getPortas() {

            return portas;

        }
    }
```

Em relação ao código, analise as afirmativas a seguir e assinale (V) para a verdadeira e (F) para a falsa.

( ) A classe Mobi é um exemplo de herança, um dos fundamentos da programação orientada a objetos.

( ) A classe Veiculo não pode ser instanciada porque é uma classe abstrata.

( ) O método getPortas() é um exemplo de encapsulamento, outro fundamento da programação orientada a objetos.

As afirmativas são, respectivamente,

(A) V – V – F.

(B) V – F – V.

(C) F – V – V.

(D) F – F – V.

37. [IV - UFG - 2024 - Prefeitura de Rio Branco - AC - Analista de Sistemas - Especialização em Desenvolvimento Front-End] Desenvolver aplicações eficientes requer reutilizar e estender o comportamento de classes existentes. No contexto da programação orientada a objetos, qual conceito permite que uma classe adquira as propriedades e métodos de outra classe?

(A) Encapsulamento.

(B) Abstração.

(C) Composição.

(D) Herança.

5 Classes abstratas, polimorfismo e interfaces

38. Explique por que não podemos ter construtores declarados com a palavra-chave *abstract*.

39. Defina uma classe para conter informações sobre um funcionário de uma empresa (classe *Funcionario*). Quais são os atributos dessa classe? Inclua entre eles o salário que o funcionário deve receber por hora trabalhada. Implemente, para essa classe, pelo menos dois métodos construtores: um que receba apenas o nome do funcionário e assuma valores padrão para os demais atributos (assuma que o funcionário deve receber dois reais por hora trabalhada); o segundo construtor deve receber, além do nome, o valor que o referido trabalhador deve receber por hora trabalhada. Identifique e implemente os demais métodos que achar conveniente para um objeto da classe *Funcionario*.

40. Crie a classe *FiguraGeometrica* que possui um método abstrato *descricao()*. Crie também as classes *Circulo*, *Quadrado* e *Triangulo* que são subclasses da classe *FiguraGeometrica* e implementam o método *descricao()* apropriado para sua classe. Por fim, crie uma classe *Principal* com um método *main* que cria um objeto de cada uma das classes e chama seus respectivos métodos *descricao()*.

    - O método *descricao()* deve exibir um texto que descreva a figura.

41. Crie uma classe *Desenho* que possui dois atributos do tipo *FiguraGeometrica* (criado na questão anterior) e suas respectivas coordenadas em um plano bidimensional. Escreva um construtor para a classe *Desenho* que inicialize todos os atributos através dos parâmetros. Implemente também o método *apresenta()* que, para cada *FiguraGeometrica* em um *Desenho*, informa suas coordenadas e imprime sua descrição. Por fim, crie uma classe executável, *Principal*, que cria dois objetos do tipo *Desenho* e chama seu meu método apresenta. O primeiro *Desenho* deve ser formado por um *Circulo* e um *Quadrado* e o segundo por um *Quadrado* e um *Triangulo*.

42. Crie uma interface *ItemDeBiblioteca* que declara quais campos e métodos uma classe que representa um item para empréstimo em uma biblioteca deve implementar. Essa interface é composta por um campo *maximoDeDiasParaEmprestimo* com valor 14 e os seguintes métodos:

    - *estaEmprestado* : retorna verdadeiro se o item estiver emprestado e falso caso contrário;
    - *empresta* : modifica para verdadeiro o estado de um campo que indica se o item está emprestado ou não;
    - *devolve* : modifica para falso o estado de um campo que indica se o item está emprestado ou não;
    - *localizacao* : retorna um texto que informa o local do item na biblioteca (e.g: "corredor 2, prateleira D");
    - *descricao* : retorna texto contendo uma descrição resumida do item (e.g.: "artigo da ECOP").

Implemente também a classe *Livro* que encapsula os dados genéricos sobre um livro e métodos para processar estes dados. Essa classe é composta pelos atributos *titulo*, *autor*, *numeroDePaginas* e *anoDaEdicao*, além dos seguintes métodos:

- construtor;
- *qualTitulo*: retorna o título do livro;
- *qualAutor*: retorna o autor do livro;
- *toString*: retorna os valores dos campos desta classe em formato textual.

Em seguida, escreva a classe *LivroDeBiblioteca* que herda os campos e métodos da classe *Livro* e implementa os métodos declarados na interface *ItemDeBiblioteca*. *LivroDeBiblioteca* também deve possuir um construtor e um método *toString*. Crie os atributos que forem necessários.

Por fim, crie a classe *DemoLivroDeBiblioteca* para demonstrar o uso de uma instância da classe *LivroDeBiblioteca*, isto é, criar um objeto do tipo *LivroDeBiblioteca* e executar seus métodos.

43. [FCC - 2023 - Copergás - PE - Analista / Suporte de Informática] A sobrecarga de métodos ocorre quando

    (A) um método de mesmo nome existe em classes diferentes em uma relação de herança.

    (B) há mais de um método com mesmo nome, recebendo parâmetros de tipos ou em quantidades diferentes, em uma classe.

    (C) um método recebe mais parâmetros do que pode suportar.

    (D) o mesmo método aparece na maioria das classes da aplicação.

    (E) um método recebe como parâmetro um valor que extrapola a capacidade do tipo usado para o recebimento.

44. [IDECAN - 2019 - IF-PB - Professor - Informática] Dadas as seguintes classes, todas no mesmo pacote:

```
public class Mamifero {
    protected void andar() {
        System.out.print("Mamifero andando");
        ouvir();
    }
    protected void ver() {
        System.out.print("Mamifero vendo");
    }
    protected void ouvir() {
        System.out.print("Mamifero ouvindo");
        ver();
    }
}
```

