# **Evaluación teórica - Arreglos y vectores**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Realice esta evaluación y logre obtener puntos de ¡**Experiencia**, **Competencia** y **Habilidad**! ¡Adelante!

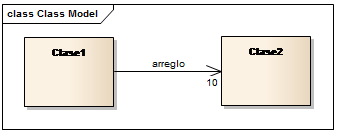
Este desafío busca evaluar la comprensión e interpretación que usted ha logrado de los contenidos del módulo 2.

Para poder responder estas preguntas, idealmente debe revisar los contenidos teóricos y prácticos presentados en las primeras secciones en donde se explican los conceptos básicos de arreglos y vectores. La estructura de esta evaluación es de selección múltiple, en la que usted deberá:

* Analizar varias implementaciones de algoritmos en sintaxis java simple.
* Responder preguntas de análisis basados en el código presentado y en diagramas UML.

El desafío consta de 12 preguntas. Cada una tiene varias opciones de respuesta pero sólo una es correcta. Al final recibirá la retroalimentación de su desempeño.

1. Considere el siguiente modelo:



La forma de declarar la asociación arreglo es:

1 punto

private Clase2[ ] arreglo;

private Clase2Clase1[ ] arreglo;

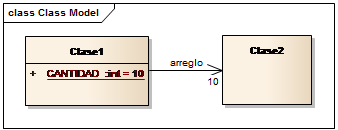
private Clase2[Clase1] arreglo;

private Clase1[ ] arreglo;

### 2.

Pregunta 2

Considere el siguiente modelo:



La forma de inicializar la asociación arreglo es:

1 punto

arreglo = new Clase2[CANTIDAD];

arreglo = new Clase1[CANTIDAD];

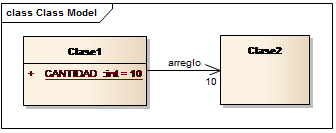
arreglo = Clase1[CANTIDAD];

arreglo = Clase2[CANTIDAD];

### 3.

Pregunta 3

Considere el siguiente modelo y el siguiente fragmento de código:



public class Clase1

{

public int metodo( )

{

int x = 0;

for ( int i = 0; i < CANTIDAD; i++ )

{

if( arreglo[ i ] != null )

{

x++;

}

}

return x;

}

}

El recorrido presentado es:

1 punto

Un recorrido parcial

Un recorrido total

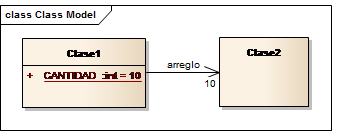
Un ciclo infinito

No es realmente un ciclo.

### 4.

Pregunta 4

Considere el siguiente modelo y fragmento de código:



public class Clase1

{

public int metodo( )

{

int x = 0;

for ( int i = 0; i < CANTIDAD && x < 5; i++ )

{

if( arreglo[ i ] != null )

{

x++;

}

}

return x;

}

}

El recorrido presentado es:

1 punto

No es realmente un recorrido

Un bucle infinito

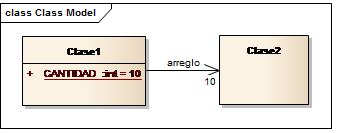
Un recorrido parcial

Un recorrido total

### 5.

Pregunta 5

Considere el siguiente modelo y el siguiente fragmento de código:



Considere el siguiente modelo y el siguiente fragmento de código:

public class Clase1

{

public void metodo( )

{

int i = 0;

while( i < CANTIDAD / 2 )

{

if( arreglo[ i ] == null )

{

arreglo[ i ] = new Clase2( );

}

i +=3;

}

}

}

El avance y el inicio del ciclo son respectivamente:

1 punto

i += 3; y i = 0;

i < CANTIDAD / 2; y i = 0;

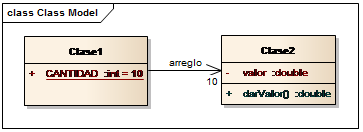
i += 3; y i < CANTIDAD / 2;

i = 0; y i += 3;

### 6.

Pregunta 6

Considere el siguiente modelo y el siguiente fragmento de código:



public Clase2 metodo( )

{

Clase2 y = null;

for ( int i = 0; i < CANTIDAD; i++ )

{

if( y == null )

