Pesquisa de Programação Orientada a Objetos

1. **Polimorfismo**
   a. **Defina o conceito de polimorfismo em POO;**

   Definimos **Polimorfismo** como um princípio 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 com valores diferentes para cada uma das classes derivadas. Resumindo, o polimorfismo refere-se à capacidade de um mesmo método ou função ser aplicado de diferentes maneiras em classes derivadas, permitindo que uma classe derivada possa ser criada de forma que pode usar o mesmo método, porém com valores distintos.

   b. Explique os dois tipos principais de polimorfismo:

   i. Polimorfismo de sobrecarga (overloading);

   A sobrecarga de métodos ocorre quando criamos métodos com o mesmo nome na mesma classe, mas com diferentes listas e números de parâmetros. A sobrecarga é resolvida em tempo de compilação, pois o compilador escolhe qual método chamar com base no número e tipo dos argumentos passados.

   ii. Polimorfismo de sobrescrita (overriding).

   A sobrescrita ocorre quando uma subclasse redefine um método com a mesma assinatura da superclasse com o "overriding", permitindo que o método na subclasse tenha um comportamento específico. A sobrescrita é resolvida em tempo de execução, permitindo que o método da subclasse seja chamado mesmo quando o objeto é referenciado pela superclasse.

c. Elabore um exemplo prático e lúdico, com classes distintas, de como o polimorfismo pode ser implementado em Java;

Temos uma classe base que se chama (CalcularImpostos) e temos um método (**CalcularIpvaPorEstado)** quando for criar uma Classe derivada "RioGrandeDoNorte" ou "Paraiba" esse método vai ser utilizado, mas com argumentos diferentes.

d. Faça o digrama de classe do exemplo elaborado.

Referências:

https://www.devmedia.com.br/conceitos-e-exemplos-polimorfismo-programacao-orientada-a-objetos/18701,

https://desenvolvimento.shift.com.br/polimorfismo-o-terceiro-pilar-da-orienta%C3%A7%C3%A3o-a-objetos-f89af473f726,

https://comoprogramarjava.com.br/polimorfismo-em-java-metodos-sobrecarregados-e-sobrescritos/ .

2. Classes abstratas

a) Explique o conceito de classe abstrata;

Na programação orientada a objetos (POO), uma classe abstrata é um conceito que atua como modelo para outras classes. Ela não pode ser instanciada diretamente, o que significa que não é possível criar objetos a partir dela. Seu principal objetivo é oferecer uma estrutura básica que será herdada por outras classes.

b) Defina as características e regras para a criação de classes abstratas;

As classes abstratas são as que não permitem realizar qualquer tipo de instância. São classes feitas especialmente para serem modelos para suas classes derivadas. As classes derivadas, via de regra, deverão sobrescrever os métodos para realizar a implementação dos mesmos. As classes derivadas das classes abstratas são conhecidas como classes concretas.

Como medida de segurança, as classes abstratas somente podem ser estendidas, sendo que a criação de um objeto a partir da mesma é um procedimento evitado. Além disso, caso um ou mais métodos abstratos estejam presentes nessa classe abstrata, a classe filha será, então, forçada a definir tais métodos, pois, caso contrário, a classe filha também se tornará abstrata.

A funcionalidade dos métodos abstratos que são herdados pelas classes filha normalmente é atribuída de acordo com o objetivo ou o propósito dessas classes. É possível, porém, não atribuirmos uma funcionalidade a esses métodos abstratos. Neste caso, faz-se necessário, pelo menos, declarar tais métodos.

c) **Explique a diferença entre uma classe abstrata e uma classe concreta é a seguinte:**

1. Classe Abstrata:

- Não pode ser instanciada diretamente.
- Pode conter métodos abstratos (sem implementação) e métodos concretos (com implementação).
- Serve como um modelo para outras classes, permitindo que subclasses implementem os métodos abstratos.

2. Classe Concreta:

- Pode ser instanciada.
- Contém implementações completas de todos os seus métodos.
- É utilizada para criar objetos que podem ser utilizados diretamente.