```
public class Primata extends Mamifero {
    protected void andar() {
        System.out.print("Primata andando");
        ouvir();
    }
}

public class Homem extends Primata {
    protected void ver() {
        System.out.print("Homem vendo");
    }
    public static void main(String[] args) {
        Mamifero m = new Homem();
        m.andar();
    }
}
```

Qual é a saída deste programa?

(A) Mamífero andando Mamífero ouvindo Mamífero vendo

(B) Primata andando Mamífero ouvindo Homem vendo

(C) Uma exception será lançada: MethodNotFoundException

(D) Mamífero andando Mamífero ouvindo Homem vendo

(E) Primata andando Mamífero ouvindo Mamífero vendo

45. [PETROBRAS - 2012 - Analista de Sistemas Júnior - Engenharia de Software] Suponha que as classes Circulo, Desenho e Figura ocupem arquivos separados. Em qual código Java elas serão compiladas sem erros?

(A) ---

```
package P1;
import P2.*;
public class Figura {
    protected double x,y;
    protected final double PI=0;
    Desenho d;
    abstract protected double dist(double x1,double y1);
}

package P1;
public class Circulo extends Figura {
     double r;
     public Circulo() {
        d.add(this);
        PI=3.14159;
     }
     public double raio() { return r; }
     public double centroX() { return x; }
```

```
     public double centroY() { return y; }
     protected double dist(double x1,double y1) {
        return Math.sqrt((x1-x)*(x1-x)+(y1-y)*(y1-y));
     }
}

package P2;
import java.util.List;
import P1.Figura;
public class Desenho {
    List<Figura> f;
    public void add(Figura p) { f.add(p); }
}
```

(B)

```
    package P1;
    import P2.*;

    abstract public class Figura {
        protected double x,y;
        protected final double PI=0;
        Desenho d;
        abstract protected double dist(double x1,double y1);
    }

    package P1;
    public class Circulo extends Figura {
     double r;
     public Circulo() {
        d.add(this);
        PI=3.14159;
     }
     public double raio() { return r; }
     public double centroX() { return x; }
     public double centroY() { return y; }
     private double dist(double x1,double y1) {
        return Math.sqrt((x1-x)*(x1-x)+(y1-y)*(y1-y));
     }
    }

package P2;
import java.util.List;
import P1.Figura;

public class Desenho {
 List<Figura> f;
 public void add(Figura p) { f.add(p); }
}
```

(C)

```java
package P1;
import P2.*;

abstract public class Figura {
 double x,y;
 final double PI=3.14159;
 Desenho d;
 abstract protected double dist(double x1,double y1);
}

package P1;

public class Circulo extends Figura {
 double r;
 public Circulo() { d.add(this); }
 public double raio() { return r; }
 public double centroX() { return x; }
 public double centroY() { return y; }
 public double dist(double x1,double y1) {
    return Math.sqrt((x1-x)*(x1-x)+(y1-y)*(y1-y));
 }
}

package P2;

import java.util.List;
import P1.Figura;
public class Desenho {
 List<Figura> f;
 public void add(Figura p) { f.add(p); }
}
```

(D)

```java
package P1;
import P2.*;

public class Circ implements ICirculo {
 double cx, cy, r;
 public double raio() { return r; }
 public double centroX() { return cx; }
}

package P2;

public interface ICirculo {
 double PI;
 double raio();
 double centroX();
 double centroY();
}
```

(E)

```
package P1;
import P2.*;

public class Circ extends ICirculo {
 double cx, cy, r;
 public double raio() { return r; }
 public double centroX() { return cx; }
 public double centroY() { return cy; }
}

package P2;

public interface ICirculo {
 double PI=3.14159;
 double raio();
 double centroX();
 double centroY();
}
```

46. [PETROBRAS - 2011 - Analista de Sistemas Júnior - Engenharia de Software] Analise os fragmentos de código dados abaixo

```
package javaapplication1;
public interface Interface1 {
    public int metodoComum();
}

package javaapplication1;
public class Concreta1 implements Interface1 {
    public int metodoComum() {
        return(1);
    }

    public int metodoExotico() {
        return(2);
    }
}

package javaapplication1;
public class Main {

    public static void main(String[] args) {
        Interface1 x = new Concreta1();
        System.out.println(x.metodoComum());
        System.out.println(x.metodoExotico());
    }
}
```

O resultado, obtido ao tentar compilar e executar esse conjunto de classes, será

(A) um erro de compilação, indicando que não é possível fazer uma conversão da classe Concreta1 para a classe Interface1.

(B) um erro de compilação, indicando que, no contexto de x, não existe metodoExotico.

(C) nenhuma saída e um erro em tempo de execução, indicando que, dada a conversão de Concreta1 para Interface1, não é possível acessar metodoExotico.

(D) impressão do número 1, seguida de um erro de tempo de execução, indicando que, dada a conversão de Concreta1 para Interface1, não é possível acessar metodoExotico.

(E) impressão dos números 1 e 2.

47. [UFC - 2013 - Analista de Tecnologia da Informação / Engenharia de Software] Na programação orientada a objetos, a possibilidade de haver mais de um método com o mesmo nome na mesma classe denomina-se:

(a) Herança.

(b) Sobrescrita.

(c) Sobrecarga.

(d) Ligação tardia.

(e) Encapsulamento.

48. [UFSC - 2023 - Técnico de Tecnologia da Informação] Considere as seguintes definições relacionadas à programação orientada a objetos, com lacunas a preencher, e assinale a alternativa que preencha corretamente as três definições, considerando sua ordem.

1. _____ é a capacidade de objetos de classes distintas responderem a uma mesma chamada de método de maneiras diferentes. Isso permite que as subclasses redefinam o comportamento de métodos herdados da classe base.

2. _____ é a capacidade de um objeto ter vários métodos com o mesmo identificador, mas com assinaturas de métodos diferentes. Isso permite que os objetos respondam a chamadas de métodos distintos, mas com identificadores idênticos, com base na quantidade e no tipo de argumentos fornecidos.

3. _____ é a capacidade de uma subclasse substituir o comportamento de um método herdado da classe base. Isso permite que uma classe modifique o comportamento de um método para atender às suas próprias necessidades, mantendo a mesma assinatura de método.

(A) Sobrecarga – Polimorfismo – Herança

(B) Sobrescrita – Polimorfismo – Encapsulamento

(C) Polimorfismo – Sobrecarga – Herança

(D) Herança – Encapsulamento – Sobrescrita

(E) Polimorfismo – Sobrecarga – Sobrescrita

49. [FAB - CIAAR - 2023 - Oficial de Apoio da Aeronáutica - Análise de Sistemas] Existe uma gama de definições sobre a orientação a objetos. No sentido da relação das classes e dos acessos aos métodos, preencha as lacunas abaixo.

    Muitas classes podem ter acesso _____, porém, _____ esse método _____. A sequência de palavras que preenche corretamente as lacunas é:

    (A) a um mesmo método / cada classe executa / de maneira diferente

    (B) a um mesmo método / outras classes executam / da mesma maneira

    (C) somente a um método / outras classes executam / de maneira diferente

    (D) somente a um método / cada classe executa / da mesma maneira

50. [FGV - PGM Niterói - 2023 - RJ • Analista de Tecnologia da Informação] Observe o método liga() do seguinte trecho de código escrito na linguagem Java.

