Quais das seguintes são membros válidos de instância de uma classe? (Escolha todas as que se aplicam.)"

**A. var var = 3;**
**B. Var case = new Var();**
**C. void var() {}**
**D. int Var() { var _ = 7; return _;}**
**E. String new = "var";**
**F. var var() { return null; }**

BOYARSKY, Jeanne; SELIKOFF, Scott. OCP **Oracle® Certified Professional Java SE 17 Developer Study Guide**. John Wiley & Sons, 2022.

## ANÁLISE DA QUESTÃO

A. var var = 3;

- *var* é **permitido apenas para variáveis locais, não** pode ser usado como tipo de variável membro de instância de uma classe. Portanto, esta declaração é inválida.

B. Var case = new Var();

- *case* é uma palavra reservada e não pode ser usada como identificador. Logo, esta declaração é inválida.

C. void var() {}

- Correto, esta é uma declaração válida de um método. *void* indica que o método não retorna nenhum valor e *var()* é um nome de método válido.

D. int Var() { var _ = 7; return _;}

- O uso de um **único sublinhado** _ como identificador é inválido a partir do Java 9.

E. String new = "var";

- *new* é **uma palavra reservada** e não pode ser usada como identificador. Portanto, esta declaração é inválida.

E. var var() { return null; }

- *var* **não pode ser usado como o tipo de retorno de um método**. O uso de *var* para inferência de tipo é restrito a variáveis locais.

A única **opção válida** é: **C. void var() {}**

## *var em Java*

Vamos falar sobre o uso do var no Java, um recurso que chegou para dar uma mãozinha na hora de declarar variáveis.

### a) Introdução ao var no Java

O *var* foi introduzido no Java na versão 10, em março de 2018. Ele faz parte de um movimento que visa **simplificar o código, melhorando a legibilidade e reduzindo a verbosidade sem sacrificar a segurança e a clareza** que são características do Java.

### b) Uso do *var* em Java

No Java, **o var é utilizado para inferir o tipo de uma variável a partir da expressão que é atribuída a ela**. Ou seja, ao invés de você especificar o tipo da variável **explicitamente**, o **compilador do Java vai fazer isso** para você com base no valor que você atribuiu à variável.

Exemplo clássico:

```
var numero = 10; // O compilador infere que "numero" é do tipo int.
```

Aqui, o *var* substitui a declaração explícita do tipo *int*. O compilador vê que você atribuiu o valor 10, que é um *int*, e então infere que a variável numero é do tipo *int*.

### c) Onde o *var* é utilizado

O *var* pode ser utilizado em diversos contextos, mas com algumas limitações. Aqui estão alguns exemplos de onde você pode usar var:

- *Dentro de Métodos:* É o uso mais comum. Você pode usar var para variáveis locais dentro de métodos.

```
public void exemplo() {
    var lista = new ArrayList<String>(); // "lista" é inferido como ArrayList<String>
}
```

- *Laços de iteração:* Ele pode ser usado em loops para simplificar a declaração das variáveis de controle..

```
for (var i = 0; i < 10; i++) {
    System.out.println(i);
}
```

- *Com Streams e Coleções:* Em expressões complexas com streams e coleções, o var pode tornar o código mais limpo.

```
var nomes = List.of("Ana", "João", "Maria");
nomes.forEach(nome -> System.out.println(nome));
```

**d) Restriçoes do *var***

O uso do var tem algumas limitações importantes:

- *Só Pode Ser Usado para Variáveis Locais:* O var não pode ser usado para variáveis de instância, variáveis de classe, parâmetros de métodos, ou variáveis de captura em expressões lambda.

- *A Inicialização é Obrigatória:* Você precisa inicializar a variável no mesmo momento em que a declara, já que o compilador precisa dessa informação para inferir o tipo.

```
var numero; // Isso não compila porque não há inicialização.
numero = 10; // Não é permitido.
```

- *Não Pode Ser Usado em Tipos Anônimos:* Não é possível usar var para tipos que não têm uma expressão clara e explícita.

```
var obj = new Object() { // Isso não é permitido.
    void metodo() {}
};
```

- *Legibilidade:* Em alguns casos, o uso de var pode prejudicar a legibilidade, especialmente quando o tipo da variável não é óbvio pelo contexto..

```
var lista = metodoQueRetornaAlgo(); // Pode ser difícil de saber o tipo de
"lista".
```

**e) Porque usar *var***

- *Redução da Verbosidade*: Em muitos casos, o tipo da variável é redundante porque já é claro a partir do lado direito da atribuição.

```
ArrayList<String> lista = new ArrayList<>(); // Verboso.
var lista = new ArrayList<String>(); // Menos verboso.
```

- *Facilita a Manutenção:* Quando você muda o tipo de uma variável, você só precisa mudar do lado direito da atribuição. O lado esquerdo (var) permanece o mesmo.

```
var valor = getAlgumValorComplexo();
```

- *Código Mais Limpo e Legível:* Em muitos casos, o uso de var pode tornar o código mais limpo, especialmente em expressões complexas.

```
var mapa = new HashMap<String, List<String>>();
```

Usar *var* é uma **questão de balanço entre simplicidade e clareza**. É uma ótima ferramenta para reduzir a verbosidade e tornar o código mais conciso, mas deve ser usada com cuidado para não sacrificar a legibilidade. A chave é entender quando ele realmente ajuda a tornar o código mais claro e quando pode gerar confusão.