NOME:
DISCIPLINA: Programação Orientada a Objetos
CURSO: Ciência da Computação
DATA: 04/05/2023
PROFESSOR: Felipe Roncalli de Paula Carneiro

Padrão MVC

1. **O que é o padrão arquitetural MVC e como ele funciona em aplicações Java?**
   É uma arquitetura de código, que é introduzida em algum projeto com a função de separar as funcionalidades em 3 camadas. Em java este padrão é utilizado em classes de controle, classes de interface e classes de domínio, onde é feito um mapeamento de todo o código para suportar todas as regras e suas operações.

2. **Quais são as principais vantagens de se utilizar o padrão MVC em aplicações Java?**
   A principal vantagem em utilizar o padrão MVC, é separar toda a lógica de negócio, separar a parte visual da interface, e por fim trabalhar todo esse fluxo da aplicação.

3. **Como o padrão MVC ajuda a separar as preocupações (separation of concerns) em aplicações Java?**
   Como o MVC consegue separar toda esse regra de negócio dentro de uma aplicação, as partes de um programa, logo, ficam totalmente separadas, onde que cada parte fica responsável por lidar com a regra de negócio estabelecida pela arquitetura MVC.

4. **Qual é a responsabilidade do modelo (Model) em aplicações Java baseadas no padrão MVC?**
   O model é responsável por intermediar e controlar como os dados se comportam por meio de suas funções, logica e regra de negocios estabelecidas.

5. **Qual é a responsabilidade da visão (View) em aplicações Java baseadas no padrão MVC?**
   A camada da visão fica responsável por apresentar ao usuário todas as informações necessarias da aplicação. Na view, é onde vai estar as caixas de textos, label, botões, entre outros componentes.

6. **Qual é a responsabilidade do controlador (Controller) em aplicações Java baseadas no padrão MVC?**
   O controller é um intermediador sobre as requisições enviadas da view e aplicar as mesmas sobre as regras do controle. Onde por fim, vai processar todos os dados que fui repassado pelas outras camadas.

7. **Como é possível implementar a camada de modelo em aplicações Java baseadas no padrão MVC?**
   O model é implementado em java, como uma classe para controlar os dados e como eles terão seu comportamente durante o código.

8. **Como é possível implementar a camada de visão em aplicações Java baseadas no padrão MVC?**
   O modelo da View é aplicada usando bibliotecas de interface (Swing por exemplo), onde vai ser criando a visualização onde o usuário vai interagir com o programa.

9. **Como é possível implementar a camada de controlador em aplicações Java baseadas no padrão MVC?**
   O modelo controller implementa os dados fornecidos pela View, transoforma as informações para serem aceitas pelo model e também verifica as condições se estes mesmos dados estão corretos.

10. **Como o modelo de dados é implementado em aplicações Java que seguem o padrão MVC?**
    Os dados que forem informados nos campos da tela da view, serão armazenadas e logo em seguida será direcionada para a camada de controller, onde vai existir a transformação desses dados para que eles possam ser aceitas pela camada de Model.

11. **Como o padrão MVC ajuda a separar a lógica de negócio da lógica de apresentação em aplicações Java?**
    Em aplicações java, na camda view fica responsável por tratar a logica de negócio, enquanto na camada view, fica responsável por tratar a lógica de apresentação.

12. **Como o padrão MVC pode ser usado em conjunto com outras tecnologias, como HTML, CSS, JavaScript e bancos de dados, para criar aplicações Java completas e robustas?**
    Na camada View: Encontra-se as tecnologias visuais, como HTML, CSS e javascript.
    Na camada Controller: Encontra a tecnologia javascript que pode ser ultilizada ainda com algum outro framework.
    No model: Onde ficaria o javascript e o banco de dados.

13. Qual das seguintes opções representa corretamente as responsabilidades do componente Controller em uma arquitetura MVC em Java?
    a) Gerenciar o acesso ao banco de dados e realizar consultas de dados.
    b) Controlar a interação entre o modelo e a visão.
    c) Definir a aparência e o comportamento da interface do usuário.
    d) Validar e processar os dados de entrada fornecidos pelo usuário.
14. Qual das seguintes opções representa corretamente as responsabilidades do componente View em uma arquitetura MVC em Java?
    a) Realizar cálculos e manipulações de dados complexas.
    b) Controlar a interação entre o modelo e o usuário.
    c) Definir a aparência e o comportamento da interface do usuário.
    d) Gerenciar o fluxo de controle da aplicação.

15. **Qual das seguintes opções representa corretamente as responsabilidades do componente Model em uma arquitetura MVC em Java?**
    a) Controlar a interação entre o usuário e a aplicação.
    b) Definir a aparência e o comportamento da interface do usuário.
    c) Realizar cálculos e manipulações de dados complexas.
    d) Gerenciar o fluxo de controle da aplicação.