```
class Aparelho {
    public void liga() {}
}

class Geladeira extends Aparelho {
    public void liga() {
        System.out.println("Tomada");
    }
}

class TV extends Aparelho {
    public void liga() {
        System.out.println("Controle remoto");
    }
}

class Main {
    public static void main(String[] args) {
        Aparelho geladeira = new Geladeira();
        Aparelho tv = new TV();
        geladeira.liga();
        tv.liga();
    }
}
```

    Em orientação a objeto, o uso de um método com comportamento diferente, como liga(), é realizado por meio do emprego de:

    (A) Lean;

    (B) Eventos;

    (C) Recursão;

    (D) Polimorfismo;

(E) Encapsulamento.

51. [FAB - EEAR - 2023 - Sargento da Aeronáutica - Informática] Assinale a alternativa que completa a lacuna do texto abaixo.

    A POO oferece um tipo especial de classe que não pode ser instanciada. Trata-se da classe _____

    (A) empacotada
    (B) concreta
    (C) abstrata
    (D) oculta

52. [Instituto Access - UFFS - 2023 - Técnico de Tecnologia da Informação] No que diz respeito à orientação a objetos, um princípio é definido como aquele em que as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas. É um mecanismo por meio do qual selecionam-se as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução. Esse princípio é conhecido por

    (A) encapsulamento.
    (B) polimorfismo.
    (C) acoplamento.
    (D) herança.
    (E) coesão.

53. [UFPR - 2023 - IF-PR - Informática] O polimorfismo permite que uma mesma mensagem, tratada por um mesmo método, enviada para diferentes objetos, apresente resultados diferentes. Isso também permite que objetos com tipos diferentes sejam tratados da mesma forma. A hierarquia de classes Java apresentada utiliza o conceito de polimorfismo. Cada classe está salva em seu arquivo específico com o nome da classe e a extensão .java.

```java
public interface Exame {
    public abstract void mostrarPreparo();
}

public class ExameImagem implements Exame {
    @Override
    public void mostrarPreparo() {
        System.out.println("EXAME DE IMAGEM PREPARO:");
        System.out.println("Nenhum preparo necessario.");
    }
}

public class EcografiaTireoide extends ExameImagem {
}
```

```
public class ExameSangue implements Exame {
    @Override
    public void mostrarPreparo() {
        System.out.println("EXAMES DE SANGUE - PREPARO:");
    }
}

public class GlicemiaJejum extends ExameSangue {
    @Override
    public void mostrarPreparo() {
        System.out.println("GLICEMIA EM JEJUM - PREPARO:");
        System.out.println("Nao ingerir bebidas alcoolicas 72 horas antes do
            exame.");
        System.out.println("Jejum de 8 a 12 horas");
    }
}

import java.util.ArrayList;
import java.util.List;

public class AgendarExame {
    public static void main(String[] args) {
        List<Exame> examesPaciente = new ArrayList<>();
        examesPaciente.add(new GlicemiaJejum());
        examesPaciente.add(new EcografiaTireoide());
        for (Exame exame : examesPaciente) {
            exame.mostrarPreparo();
        }
    }
}
```

Qual o resultado da execução do código descrito no método main da classe AgendarExame?

(A) O método não pode ser executado, pois a classe EcografiaTireoide não tem implementação para o método mostrarPreparo.
GLICEMIA EM JEJUM - PREPARO:
Nao ingerir bebidas alcoolicas 72 horas antes do exame.
Jejum de 8 a 12 horas

(B) GLICEMIA EM JEJUM - PREPARO:
Nao ingerir bebidas alcoolicas 72 horas antes do exame.
Jejum de 8 a 12 horas
EXAME DE IMAGEM PREPARO:
Nenhum preparo necessario.

(C) Nao ingerir bebidas alcoolicas 72 horas antes do exame.
Jejum de 8 a 12 horas
EXAME DE IMAGEM PREPARO:

Nenhum preparo necessario.
GLICEMIA EM JEJUM - PREPARO:

(D) Nao ingerir bebidas alcoolicas 72 horas antes do exame.
Jejum de 8 a 12 horas

(E) O método não pode ser executado, pois Exame não define um tipo para o objeto, já que não é uma classe e sim uma interface.

54. [BIO-RIO - 2016 - Prefeitura de São Gonçalo - RJ - Analista da Área Tecnológica] No que diz respeito à Orientação a Objetos, dois princípios são caracterizados a seguir.

I constitui um mecanismo que tem por objetivo organizar os dados que sejam relacionados, agrupando-os em objetos, reduzindo as colisões de nomes de variáveis e, da mesma forma, reunindo métodos relacionados às suas propriedades. Este padrão ajuda a manter um programa com centenas ou milhares de linhas de código mais legível e fácil de trabalhar e manter.

II constitui um mecanismo a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas. De acordo com este princípio, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes.

Os princípios I e II são conhecidos respectivamente como

(A) coesão e herança.

(B) polimorfismo e coesão.

(C) herança e acoplamento.

(D) acoplamento e encapsulamento.

(E) encapsulamento e polimorfismo.

55. [CESGRANRIO - 2024 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação] Considere um sistema bancário em Java que possui a classe Cliente e suas subclasses, ClientePessoaFisica e ClientePessoaJuridica, onde Cliente é uma classe abstrata. Nesse sistema, um método getDesconto(valor) deve fornecer o cálculo do desconto para um tipo de cliente, de forma que os clientes do tipo pessoa física e os clientes do tipo pessoa jurídica tenham descontos diferenciados. Suponha que, utilizando corretamente os mecanismos associados à herança e ao polimorfismo, se deseje implementar essas classes de modo que o método getDesconto possa ser aplicado indistintamente a qualquer instância que tenha sido declarada como da classe Cliente.

Para atender a essa condição, a implementação dessas classes deve possuir o método getDesconto

(A) apenas na classe Cliente, identificando a qual subclasse o cliente pertence e calculando o desconto por meio de uma estrutura se-então dentro da implementação.

(B) apenas nas classes ClientePessoaFisica e ClientePessoaJuridica, pois não há instâncias da classe Cliente no sistema, já que é uma classe abstrata.

(C) em todas as três classes, sendo possível, nesse caso, que a função getDesconto da classe Cliente seja abstrata.

(D) fora das três classes, dado que uma estrutura do tipo se-então deve ser usada para descobrir qual é a classe adequada.

(E) implementada totalmente em todas as classes, sendo que a função getDesconto da classe Cliente chama a função getDesconto das classes ClientePessoaFisica e ClientePessoaJuridica de acordo com a necessidade.

56. [FGV - 2024 - DATAPREV - ATI - Arquitetura, Engenharia e Sustentação Tecnológica] Em um sistema de gerenciamento de pagamentos, existem as classes Pagamento (classe base), PagamentoCartao e PagamentoBoleto (ambas herdam de Pagamento). A classe Pagamento define o método realizarPagamento(), que é sobrescrito tanto por PagamentoCartao quanto por PagamentoBoleto para implementar comportamentos específicos de cada tipo de pagamento. Considere o seguinte código:

```
public void processarPagamento(Pagamento pagamento) {
    pagamento.realizarPagamento();
}
```

Assinale a opção que indica o conceito de orientação a objetos que está sendo aplicado quando o método realizarPagamento() é chamado em um objeto do tipo Pagamento, mas o comportamento específico é definido pelas subclasses (PagamentoCartao ou PagamentoBoleto).

(A) Encapsulamento, pois o método realizarPagamento() está escondido da implementação interna.

(B) Polimorfismo, pois o método é definido na classe base, mas o comportamento é determinado pelas subclasses.

(C) Herança, pois as classes filhas estão herdando o método realizarPagamento() da classe Pagamento.

(D) Sobrecarga de métodos, pois o método realizarPagamento() está definido com diferentes assinaturas.

(E) Abstração, pois o método realizarPagamento() oculta a complexidade da lógica interna de pagamento.

57. [UFU-MG - 2023 - Analista de Tecnologia da Informação Área 1 - Desenvolvimento de Sites, Aplicações e Sistemas] Considere o caso de orientação a objeto, apresentado no código abaixo, para analisar as asserções apresentadas.

```
public class Phone{
    public void callNumber(long number) {
        System.out.println("Phone: Discando numero: " + number);
    }

    public boolean isRinging(){
        System.out.println("Phone: Verificar se telefone esta tocando");
        boolean ringing = false;
```

```
        return ringing;
    }
}

public class SmartPhone extends Phone{
    public void sendEmail(String message, String address){
        System.out.println("SmartPhone: Enviando E-mail");
    }

    public String retrieveEmail(){
        System.out.println("SmartPhone: Recuperando e-mail");
        String messages = new String();
        return messages;
    }

    public boolean isRinging() {
        System.out.println("SmartPhone: Verificar se smartphone esta tocando");
        boolean ringing = false;
        return ringing;
    }
}

public class App {
    public static void main(String[] args) {
        new App();
    }
    public App() {
        SmartPhone smartPhone = new SmartPhone();
        testPhone(smartPhone);
        testSmartPhone(smartPhone);
    }

    private void testPhone(Phone phone){
        phone.callNumber(34999999999);
        phone.isRinging();
    }

    private void testSmartPhone(SmartPhone phone){
        phone.sendEmail("Oi", "teste@ufu.br");
        phone.retrieveEmail();
    }
}
```

FONTE: FINEGAN, Edward. OCA Java SE 8: Guia de estudos para o exame 1Z0-808. Porto Alegre: Bookman, 2018.

I O caso apresentado demonstra um exemplo simples de herança ao definir a classe “SmartPhone” com uma extensão da classe “Phone”; no entanto, há um erro no construtor App() quando é executada a linha testPhone(smartPhone), visto que o

método testPhone() requer como argumento um objeto do tipo Phone.

II Sabendo-se que o polimorfismo é unidirecional, o método testSmartPhone() não pode ser usado com um objeto Phone como seu argumento.

III A execução da linha testPhone(smartPhone), descrita dentro do construtor App(), terá como resultado as respectivas mensagens: “Phone: Discando numero: 34999999999” e “SmartPhone: Verificar se smartphone está tocando”.

IV A execução da linha testPhone(smartPhone), descrita dentro do construtor App(), terá como resultado as respectivas mensagens: “Phone: Discando numero: 34999999999” e “Phone: Verificar se telefone está tocando”.

Estão corretas apenas as asserções

(A) II e IV.

(B) II e III.

(C) I e III

(D) I e IV.

58. [IFRN - 2009 - Professor - Sistemas de Informação] O código, abaixo, apresenta a implementação de uma classe na linguagem de programação Java. Com base nessa classe, marque a alternativa verdadeira.