Resumindo: a classe abstrata é um esboço que não pode ser usado sozinha, enquanto a classe concreta é uma implementação completa que pode ser utilizada para criar objetos.

d) **Elabore um exemplo prático e lúdico, com classes distintas, do uso de classes abstratas em POO;**
e) **Faça o diagrama de classe do exemplo elaborado.**

Referências:

https://www.devmedia.com.br/classes-abstratas-e-interfaces-em-java/23145#:~:text=Classes%20abstratas%20s%C3%A3o%20aquelas%20que,especializa%C3%A7%C3%A3o%20elas%20podem%20ser%20utilizadas, https://www.devmedia.com.br/conceitos-classes-abstratas-programacao-orientada-a-objetos/18812.

3. Interfaces

a) - Defina o que é uma interface em POO;

Uma **interface** em **Programação Orientada a Objetos** é uma classe como um contrato, uma "obrigação", que contém métodos que DEVEM ser contidas em outras classes que forem implementadas pela interface definida, relacionando-se e utilizando três dos 4 pilares do paradigma: **abstração, herança e polimorfismo**. Porém, não se é utilizado a herança em seu sentido "tradicional", que é **herdar implementação** (atributos, construtores, métodos). Mas, sim, **implementação**. Sendo assim, as classes que irão "herdar" os métodos de contrato, terão que definir, logo após o nome da classe "implements nomeDaInterface".

A interface se assemelha grandemente a uma ***classe abstrata***, mas que **não pode** atribuir e declarar outras propriedades (atributos que não sejam constantes, construtores, etc.), **se não**, métodos e atributos constantes (também não se pode instanciar objetos, sendo outra semelhança). Ou seja, a interface "'diz' o que se deve fazer".

b) - Explique como as interfaces ajudam a estabelecer contratos entre classes;

Elas ajudam a manter uma **comum implementação**, por meio de métodos que **DEVEM** ser implementados nas classes que estiverem sendo implementadas pela interface declara e/ou escolhida, como regras em um contrato, tornando a manutenção e gerenciamento mais prática, flexível e fácil.

c) - Diferencie uma interface de uma classe abstrata;

Numa **classe abstrata**, a(s) classe(s) filha(s) que herda(m) desta classe (super), herda(m) todos os *atributos*, *construtores*, *métodos (concretos e abstratos)* e tudo o que haver na classe. Utilizada para que herde para outras classes que contém comportamentos em comum ou semelhantes.

Já na **interface**, é implementado na classe que implementa essa interface o contrato de que se **DEVE** haver X comportamentos, que serão pelos **métodos**, garantido um comportamento comum. Geralmente contém apenas ***métodos abstratos***, em versões mais antigas, na maioria das linguagens de programação. Mas, pode haver métodos *default* (que permite ser utilizado métodos sem obrigação de sobrescrita) e *static* (agrupa funcionalidades que auxiliam, por métodos, que podem ser chamadas diretamente pela interface, sem haver necessidade de ser por um objeto), em versões mais recentes.

A semelhança é que ambas podem conter métodos abstratos, que obrigam as classes que herdam ou implementem, fornecer uma implementação.

A presença do polimorfismo é mostrada, no **exemplo (*classe Animal*)**, ao ser utilizados dois métodos com a mesma assinatura (nome) – (**emitirSom** e **seMover**), em classes diferentes, mas resultando em saídas diferentes, de textos (e o que for presente e utilizado), por exemplo, a depender do objeto instanciado (se é do Cachorro, Galinha, etc).

d) - Elabore um exemplo que demonstra como implementar múltiplas interfaces em uma classe concreta e abstrata;

e) Crie o diagrama de classe do exemplo.

Referências:

https://www.dio.me/articles/poo-em-java-compreendendo-as-interfaces,
https://www.youtube.com/watch?v=y0nxLH2Ta2Y,
https://www.youtube.com/watch?v=2bZl62vNU0A,
https://www.devmedia.com.br/interfaces-programacao-orientada-a-objetos/18695.