{

y = arreglo[ i ];

}

else if( y != null && arreglo[ i ] != null && y.darValor( ) < arreglo[ i ].darValor( ) )

{

y = arreglo[ i ];

}

}

return y;

}

Al final, el método debe retornar:

1 punto

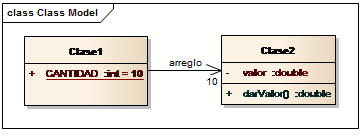
El elemento con menor valor del arreglo o null en caso que el arreglo no tenga elementos

El primer elemento no nulo del arreglo o null en caso que el arreglo no tenga elementos

El elemento con mayor valor del arreglo o null en caso que el arreglo no tenga elementos

El último elemento no nulo del arreglo o null en caso que el arreglo no tenga elementos

Considere el siguiente modelo y el siguiente fragmento de código:



public Clase2 metodo( )

{

Clase2 y = null;

for ( int i = 0; i < CANTIDAD && y == null; i++ )

{

if( y == null )

{

y = arreglo[ i ];

}

else if( y != null && arreglo[ i ] != null && y.darValor( ) < arreglo[ i ].darValor() )

{

y = arreglo[ i ];

}

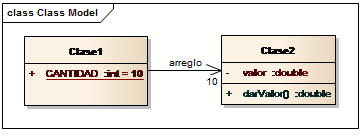
}

return y;

}

Pregunta 7

Considere el siguiente modelo y el siguiente fragmento de código:



22

23

24

25

26

       }

       return y;

}

Al final, el método debe retornar:

1 punto

El elemento con menor valor del arreglo o null en caso que el arreglo no tenga elementos

El último elemento no nulo del arreglo o null en caso que el arreglo no tenga elementos

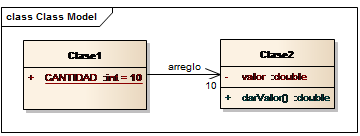
El elemento con mayor valor del arreglo o null en caso que el arreglo no tenga elementos

El primer elemento no nulo del arreglo o null en caso que el arreglo no tenga elementos

### 8.

Pregunta 8

Considere el siguiente modelo y el siguiente fragmento de código:



public Clase2 metodo( )

{

Clase2 y = null;

for ( int i = 0; i < CANTIDAD; i++ )

{

if( y == null )

{

y = arreglo[ i ];

}

else if( y != null && arreglo[ i ] != null && y.darValor( ) <= arreglo[ i ].darValor( ) )

{

y = arreglo[ i ];

}

}

return y;

}

Cuál de las siguientes afirmaciones con respecto a la implementación del método metodo( ) es verdadera:

1 punto

Devuelve el elemento con mayor valor del arreglo y en casos que existan varios con el mayor valor devuelve el último de estos

Devuelve el elemento que es mayor o igual que todos los demás elementos del arreglo

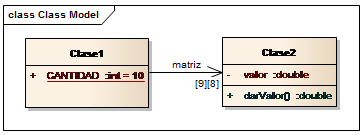
Devuelve el elemento con mayor valor del arreglo y en casos que existan varios con el mayor valor devuelve el primero de estos

Devuelve siempre el elemento en la posición más grande del arreglo, o null si está no contiene ningún elemento.

### 9.

Pregunta 9

Considere el siguiente modelo:



La forma de declarar la asociación matriz es:

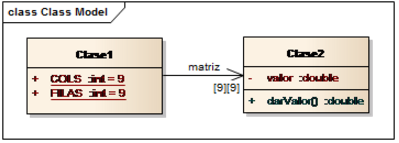
private Clase1[ ][ ] matriz;

private Clase2[ ][ ] matriz;

private Clase2[Clase1][9] matriz;

private Clase2Clase1[ ][ ] matriz;

Considere el siguiente modelo y el siguiente fragmento de código:



public int metodo( )

{

int x = 0;

for ( int i = 0; i < FILAS; i++ )

{

x += matriz[ i ][ i ].darValor( );

}

return x;

}

Cuál de las siguientes afirmaciones sobre el método metodo( ) es verdadera:

1 punto

Suma todos los valores de la diagonal que inicia en la esquina superior derecha de la matriz

No corresponde a un recorrido sobre una matriz ya que no utiliza un doble ciclo

Suma todos los valores de la diagonales que pasan por el elemento central de la matriz

Suma todos los valores de la diagonal que inicia en la esquina superior izquierda de la matriz

### 11.

Pregunta 11

Suponga que usted tiene la siguiente matriz:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

Considere el siguiente algoritmo que manipula la matriz.