```
public final class Password {
    private String senha;
    public String getSenha() {
        return senha;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }
}
```

(A) Esta classe não poderá ser estendida por outras classes, porém seus métodos poderão ser sobrescritos.

(B) Esta classe não poderá ser estendida por outras classes.

(C) Esta classe poderá ser estendida por outra classe.

(D) Esta classe poderá ser estendida por outra classe, porém seus métodos não poderão ser sobrescritos.

59. [FGV - 2024 - Prefeitura de Caraguatatuba - SP - Técnico em Processamento de Dados] (adaptada) Em um jogo de estratégia online, você tem diferentes classes de personagens, como “Guerreiro” e “Mago”, que herdam de uma classe base chamada “Personagem”. A classe base possui um método chamado “ExecutarHabilidade”, que funciona de maneira diferente quando chamado por um personagem guerreiro em comparação com um personagem mago.

Considerando princípios de programação orientada a objetos, assinale a abordagem mais adequada para implementar essa diferenciação.

(A) Utilizar variáveis de controle condicional dentro do método “ExecutarHabilidade” para verificar explicitamente o tipo da instância e ajustar o comportamento.

(B) Cria versões específicas do método “ExecutarHabilidade” para “Guerreiro” e “Mago” com implementações distintas.

(C) Marcar o método “ExecutarHabilidade” como abstrata na classe base “Personagem” e fornecendo implementações específicas nas classes derivadas.

(D) Permitir que “Guerreiro” e “Mago” herdem métodos específicos de diferentes classes e personalizem o comportamento de “ExecutarHabilidade”.

(E) Ocultar a implementação do método “ExecutarHabilidade” na classe base “Personagem” e redefinindo-o nas classes derivadas para comportamentos específicos.

6 Coleções genéricas

60. Escreva um programa que cria um objeto *LinkedList* de 10 caracteres e, então, cria um segundo objeto *LinkedList* contendo uma cópia da primeira lista, mas na ordem inversa. Não devem ser utilizados métodos da Java API para realizar a inversão.

61. Escreva um programa que utilize a estrutura de dados do tipo Mapa para contar o número de ocorrências de cada letra em uma *String*. Por exemplo, a string *"HELLO THERE"* contém dois *H*'s, três *E*'s, dois *L*'s, um *O* , um *T* e um *R*. Exiba os resultados em ordem alfabética.

    - Serão totalmente desconsideradas respostas que não utilizarem um Mapa como elemento principal da estratégia de solução desta questão.

62. Escreva um programa que utilize uma pilha para verificar se uma *String* de entrada formada apenas por *'('* e *')'* está balanceada.

    É proibido utilizar contador(es) na solução desta questão.

    Os parênteses estão balanceados quando, na expressão, para cada abre parênteses há um correspondente fecha parênteses e os pares de parênteses estão aninhados.

    Exemplos de *Strings* de parênteses corretamente balanceadas:

    ```
    (()()()())
    (((()())))
    (()((())()))
    ```

    Exemplos de *Strings* de parênteses não são balanceadas:

    ```
    ((((((())
    ()))
    (()()(()
    ```

63. Escreva um programa que simule cada minuto de um dia de atendimento de um consultório de um médico com as seguintes especificações:

    - O consultório mantém apenas dados número de RG e idade de seus pacientes;
        (i) Crie uma classe para representar um paciente.
    - Pacientes maiores de 60 anos são colocados na fila prioritária e os demais na fila comum;
    - Pacientes da fila prioritária são sempre atendidos primeiro;
    - O primeiro paciente chega ao consultório no momento de sua abertura e a cada 4 minutos um novo paciente chega ao consultório;
        (i) Pesquise como gerar números aleatórios e utilize essa técnica para determinar o RG e a idade de cada paciente.
    - Uma consulta demora 5 minutos e o próximo paciente da fila é chamado;
    - O consultório atende 20 pacientes por dia.

64. [2023 - CS-UFG - if-goiano - Técnico de Tecnologia da Informação] Em um programa orientado a objetos foi implementada uma superclasse chamada Pessoa e duas subclasses de Pessoa chamadas TecnicoDeTI e TecnicoDeLab. Considerando que listaDePessoas se refere à lista de objetos do tipo Pessoa, qual dos seguintes trechos de código escritos em linguagem Java calcula e armazena corretamente, na variável q, a quantidade de objetos do tipo TecnicoDeTI?

(A)
```
int q = 0;
Iterator i = listaDePessoas.iterator();
TecnicoDeTI t;
while (i.hasNext()) {
    t = (Pessoa) i.next();
    if (t instanceof i)
        q++; }
```

(B)
```
int q = 0;
Iterator i = listaDePessoas.iterator();
TecnicoDeTI t;
while (i.hasNext())
    if (i instanceof TecnicoDeTI) q++;
```

(C)
```
int q = 0;
Iterator i = listaDePessoas.iterator();
TecnicoDeTI t;
while (i.hasNext()) {
    t = (Pessoa) i.next();
    if (t instanceof TecnicoDeTI)
        q++; }
```

(D)
```
int q = 0;
Iterator i = listaDePessoas.iterator();
TecnicoDeTI t;
while (i.hasNext())
    if (i instanceof t) q++;
```

65. [APICE - 2021 - DPE-PB - Analista de Desenvolvimento de Sistemas] Em Programação Orientada a Objetos – POO, os tipos genéricos (Generics) têm o propósito de criar conjuntos com consistência entre os tipos. Nas linguagens de programação Java e C#, por exemplo, existem classes e/ou interfaces como a List que funcionam como uma espécie de matriz, com tamanho aumentado de maneira dinâmica, não havendo necessidade de definir quantos objetos serão inseridos. Com base neste assunto e, supondo que todas as condições são satisfeitas para utilização de listas (jdk adequado, importação de bibliotecas e condições para compilação e execução) assinale a alternativa que contém APENAS INFORMAÇÕES INCORRETAS:

(A) Uma maneira de remover todos elementos de uma lista em Java por exemplo, é invocando o método clear( ) da interface List.

(B) O método público remove( ) da inteface List em Java pode ser utilizado para remover elementos de uma lista. Esse é um método que aceita o índice do objeto a ser removido. Por exemplo, supondo que uma lista de nome minhaLista foi inicializada corretamente com cinco elementos, uma forma de remover o primeiro elemento desta lista é utilizar o comando minhaLista.remove(1);

(C) Normalmente, as linguagens de programação implementam métodos ou propriedades para obter a quantidade de objetos inseridos num objeto de lista. Por exemplo, o método size() da interface List em Java é usado para obter o número de elementos nesta lista.

(D) Existem métodos públicos para manipulação de objetos dentro de uma lista. Por exemplo, o método add() da interface List em Java pode ser usado adicionar um elemento numa lista.

(E) É possível ordenar uma lista utilizando seus próprios métodos públicos. Por exemplo, o método sort() da interface List em Java pode ser usado para tal função.

66. [FADE UFPE - 2023 - UFPE - Analista de Tecnologia da Informação - Área: Sistemas] Suponha que, em Java, utilizamos a classe LinkedList para implementar uma estrutura de dados dinâmica. Vamos considerar duas possibilidades para inserção e remoção: i. usar apenas os métodos addFirst() e removeFirst(); ou ii. usar apenas os métodos addLast() e removeLast(). Podemos, então, afirmar que,

(A) no primeiro caso, trata-se de uma pilha e, no segundo, de uma fila.

(B) no primeiro caso, trata-se de uma fila e, no segundo, de uma pilha.

(C) no primeiro caso, trata-se de uma lista encadeada e, no segundo, de uma fila.

(D) em ambos os casos, trata-se de uma fila.

(E) em ambos os casos, trata-se de uma pilha.

67. [IBFC - 2024 - Correios - Analista de Sistemas – Desenvolvimento de Sistemas] Dado o código abaixo na Linguagem Java:

```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        numbers.remove(1);
        System.out.println(numbers);
    }
}
```

Assinale a alternativa que apresenta a sequência correta da saída do programa.

(A) [1, 2, 3]

(B) [1, 3]

(C) [2, 3]

(D) [1, 2]

68. [FGV - 2024 - DATAPREV - ATI - Arquitetura, Engenharia e Sustentação Tecnológica] Considere o seguinte trecho de código:

```
List<Integer> numeros = Arrays.asList(1, 2, 3, 4, 5);
Iterator<Integer> iterador = numeros.iterator();

while (iterador.hasNext()) {
    for(int i = 0; i < numeros.size(); i++) {
        if(i%2==0) {
            iterador.remove(i);
        }
    }
}
```

Esse código tem por objetivo remover as posições pares de uma lista "números".

Assinale a opção que identifica os problemas com esse código.

(A) A remoção de elementos dentro de um laço for deve ser feita usando os métodos next() e remove() do iterador.

(B) O laço for está mal posicionado, e deveria ser usado antes do laço while para iterar corretamente.

(C) O método remove nunca pode ser chamado durante a iteração com um iterador, pois isso sempre resulta em uma exceção.

(D) A variável i utilizada no laço for está causando um erro de indexação ao tentar acessar elementos que já foram removidos.

(E) O laço while deve ser substituído por um laço for para evitar conflitos com o iterador.

7 Referências

SANTOS, R. Introdução à programação orientada a objetos usando JAVA. 2. ed. Rio de Janeiro: Campus, 2013. 336p.

DEITEL, Paul; DEITEL, Harvey. Java: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

BATISTA, Rogério da Silva; MORAES, Rafael Araújo de. Introdução à Programação Orientada a Objetos. 2013. Disponível em: http://proedu.rnp.br/handle/123456789/611. Acesso em: 18 ago. 2021.

SANTANCHÈ, André. Classes: lista de exercícios. 2011. Disponível em: https://www.ic.unicamp.br/~santanch/teaching/oop/exercicios/poo-exercicios-02-classes-v01.pdf. Acesso em: 30 mar. 2022.

BACALÁ JÚNIOR, Sílvio. Revisão de POO em Java: lista de exercícios 1. 2022. Disponível em: http://www.facom.ufu.br/~bacala/POO/lista1.pdf. Acesso em: 30 mar. 2022.

INSTITUTO METRÓPOLE DIGITAL. Programação Orientada a Objetos. 2015. Disponível em: https://materialpublic.imd.ufrn.br/curso/disciplina/5/8. Acesso em: 21 out. 2020. 2ª Edição. ISBN: 978-85-7064-002-4.

GARCIA, Islene Calciolari. MC202 - Estruturas de Dados: Lista de Exercícios 1. Disponível em: https://www.ic.unicamp.br/~islene/mc202/lista1.pdf. Acesso em: 24 maio 2022.

BACALÁ JÚNIOR, Sílvio. Exceções. Disponível em: https://www.facom.ufu.br/~bacala/POO/08-CriandoExcecoes.pdf. Acesso em: 31 maio 2022.

FERREIRA, Nickerson. Lista de Exercícios - Herança. Disponível em: https://docente.ifrn.edu.br/nickersonferreira/disciplinas/programacao-estruturada-e-orientada-a-objetos/lista-de-exercicios-heranca/. Acesso em: 8 mar. 2023.

USP. Parênteses Balanceados. Disponível em: https://panda.ime.usp.br/panda/static/pythonds_pt/03-EDBasicos/06-ParentesesBalanceados.html. Acesso em: 23 mar. 2024.

GOVERNO DO ESTADO DE RONDÔNIA. Edital n. 287-2022-SEGEP-GCP-Abertura Concurso Publico Secretaria de Estado da Assistencia e do Desenvolvimento Social-SEAS-RO-Atuallizado-Retificacao I. 2022. Disponível em: https://rondonia.ro.gov.br/publicacao/16-11-2022-edital-n-287-2022-segep-gcp-abertura-concurso-publico-secretaria-de-estado-da-assistencia-e-do-desenvolvimento-social-seas-ro/. Acesso em 20 ago. 2024.

GOVERNO DO ESTADO DE RORAIMA. Edital n. 001/2022 - Secretaria de Estado da Fazenda de Roraima (SEFAZ/RR). 2022. Disponível em: https://concurso.idecan.org.br/Concurso.aspx?ID=60. Acesso em: 22 ago. 2024.

UNIVERSIDADE FEDERAL DE UBERLÂNDIA. Edital 27/2023 - Pró-Reitoria de Gestão de Pessoas - PROGEP. 2023. Disponível em: https://www.portalselecao.ufu.br/servicos/Edital/cronograma/1334. Acesso em: 27 ago. 2024.

GOVERNO DO ESTADO DE MATO GROSSO - Edital n. 01/2023 - Universidade do Estado de Mato Grosso - UNEMAT. 2023. Disponível em: https://arquivos.qconcursos.

com/regulamento/arquivo/78029/unemat_2023_ptes_edital_n_1-edital.pdf. Acesso em: 31 ago. 2024.

TRIBUNAL REGIONAL DO TRABALHO DA 14ª REGIÃO - RO E AC. Edital n. 01/2022 - Fundação Carlos Chagas. 2022. Disponível em: https://www.concursosfcc.com.br/concursos/trt14122/edital_de_abertura_trt14122_26_09_22.pdf. Acesso em: 31 ago. 2024.

PREFEITURA MUNICIPAL DE RIO BRANCO - AC. Edital complementar n. 01/2024 - Instituto Verbena - Universidade Federal de Goiás. 2024. Disponível em: https://sistemas.institutoverbena.ufg.br/2024/concurso-rio-branco-ed1/sistema/arquivos/editais/EDITAL_COMPLEMENTAR_N%C2%BA.1.pdf. Acesso em: 30 ago. 2024.

FUNDAÇÃO GETÚLIO VARGAS. Edital n. 1/2024 - Comissão de Valores Mobiliários - CVM. 2024. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77433/cvm_2024_edital_n_1-edital.pdf. Acesso em: 30 ago 2024.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA - Edital n. 148/2018 - Instituto de Desenvolvimento Educacional, Cultural e Assistencial Nacional. 2012. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/18598/if_pb_2018_edital_n_148-edital.pdf. Acesso em: 31 ago. 2024.