4. Comparação entre Classes Abstratas e Interfaces

a) Faça um quadro comparativo entre classes abstratas e interfaces;

Característica	Classes abstratas	Interface
Tipo de métodos	Contém ou pode conter **métodos concretos** (com implementação); Contém ou pode conter **métodos abstratos.**	Contém ou pode conter métodos concretos (neste caso, métodos ***static*** ou ***default***, com implementação); Contém ou pode conter **métodos abstratos.**
Herança	Herda implementação, com extends (atributos, métodos, etc); Uma classe pode herdar de apenas uma classe abstrata.	Implementa funções, com implements (define ações nos métodos); Uma classe pode implementar diversas e múltiplas interfaces.
Construtores	Pode conter construtor(es);	Não pode conter construtor(es);
Atributos	Pode conter atributos;	Não pode conter atributos, a não ser que sejam constantes;
Flexibilidade	Desempenha uma flexibilidade, praticidade e abstração, pois herda	Mantém um comportamento padrão (contrato) com as classes que a implementam;

	implementação e, também, pode estabelecer contratos.	
Uso Principal	É feita para servir como modelo para outras classes;	É feita para manter uma comum implementação (contrato);

b) Quando utilizar uma classe abstrata e quando utilizar uma interface?

Utiliza-se **classe abstrata** quando há uma comum e semelhança nas classes. Ela servirá como uma classe base, de referência, modelo, que irá conter propriedades e definições análogas ou semelhantes, que as classes filhas irão herdar.

Utiliza-se **interface** quando se quer manter um comum e padronizado comportamento, fazendo como que um contrato para que as classes que a implementem mantenham uma funcionalidade igual, mas com valores diferentes, se tornando mais flexível para mudanças e gerenciamento.

c) Quais são as vantagens e desvantagens de cada uma?

	Classes abstratas	**Interface**
Vantagens	- Mais abrangente, prática e reutilizável; - Pode conter quaisquer atributos, métodos e propriedades;	- Padronização, comum ações, com saídas diferentes; - Cria um contrato de comportamento (com métodos); - Melhor manutenção, gerenciamento e alterações.
Desvantagens	- Não se pode instanciar objetos desta classe; - Uma classe só pode herdar de uma classe abstrata; - Menos flexível em herdar e mais flexível que interface.	- Sem implementação comum; - Não pode conter construtores.

Referências:

https://cursos.alura.com.br/forum/topico-diferenca-entre-classe-abstrata-e-interface-144776,
https://www.devmedia.com.br/interfaces-x-classes-abstratas/13337,
https://www.dio.me/articles/interfaces-x-classes-abstratas.

5. Conclusão

a) Resuma os principais pontos abordados no trabalho;

Polimorfismo: é quando um mesmo objeto pode agir de jeitos diferentes, dependendo da situação. Existem dois tipos principais:

Polimorfismo de sobrecarga: quando métodos têm o mesmo nome, mas parâmetros diferentes.

Polimorfismo de sobrescrita: quando uma subclasse muda o comportamento de um método da superclasse.

Classe abstrata: funciona como um molde que serve de base para outras classes, mas que não pode ser usada diretamente (não é possível instanciar objetos).

Interface: é como um acordo que diz quais métodos uma classe precisa ter, sem dizer como eles devem funcionar. Mas, que a funcionalidade é implementada na classe que implementa a interface.

b) Explique como o entendimento de polimorfismo, classes abstratas e interfaces contribui para a criação de sistemas orientados a objetos mais flexíveis e reutilizáveis.

Compreender e obter o entendimento de polimorfismo, classes abstratas e interfaces contribui para um sistema mais organizado, fluido, flexível, etc.

Polimorfismo:

O polimorfismo permite que objetos distintos respondam de maneiras diferentes a um mesmo comando.

Classes abstratas:

A classe abstrata estabelece uma base compartilhada para outras classes, prevenindo a necessidade de duplicação de código.

Interface:

A interface, que funciona como um contrato, torna um sistema mais estável, prático e organizado que ajuda a manter uma comum implementação.

Em conjunto, esses conceitos permitem uma ampliação e gerenciamento muito mais eficaz, livre e permite uma melhor futura alteração.