Java OO: Entendendo a Orientação a Objetos

⭐ Questão 1 (Orientação a Objetos)

Com base no que você ouviu no vídeo, selecione a alternativa que expressa a ideia central do paradigma da Orientação a Objetos.

Resposta: Dado e funcionalidade sobre ele andam juntos.

⭐ Questão 2 (Orientação a Objetos)

De acordo com as situações citadas no vídeo, escolha a única alternativa que não é sinal de programação procedural.

Resposta: Várias equipes trabalhando em um único projeto.

> **Comentário da Alura:** Para que várias equipes consigam trabalhar em um mesmo projeto, é necessário que as responsabilidades de cada código estejam bem definidas e claras, evitando conflitos na hora de realizar mudanças e evoluções. Código com responsabilidades coesas é sinal do paradigma OO.

⭐ Questão 3 (Orientação a Objetos)

Fulano é novo na equipe da empresa KWZ. Ficou responsável por uma tela de cadastro, onde existe um campo que deve ser validado conforme a regra de negócio XYZ. Ele pergunta a seu gerente onde pode obter informações sobre a tal regra. Marque as alternativas que podem ser identificadas como respostas de quem usa o paradigma procedural em seus projetos.

Resposta:

- Dependendo do módulo, Fulano deve perguntar ao responsável técnico do mesmo.
- "Copie o código de validação que está no formulário ABCD."
- Fulano deve entrar em contato com alguém da equipe de analistas de negócio da empresa, para ele entender a regra e implementá-la.

⭐ Questão 4 (Orientação a Objetos)

Como chamamos, em orientação a objetos, as características de uma classe?

Resposta: Atributos.

⭐ Questão 5 (Orientação a Objetos)

A partir do conhecimento adquirido de classes, leia as frases abaixo e responda a opção correta.

- Uma classe é uma especificação de um tipo, definindo atributos e comportamentos.
- Um objeto é uma instância de uma classe onde podemos definir valores para seus atributos.
- Para criar uma instância é obrigatório preencher os valores de todos os atributos.
- Para criar uma instância precisamos usar a palavra chave new

Resposta: A terceira alegação é a única errada.

⭐ Questão 6 (Orientação a Objetos + Java)

Jonas criou uma classe do tipo Pessoa para representar um personagem de um jogo que está criando. Observe a classe que ele criou:

```java
public class Pessoa {
    String nome;
    int idade;
    int peso;
}
```

Qual das opções abaixo é a correta para criar um objeto e definir um valor para seus atributos?

Resposta:

```java
Pessoa heroi = new Pessoa();
heroi.nome = "Jonny";
```

⭐ Questão 7 (Orientação a Objetos + Java)

Usando o aprendizado sobre referências e atribuição de valores vamos definir uma classe abaixo.

```java
public class Conta {
    double saldo;
}
```

A partir desta classe, diga o que imprime o código:

```java
public class Teste {
    public static void main(String [] args) {
        Conta minhaConta = new Conta();
        minhaConta.saldo = 500.0;
        Conta outraConta = minhaConta;
        outraConta.saldo += 1000.0;
        System.out.println(minhaConta.saldo);
    }
}
```

Resposta: Imprime 1500.0 pois outraConta recebeu a referência de minhaConta. Assim, o incremento de 1000.0 também afetou minhaConta.

⭐ Questão 8 (Java)

O que aprendemos sobre métodos?

Respostas:

- Um método define o comportamento ou a maneira de fazer algo.
- É possível que um método não tenha nenhum parâmetro.
- Por convenção, o nome do método no mundo Java deve começar com letra minúscula.

⭐ Questão 9 (Java)

Qual é a sintaxe e ordem correta para chamar um método com Java?

Resposta:

```
nomeDaReferencia.nomeDoMetodo();
```

⭐ Questão 10 (Java)

Todas as afirmações abaixo são verdadeiras exceto:

Resposta: O uso do this é obrigatório.

⭐ Questão 11 (Java)

Assumindo que cada método abaixo está dentro de uma classe, quais declarações estão válidas (compilam)?

Respostas:

```
void deposita(double valor, int numero){

}

void deposita(){

}
```

⭐ Questão 12 (Java)

A Ana está praticando OO com Java e criou uma outra classe Conta com apenas dois atributos e um método. No entanto, como o uso do this é opcional, ela está com dúvida onde se usa a palavra chave this dentro de uma classe. Ela nos enviou o código abaixo usando os caracteres [] em vários lugares, segue o código:

```
class Conta {

        [1] double saldo;
        int numero;

        void deposita([2] double valor) {
            [3]saldo = [4]saldo + [5]valor;
        }
}
```

Repare que temos [1], [2], [3], [4] e [5] como possíveis lugares para colocar o this, mas qual realmente irá funcionar e ser compilado?

Resposta: Apenas [3] e [4].

⭐ Questão 13 (Java)

O Pedro escreveu o método saca abaixo que não está compilando:

```
//assumindo que esse método está dentro da classe Conta que possui os atributos
public void saca(double valor) {

    if(saldo >= valor) {
        saldo -= valor;
        return true;
    } else {
        return false;
    }
}
```

O que está errado com o método?

Resposta: Faltou definir o tipo correto do retorno no método (nesse caso, boolean).