COMPANHIA PERNAMBUCANA DE GÁS - COPERGÁS - Edital n. 01/2022. - Fundação Carlos Chagas. 2022. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/73666/copergas_pe_2022_edital_n_1-edital.pdf. Acesso em: 31 ago. 2024.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS - Edital n. 01/2023 - Fundação Getúlio Vargas. 2023. Disponível em: https://conhecimento.fgv.br/sites/default/files/concursos/edital-retificado-_tecnologista.pdf. Acesso em: 31 ago. 2024.

IFRN. Concurso público para Professor de Sistemas de Informação - Edital Nº. 04/2009-DIGPE.2009. Disponível em: https://www.pciconcursos.com.br/provas/download/professor-sistemas-de-informacao-if-rn-if-rn-2009. Acesso em: 09 set. 2024.

PETROBRAS - Edital n. 5, PSP-RH-1/2012 - Processo Seletivo Público para Preenchimento de Vagas e Formação de Cadastro em Cargos de Nível Superior e de Nível Médio - 2012. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/1884/petrobras_2012_diversos_cargos-edital.pdf. Acesso em: 11 set. 2024.

PETROBRAS - Edital n.4, PSP-RH-1/2011 - Processo Seletivo Público para Preenchimento de Vagas e Formação de Cadastro Reserva em Cargos de Nível Superior e de Nível Médio. 2011. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/1346/petrobras_2011_edital_psp_rh_1_2011-edital.pdf. Acesso em: 25 set. 2024.

UFC - Edital n.262/2013 - Concurso Público para Provimento de Cargos Técnico-Administrativos em Educação. 2013. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/17365/ufc_2013_edital_n_262-edital.pdf. Acesso em: 29 set. 2024.

UFSC - Edital n. 002/2023/DDP - do Concurso Público para provimento de cargos da carreira Técnico-Administrativa em Educação (TAE). 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77063/ufsc_2023_edital_n_2-edital.pdf. Acesso em: 16 out. 2024.

UFRJ - Edital n. 843/2023 - Concurso Público para provimento de vagas de cargos Técnico-Administrativos. 2023. Disponível em https://arquivos.qconcursos.com/regulamento/arquivo/77533/ufrj_2023_tecnico_administrativo_em_educacao_nivel_superior_edital_n_490-edital.pdf. Acesso em: 16 out. 2024.

CENTRO DE INSTRUÇÃO E ADAPTAÇÃO DA AERONÁUTICA - Concurso Público para Oficial de Apoio da Aeronáutica - Análise de Sistemas. 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77188/ciaar_2024_oficial_de_apoio_eaoap-edital.pdf. Acesso em: 13 nov. 2024.

PROCURADORIA GERAL DO MUNICÍPIO DE NITERÓI - Edital n. 01/2023 - Concurso público para o provimento de vagas nos cargos de nível superior de analista processual, Analista contábil e analista de tecnologia da informação e para nível médio de técnico de Procuradoria da procuradoria geral do município de Niterói - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/69167/pgm_niteroi_2023_edital_n_1-edital.pdf. Acesso em: 13 nov. 2024.

ESCOLA DE ESPECIALISTAS DE AERONÁUTICA - EEAR - 2024 - Sargento - EAGS - Exame de Admissão ao Estágio de Adaptação à Graduação de Sargento da Aeronáutica de nível médio/técnico. 2024. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77026/eear_2024_sargento_eags-edital.pdf. Acesso em: 13 nov. 2024.

UNIVERSIDADE FEDERAL DA FRONTEIRA SUL - UFFS - Edital n. 59/GR/UFFS/2023. Concurso público para provimento de cargos efetivos de técnico administrativos de nível médio e superior - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77092/uffs_2023_edital_n_59-edital.pdf. Acesso em: 27 nov. 2024.

ESCOLA DE SAÚDE E FORMAÇÃO COMPLEMENTAR DO EXÉRCITO - EsFCEx. Concurso público para quadro complementar e capelães - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77515/esfcex_2024_quadro_complementar_e_capelaes-edital.pdf. Acesso em: 27 nov. 2024.

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO - UFRJ - Edital nº 491, de 29 de abril de 2023. Concurso Público para provimento de vagas de cargos Técnico-Administrativos - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77534/ufrj_2023_tecnico_administrativo_em_educacao_nivel_medio_tecnico_edital_n_491-edital.pdf. Acesso em: 27 nov. 2024.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTOS - UFES - Edital n. 46/2023 - Concurso público para os cargos técnico-administrativos em educação - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77660/ufes_2023_edital_n_42-edital.pdf. Acesso em: 27 nov. 2024.

ASSEMBLEIA LEGISLATIVA DO ESTADO DO MARANHÃO - ALEMA - Edital n. 01/2023. Concurso público para provimento de vagas e cadastro de reserva Do quadro de pessoa - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/76911/al_ma_2023_edital_n_1-edital.pdf. Acesso em 28 nov. 2024.

UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA - Edital n. 172/2023-PROGEP. Concurso público para pessoal técnico–administrativo em educação - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77323/ufma_2023_edital_n_120-edital.pdf. Acesso em 28 nov. 2024.

