1) Assinale V ou F:

a) (F) É mais fácil implementar código reutilizável, do que código não-reutilizável.

R: um código reutilizável demora até que fique um código bom para ser reutilizado

b) (v) Um framework é um artefato mais amplo do que uma biblioteca.

c) (v) É possível somar uma casa e uma banana, se esses itens forem objetos que possuam métodos que compreendam uma mensagem de acionamento de método chamada "soma"

d) (f) Não é possível criar um vetor de elementos diferentes, mesmo que eles possuam relações de herança entre si (por exemplo, são classes "irmãs").

e) (f) Não é possível criar uma lista de elementos diferentes.

f) (v) Classes diferentes podem possuir nomes de métodos iguais.

2)Suponha que você possui uma função chamada SomarMatrizes, que recebe duas matrizes e retorna a soma entre elas.

Você deseja converter essa função para uma versão de programa orientada a objetos.

Considere que haverá uma classe chamada Matriz e que o método "somar" receberá como parâmetro um objeto do tipo Matriz. Esse método retornará a soma das matrizes. Considerando esse novo código orientado a objetos, forneça:

a) a assinatura de método "somar";

public Matriz SomarMatrizes(Matriz m1, Matriz 2){};

b) um exemplo de código que cria duas matrizes e armazena a matriz resultante da soma em outro objeto.

R: código feito em java.

```
public Matriz(int[][] elemento) {
   this.elemento = elemento;
}
```

```
public Matriz soma(Matriz m2){
   if(this.getElemento().length != m2.getElemento().length ){
       throw new IllegalArgumentException("dimensão diferente");
   }
   Matriz resultado=new Matriz(getElemento());
   for (int i = 0; i < this.getElemento().length; i++) {
       for (int j = 0; j < this.getElemento().length; j++) {
           resultado.getElemento()[i][j]= this.getElemento()[i][j]+m2.getElemento()[i][j];
       }
   }
   return resultado;
}
```

int

3) Classes abstratas podem possuir métodos concretos?
R: Sim, consigo implementar métodos em classes abstratas.

4) Interfaces podem possuir métodos concretos?

R:Não consigo implementar métodos em interfaces, apenas criar as assinaturas dos métodos e obrigar as classes que implementam a interface a implementar o método.

5) Informe se os arquivos de persistência XML e JSON a seguir são idênticos em estrutura.

https://gitlab.com/hvescovi/prog24/-/tree/main/oo/java/patterns/03-adapter/MonitoringMaven_xml_json/bdxml

https://gitlab.com/hvescovi/prog24/-/tree/main/oo/java/patterns/03-adapter/MonitoringMaven_xml_json/bdjson

R: Sim.

NÃO FIZ

6) Porque a classe anotada para XML continua funcionando quando é utilizada persistência em JSON?

https://gitlab.com/hvescovi/prog24/-/blob/main/oo/java/patterns/03-adapter/MonitoringMaven_xml_json/src/main/java/model/Produto.java

7) Implemente uma maneira de garantir que os dois DAOs e novos DAOs que surgirem possuam os mesmos métodos, de maneira que um cliente possa "trocar" o DAO que utiliza sem precisar alterar as chamadas feitas ao DAO.

https://gitlab.com/hvescovi/prog24/-/tree/main/oo/java/patterns/03-adapter/MonitoringMaven_xml_json/src/main/java/dao

8) Qual a diferença entre métodos concretos e métodos abstratos?

R: Métodos concretos há a implementação, já métodos abstratos colocamos apenas a assinatura.

9) Para implementar persistência em uma ou mais classes pode-se fazer uso de mapeadores objeto-relacional (ORM: Object-Relational Mapping). Forneça um exemplo de duas classes relacionadas persistentes.