16. **Qual das seguintes opções representa corretamente a sequência de passos no padrão arquitetural MVC em Java?**
    a) O usuário interage com a View, que envia uma solicitação ao Model, que processa a solicitação e envia uma resposta para a Controller, que atualiza a View.
    b) O usuário interage com a Controller, que envia uma solicitação ao Model, que processa a solicitação e envia uma resposta para a View, que atualiza a Controller.
    c) O usuário interage com a Model, que envia uma solicitação para a Controller, que processa a solicitação e envia uma resposta para a View, que atualiza a Model.
    d) O usuário interage com a View, que envia uma solicitação para a Controller, que processa a solicitação e envia uma resposta para a Model, que atualiza a View.

17. **Qual das seguintes opções representa corretamente as principais vantagens de se utilizar o padrão arquitetural MVC em Java?**
    a) Permite uma fácil integração com bancos de dados e outras tecnologias.
    b) Promove uma maior separação de preocupações e uma melhor organização do código.
    c) Aumenta a eficiência da aplicação ao evitar a duplicação de código.
    d) Simplifica o processo de desenvolvimento ao fornecer um conjunto padrão de classes e interfaces.

Interfaces

1. **O que é uma interface em Java?**
    a) Um tipo de dado que pode armazenar múltiplos valores.
    b) Um tipo de classe que contém apenas métodos abstratos e constantes.
    c) Um tipo de classe que pode ser instanciado diretamente.
    d) Um tipo de classe que só pode ser estendido por outras classes.
2. Quais são as principais diferenças entre uma classe abstrata e uma interface em Java?
    a) Uma classe abstrata pode ser instanciada, enquanto uma interface não pode.
    b) Uma classe abstrata pode conter métodos concretos, enquanto uma interface não pode.
    c) Uma classe abstrata pode implementar interfaces, enquanto uma interface não pode.
    d) Uma classe abstrata pode ter construtores, enquanto uma interface não pode.

3. Como é possível implementar uma interface em uma classe em Java?
   a) Utilizando a palavra-chave "extends".
   b) Utilizando a palavra-chave "implements".
   c) Utilizando a palavra-chave "implements" seguida da palavra-chave "extends".
   d) Utilizando a palavra-chave "interface" seguida do nome da classe.
4. Quais são os benefícios de se usar interfaces em Java?
   a) Permite a reutilização de código através da herança.
   b) Facilita a implementação de múltiplas interfaces em uma mesma classe.
   c) Ajuda a garantir a consistência e a modularidade do código.
   d) Todas as opções acima estão corretas.
5. Qual é a finalidade do método default em uma interface em Java?
   a) Definir um método concreto que pode ser sobrescrito pelas classes que implementam a interface.
   b) Definir um método que não precisa ser implementado pelas classes que implementam a interface.
   c) Definir um método que pode ser acessado apenas dentro da interface.
   d) Definir um método que pode ser acessado apenas por classes que estendem a interface.
6. O que é uma interface funcional em Java?
   a) Uma interface que contém apenas um método abstrato e é usada para implementar programação funcional em Java.
   b) Uma interface que contém apenas métodos concretos e é usada para implementar programação orientada a objetos em Java.
   c) Uma interface que contém apenas constantes e é usada para armazenar valores imutáveis.
   d) Uma interface que contém apenas métodos abstratos e é usada para armazenar valores mutáveis.

7. **O que é uma interface em Java e para que ela é usada?**
   Basicamente, a interface obriga a classes diferentes a possuirem metodos iguais.

8. **Quais são as principais diferenças entre uma classe abstrata e uma interface em Java?**
   A classe abstrata não necessariamente, vai ser acionada diretamente no código, uma vez que ela só vai ser usada em uma outra classe de heranca. Diferente de uma interface, que é acionada diretamente quando chamada no código.

9. **Como é possível implementar uma interface em uma classe em Java e quais são as regras para isso?**
   Em java, voce precisar criar uma interface, onde ela vai ter os metodos iguais que serão ultilizados nas outras classes. Após isso na classe que vai se ultilizar uma interface, utiliza-se o implements para que possa ser aplicada a interface criada.

10. **É possível criar uma instância de uma interface em Java? Explique sua resposta.**
    Não é possivel criar uma instancia diretamente na interface, somente pela classe que implementa a interface.

11. **Como é possível estender uma interface em Java e quais são as implicações dessa extensão?**
    É possível estender uma interface, criando uma nova interface filho que vai estender na interface pai.

12. Como é possível criar uma hierarquia de interfaces em Java e como essa hierarquia pode ser usada para melhorar a reutilização de código?
    É possível criar uma hierarquia de do mesmo modo que se cria em classes e assim se adaptar para implementar outras interfaces.

13. Considere o seguinte trecho de código

```
public interface Animal {
    public void makeSound();
}

public class Cat implements Animal {
    public void makeSound() {
        System.out.println("Meow");
    }
}

public class Dog implements Animal {
    public void makeSound() {
        System.out.println("Woof");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal cat = new Cat();
        Animal dog = new Dog();
        cat.makeSound();
        dog.makeSound();
    }
```

Qual é o objetivo da interface Animal neste trecho de código?

a) Fornecer uma implementação padrão para o método makeSound().
b) Definir um tipo de objeto que pode ser utilizado para criar instâncias de Cat e Dog.
c) Especificar um contrato que as classes Cat e Dog devem seguir ao implementar o método makeSound().
d) Permitir que a classe Main crie instâncias de Animal diretamente.