⭐ Questão 14 (Java)

Juarez criou as seguintes classes:

```java
public class Pessoa {

    String nome;
    String cpf;
    int idade;
    String logradouro;
    String complemento;
    String numero;
    String bairro;
    String cidade;
    String cep;

}

public class Empresa {

    String razaoSocial;
    String cnpj;
    String logradouro;
    String complemento;
    String numero;
    String bairro;
    String cidade;
    String cep;
}
```

Podemos perceber que os atributos logradouro, complemento, numero, bairro, cidade e cep são os mesmos nas duas classes. Essas informações são fortes candidatas para serem externalizadas na classe Endereco e associadas às classes Pessoa e Empresa através de composição.

```java
public class Endereco {

    String logradouro;
    String complemento;
    String numero;
    String bairro;
    String cidade;
    String cep;

}
```

Marque a única alternativa verdadeira que modifica corretamente as classes Pessoa e Empresa para utilizarem a classe Endereco.

Resposta:

```java
public class Pessoa {

    String nome;
    String cpf;
    int idade;
    Endereco endereco;
}

public class Empresa {

    String razaoSocial;
    String cnpjs;
    Endereco endereco;
}
```

⭐ Questão 15 (Java)

Fernanda decidiu aplicar seu conhecimento sobre composição utilizando as classes Pessoa e Endereço criadas por Juarez. Porém, seu código não esta funcionando em tempo de execução (runtime). Vejamos seu código:

```
public class Pessoa {

    String nome;
    String cpf;
    int idade;
    Endereco endereco;
}

public class Endereco {

    String logradouro;
    String complemento;
    String numero;
    String bairro;
    String cidade;
    String cep;

}

public class Programa {

    public static void main(String args[]) {

        Pessoa p = new Pessoa();
        p.nome = "Paulo";
        p.endereco.logradouro  = "Avenida XYZ";
    }
}
```

Marque a opção que explica corretamente o erro no código de Fernanda.

Resposta: Ela esta acessando uma propriedade de um objeto que não foi inicializado.

> **Comentário da Alura:** Como a classe Pessoa não instanciou internamente a propriedade endereco ela é null. É por isso que a instrução p.endereco.logradouro resulta na exceção NullPointerException, pois p.endereco é null.

⭐ Questão 16 (Java)

A classe Pessoa e Endereco de Juarez fez sucesso:

```
public class Endereco {

    String logradouro;
    String complemento;
    String numero;
    String bairro;
    String cidade;
    String cep;
}

public class Pessoa {

    String nome;
    String cpf;
    int idade;
    Endereco endereco;

}

public class Programa {

    public static void main(String args[]) {

        Pessoa p = new Pessoa();
        p.nome = "Paulo";
        p.endereco.logradouro  = "Avenida XYZ";
    }
}
```

Marque as opções que contém a alteração que fará o código funcionar em tempo de execução:

Respostas:

```
public class Pessoa {

    String nome;
    String cpf;
    int idade;
    Endereco endereco = new Endereco();
}
```

```java
public class Programa {

    public static void main(String args[]) {

        Pessoa p = new Pessoa();
        p.nome = "Paulo";
        p.endereco = new Endereco();
        p.endereco.logradouro  = "Aveninda XYZ";
    }
}
```

⭐ Questão 17 (Java)

Luan resolveu criar um modelo para treinar orientação a objeto e encapsulamento:

```java
public class Cliente {
    String nome;
    private String cpf;
    int idade;
}
```

E está criando um objeto Cliente na outra classe:

```java
public class Banco {
  public static void main(String[] args) {
    Cliente cliente = new Cliente();
    cliente.nome = "José";
    cliente.cpf = "12312312312";
    cliente.idade = 49;
  }
}
```

O que podemos afirmar sobre o código?

Resposta: Não compila pois o cpf é privado.

⭐ Questão 18

Paulo está criando getters e setters para seguinte classe:

```
public class Aluno {
    private String nome;
    private String matricula;
}
```

Como ficaria, seguindo a convenção explicada no vídeo anterior, a declaração dos getters e setters para os dois atributos da classe?

Resposta:

```
public String getNome(){
   return this.nome;
}

public String getMatricula(){
   return this.matricula;
}

public void setMatricula(String matricula){
   this.matricula = matricula;
}

public void setNome(String nome){
   this.nome = nome;
}
```

⭐ Questão 19 (Java)

Rômulo criou uma classe com diversos atributos privados, porém não sabe exatamente qual vantagem de utilizar esta abordagem. Qual das opções melhor define a vantagem do uso de atributos privados?

Resposta: A implementação interna pode ser modificada sem afetar nenhum código fora da própria classe.

⭐ Questão 20 (Java)

Abaixo temos algumas afirmações a respeito da utilização de construtores, qual delas é verdadeira?

Resposta: Construtores são utilizados para inicialização dos atributos.

⭐ Questão 21 (Java)

Ainda sobre o jogo de Luan, temos outro trecho do código:

```
public class Jogo {
    //Código omitido
    private Componente comp;
    public Jogo(Usuario usuario){
       this.comp = usuario;
    }
}
```

Porém o código acima sequer compila. Qual dos motivos abaixo explica a razão desse acontecimento?

Resposta: Está sendo feita uma atribuição de objetos de tipos diferentes.

⭐ Questão 22 (Java)

Luan desenvolveu um jogo, e quer sempre manter o número de jogadores atualizados. Para isso Luan escreveu o código abaixo:

```
public class Jogador {
    //Código omitido
    private int total = 0;

    public Jogador(//atributos){
       total++;
    }
}
```

Porém o contador sempre apresenta 1 após inserir um novo jogador. Qual dos motivos abaixo explica a razão desse acontecimento?

Resposta: O total deveria estar como static, assim sempre que fosse criado um novo objeto do tipo Jogador não seria criado um novo total, mantendo o valor correto.