Qual é o resultado de compilar e executar o seguinte programa?

```
public class Exercicio_019 {
    public static void main(String[] args) {
    var x = 5;
    var j = 0;
    OUTER: for (var i = 0; i < 3;)
        INNER: do {
        i++;
        x++;
        if (x> 10) break INNER;
        x += 4;
        j++;
        } while (j <= 2);
    System.out.println(x);
    }
}
```

A. 10
B. 11
C. 12
D. 17
E. O código não compilará por causa da linha 5.
F. O código não compilará por causa da linha 6.

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

Vamos analisar este código passo a passo:

1. Inicialmente, temos: `x = 5 j = 0 i = 0` (declarado no loop for)
2. Entramos no loop for (OUTER):
3. Entramos no loop do-while (INNER):
    - Primeira iteração: `i++ (i = 1) x++ (x = 6) x += 4 (x = 10) j++ (j = 1)`
    - Segunda iteração: `i++ (i = 2) x++ (x = 11)` Aqui, x > 10, então o break INNER é acionado, saindo do loop do-while
4. Voltamos ao loop for: `i = 2`, que é menor que 3, então continuamos
5. Entramos novamente no do-while: `i++ (i = 3) x++ (x = 12)` break INNER é

acionado imediatamente
6. O loop for termina porque `i = 3`
7. Finalmente, imprimimos o valor de x

Portanto, o resultado final será:

```
12
```

A alternatica correta para a questão, portanto, é a **C**.

Note também que você pode ter um laço `for` sem o bloco `{}` em Java. Quando você omite as chaves `{}`, o corpo do laço é considerado apenas a próxima única instrução após o laço for. Isso se aplica a todos os tipos de laços (`for, while, do-while`) e estruturas condicionais (`if, else`)

O uso de labels em estruturas de repetição em Java

Labels são identificadores que você pode usar para nomear loops e, em seguida, usar com as instruções `break` e `continue` para controlar o fluxo de execução de forma mais precisa. Vamos ver exemplos para cada tipo de loop:

3. FOR loop com label:

```java
outerLoop: for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
        if (i * j > 6) {
            System.out.println("Saindo do loop externo");
            break outerLoop;
        }
        System.out.println(i + " * " + j + " = " + (i*j));
    }
}
```

O resultado será:

```
0 * 0 = 0
0 * 1 = 0
0 * 2 = 0
0 * 3 = 0
0 * 4 = 0
```

```
1 * 0 = 0
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
2 * 0 = 0
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
Saindo do loop externo
```

4. **`WHILE` loop com label:**

```java
int i = 0;
outerWhile: while (i < 5) {
    int j = 0;
    while (j < 5) {
        if (i * j > 6) {
            System.out.println("Continuando o loop externo");
            i++;
            continue outerWhile;
        }
        System.out.println(i + " * " + j + " = " + (i*j));
        j++;
    }
    i++;
}
```

O resultado será:

```
0 * 0 = 0
0 * 1 = 0
0 * 2 = 0
0 * 3 = 0
0 * 4 = 0
1 * 0 = 0
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
2 * 0 = 0
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
```

```
Continuando o loop externo
3 * 0 = 0
3 * 1 = 3
3 * 2 = 6
Continuando o loop externo
4 * 0 = 0
4 * 1 = 4
Continuando o loop externo
```

5. **`DO...WHILE` loop com label:**

```java
int i = 0;
outerDoWhile: do {
    int j = 0;
    do {
        if (i * j > 6) {
            System.out.println("Saindo do loop externo");
            break outerDoWhile;
        }
        System.out.println(i + " * " + j + " = " + (i*j));
        j++;
    } while (j < 5);
    i++;
} while (i < 5);
```

O resultado será:

```
0 * 0 = 0
0 * 1 = 0
0 * 2 = 0
0 * 3 = 0
0 * 4 = 0
1 * 0 = 0
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
2 * 0 = 0
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
Saindo do loop externo
```

Nos exemplos acima, os labels (`outerLoop`, `outerWhile`, `outerDoWhile`) são usados para identificar o loop externo. Isso permite que as instruções `break` e `continue` afetem o

loop externo, mesmo quando são chamadas de dentro do loop interno.

O uso de labels pode tornar o código mais legível e controlável, especialmente em situações com loops aninhados complexos. No entanto, é importante usá-los com moderação, pois o uso excessivo pode tornar o código mais difícil de entender.