O Hibernate **é um framework de mapeamento objeto-relacional** (ORM) que **facilita o mapeamento de classes Java para tabelas de banco de dados** relacionais e vice-versa. Ele fornece uma camada de abstração entre o código orientado a objetos e o banco de dados, **permitindo que os desenvolvedores trabalhem com objetos Java em vez de escrever código SQL diretamente**. O Hibernate é responsável pela persistência de dados, gerenciamento de sessões, cache, consultas, entre outras funcionalidades relacionadas à persistência. a anotação @entity @column pertence ao hibernete e não ao springboot

A persistência é essencialmente a capacidade de salvar (persistir) os dados de uma entidade ou objeto em um meio de armazenamento duradouro, como um banco de dados,

Faça o código livro e autor onde um autor tenha vários livros coloque as anotações e ja era

10) Considere o caso das classes Pessoa, Motorista e Vendedor. Para fazer o reaproveitamento de informações, é melhor utilizar composição ou herança? Justifique.

R: composição, pois uma pessoa pode ser um motorista um vendedor um programador, ou seja se eu colocar como herança terá a repetição de dados

exemplo:

A classe pessoa tem nome e cpf como atributos, agora as classes motorista vendedor herdam da classe pessoa, nesse contexto observe que tem a repetição dos dados nome e cpf na classe vendedor e motorista programador etc.

como é composição consigo fazer com que a o motorista, vendedor são a pessoa X

.**Hylson:** Se usar herança, fica assim:

m1 = new Motorista(“João da Silva”, “AKE123”)
v1 = new Vendedor(“João da Silva”, 0.05)
A pessoa vai ser REPETIDA. É melhor, portanto, usar compsição.

11) Considere o caso das classes Veículo, Carro e Moto. Para fazer o reaproveitamento de informações, é melhor utilizar composição ou herança? Justifique.

R: Herança, consigo fazer um aproveitamento melhor das informações, fazendo com que as classes herdam os atributos das classes pai.

12) Considere uma classe chamada Veículo, que possui dentre outros métodos um método chamado "toString". Esse método retorna os valores dos atributos em versão textual. Existe também uma classe chamada Carro, que é uma classe "filha" da classe Veiculo. Ao confeccionar o método "toString" da classe Carro, é possível reaproveitar o método toString da classe pai. Como isso pode ser feito? Forneça um exemplo.

```
@Override
public String toString() {
//super da classe produto
   return super.toString()+ "UsedProduct{" +
           "manufactureDate=" + manufactureDate +
           '}';
}
```

13) É possível instanciar objetos a partir de classes abstratas? Se sim, forneça um exemplo. Se não, explique a utilidade de uma classe abstrata.

R:Não, a classe abstrata serve para ser algo mais genérico de algo mais específico, com isso conseguimos ter outras classes específicas dessa classe generica.

14) Classes concretas podem possuir métodos abstratos?

R: Ao criar um método abstrato, em java gera um erro informando que a classe deve ser abstrata.

15) Quando ocorre herança a partir de classes concretas, os atributos também são herdados?

R: Sim.

16) Se houver alguma opção abaixo que não seja uma forma de reuso de software, aponte e explique:

a) Desenvolver sistemas a partir do zero

b) Fazer a manutenção de sistemas: requer atualizar o código, com isso posso usar um método que já tem para realizar o trecho que eu preciso.

c) Corrigir erros em sistemas: posso ter um método que faça a correção/tratamento do erro em outra parte do codigo e so utilizar metodo

d) Adaptar o sistema a um novo hardware

e) Melhorar o desempenho de um sistema

f) Ampliar a funcionalidade de um sistema

17) Analise a seguinte frase:

"Espera-se que programadores gastem tempo tanto quanto em ler código antigo e ver como reusá-lo, assim como para criar novos códigos".

Comente algo sobre essa afirmação.

R: Concordo com a afirmação, acho essencial estudar/ler um código antigo para saber o que o código faz onde eu posso usar, falando de outra forma estudar a ferramenta para utilizar em códigos futuros. E a importância de gastar tempo na criação também é muito essencial, pois se esse código for bem feito quando ele virar um código “antigo” ou outro programador for visualizar facilitará o entendimento, esses são temas importantes abordados do livro “codigo”.

FIM