TRIBUNAL REGIONAL FEDERAL DA 2ª REGIÃO - Edital n. 01/2016. Concurso Público de Provas, destinado à formação de cadastro reserva para provimento de cargos do Quadro de Pessoal do Tribunal Regional Federal da 2ª Região e da Justiça Federal de Primeiro Grau das Seções Judiciárias do Rio de Janeiro e do Espírito Santo - 2016. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/11724/trf_2_regiao_2016-edital.pdf. Acesso em 20 dez. 2024.

CÂMARA MUNICIPAL DE ANÁPOLIS-GO - Edital n. 01/2023. Concurso Público para provimento dos cargos efetivos da Câmara Municipal de Anápolis – GO - 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/78492/camara_de_anapolis_go_2023_edital_n_1-edital.pdf. Acesso em 23 dez. 2024.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PARANÁ - Edital n. 160/2022. Concurso Público para provimento de cargos da Carreira de MAGISTÉRIO DO ENSINO BÁSICO, TÉCNICO E TECNOLÓGICO - 2022. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/76995/if_pr_2022_edital_n_160_professor-edital.pdf. Acesso em 26 dez. 2024.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO (IF GOIANO) - Edital n. 19/2023. Concurso Público para provimento de cargos do quadro de pessoal Técnico-Administrativo em Educação (TAE). 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77428/if_goiano_2023_tecnico_administrativ
em_educacao_edital_n_19-edital.pdf. Acesso em 30 dez. 2024.

BANCO DA AMAZÔNIA S.A. - Edital n. 02/2024. Concurso público para preenchimento de vagas e formação de cadastro em cargos de Níveis médio e superior - Técnico científico. 2024. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/80493/banco_da_amazonia_2024_tecnico_edital_n_1-edital.pdf. Acesso em 02 jan. 2025.

Prefeitura de Caraguatatuba - SP. Edital n. 03/2023. Concurso Público para provimento de vagas para diversos cargos para a Prefeitura de Caraguatatuba - Técnico em Processamento de Dados. 2023. Disponível em https://arquivos.qconcursos.com/regulamento/arquivo/78976/prefeitura_de_caraguatatuba_sp_2023_edital_n_3-edital.pdf. Acesso em 02 jan. 2025.

Prefeitura de Rio Branco - AC. Edital n. 01/2024. Concurso Público para provimento dos cargos efetivos do quadro de pessoal do Município de Rio Branco - AC. 2024. Disponível em https://arquivos.qconcursos.com/regulamento/arquivo/79315/prefeitura_de_rio_branco_ac_2024_edital_n_1-edital.pdf. Acesso em 02 jan. 2025

DEFENSORIA PÚBLICA DO ESTADO DA PARAÍBA - Edital n. 0001/2021. Concurso Público para Analista de Desenvolvimento de Sistemas. 2021. Disponível em https://arquivos.qconcursos.com/regulamento/arquivo/40263/dpe_pb_2021_edital_n_001-edital.pdf. Acesso em 02 jan. 2025.

PREFEITURA MUNICIPAL DE SÃO GONÇALO (RJ) - Edital n. 02/2016. Concurso Público para Analista na Área Tecnológica. 2016. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/8108/prefeitura_de_sao_goncalo_rj_2016_edital_n_02-edital.pdf. Acesso em 06 jan. 2025.

UNIVERSIDADE FEDERAL DE PERNAMBUCO - Edital n. 10/2023. Concurso Público para provimento de cargos técnico-administrativos. 2023. Disponível em: https://

arquivos.qconcursos.com/regulamento/arquivo/77851/ufpe_2023_tecnico_administrativo_em_educacao_edital_n_10-edital.pdf. Acesso em 06 jan. 2025.

PREFEITURA MUNICIPAL DE JARU (RO) - Edital n. 001/2023. Concurso Público para provimento de cargos e cadastro reserva para seu quadro de pessoal, 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/79033/prefeitura_de_jaru_ro_2023_edital_n_1-edital.pdf. Acesso em 07 jan. 2025.

UNIVERSIDADE FEDERAL DO PERNAMBUCO (UFPE) - Edital n. 10/2023. Concurso Público para provimento de cargos técnico-administrativos. 2023. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/77851/ufpe_2023_tecnico_administrativo_em_educacao_edital_n_10-edital.pdf. Acesso em 07 jan. 2025.

EMPRESA BRASILEIRA DE CORREIOS E TELÉGRAFOS - Edital n. 271/2024. Concurso Público, para o provimento de vagas, sob o regime celetista, em cargos, do quadro de pessoal da Empresa Brasileira de Correios e Telégrafos. 2024. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/79364/correios_2024_analista_edital_n_271-edital.pdf. Acesso em 07 jan. 2025.

EMPRESA DE TECNOLOGIA E INFORMAÇÕES DA PREVIDÊNCIA – DATAPREV S/A - Edital n. 1/2024. Concurso público para provimento de vagas e formação de cadastro de reserva para cargos do quadro de pessoal da Empresa de Tecnologia e Informações da Previdência. 2024. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/79983/dataprev_2024_edital_n_1-edital.pdf. Acesso em 14 jan. 2025.

COMPANHIA DE ÁGUA E ESGOTOS DA PARAÍBA (CAGEPA) - Edital n. 1/2024. Concurso público para o provimento de vagas em cargos de nível superior e de nível médio técnico. 2024. Disponível em: https://arquivos.qconcursos.com/regulamento/arquivo/78089/cagepa_pb_2024_edital_n_1-edital.pdf. Acesso em 14 jan. 2025.