Nota: Pares son las posiciones 0,2,4, etc.

class Main

{

public int [][] matrix;

public Main()

{

matrix = new int[3][4];

}

public void metodo()

{

int a = 1;

for(int i = 0; i<matrix.length;i++)

{

for(int j = 0; j<matrix[0].length;j++)

{

matrix[i][j] = a;

a++;

}

}

}

public int metodo1()

{

int num = 0;

for(int i = 0; i < matrix[0].length; i++)

{

num+=matrix[matrix.length-1][i];

}

return num;

}

public int metodo2()

{

int num = 0;

for(int i = 1; i < matrix[0].length; i+=2)

{

for(int j = 0; j < matrix.length; j++)

{

num+=matrix[j][i];

}

}

return num;

}

public static void main(String[] args)

{

Main m = new Main();

m.metodo();

System.out.println(m.metodo1()+m.metodo2());

}

}

¿Qué está haciendo el algoritmo completo (método main)?

1 punto

Crea la matriz de números enteros.

Inicializa la matriz de números.

Suma los valores de la primera fila de la matriz.

Suma los valores de las columnas pares de la matriz.

Imprime la suma de los dos pasos anteriores.

Crea la matriz de números enteros.

Inicializa la matriz de números.

Imprime la suma de los valores de la última fila de la matriz.

Crea la matriz de números enteros.

Inicializa la matriz de números.

Suma los valores de la última fila de la matriz.

Suma los valores de las columnas impares de la matriz.

Imprime la suma de los dos pasos anteriores.

Crea la matriz de números enteros.

Inicializa la matriz de números.

Suma los valores de la primera fila de la matriz.

Suma los valores de las columnas impares de la matriz.

Imprime la suma de los dos pasos anteriores.

Suponga que usted tiene el siguiente arreglo:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 | 23 | 15 | 7 | 14 | 9 |

Considere el siguiente algoritmo que manipula el arreglo.

Nota: Pares son las posiciones 0,2,4, etc.

class Main

{

public int [] arreglo;

public Main()

{

arreglo = new int[6];

}

public void metodo()

{

arreglo[0]=32;

arreglo[1]=23;

arreglo[2]=15;

arreglo[3]=7;

arreglo[4]=14;

arreglo[5]=9;

}

public int metodo1()

{

int a = 0;

for(int i = 0; i < arreglo.length;i+=2)

{

a+=arreglo[i];

}

return a;

}

public int metodo2()

{

int a = 0;

for(int i = 1; i < arreglo.length;i+=2)

{

a+=arreglo[i];

}

return a;

}

public static void main(String[] args) {

Main main = new Main();

main.metodo();

System.out.println(main.metodo1()- main.metodo2());

}

}

Pregunta 12

Suponga que usted tiene el siguiente arreglo:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 | 23 | 15 | 7 | 14 | 9 |

Considere el siguiente algoritmo que manipula el arreglo.

Nota: Pares son las posiciones 0,2,4, etc.

37

38

39

40

41

42

43

44

45

        return a;

    }

  public static void main(String[] args) {

        Main main = new Main();

        main.metodo();

        System.out.println(main.metodo1()- main.metodo2());

  }

}

¿Qué está haciendo el algoritmo completo (método main)?

1 punto

Crea el arreglo.

Inicializa el arreglo con sus valores iniciales.

Imprime la resta entre la suma de los valores que se encuentran en las posiciones impares y la suma de los valores que se encuentran en las posiciones pares.

Crea el arreglo.

Imprime la resta entre la suma de los valores que se encuentran en las posiciones impares y la suma de los valores que se encuentran en las posiciones pares.

Crea el arreglo.

Imprime la resta entre la suma de los valores que se encuentran en las posiciones pares y la suma de los valores que se encuentran en las posiciones impares.

Crea el arreglo.

Inicializa el arreglo con sus valores iniciales.

Imprime la resta entre la suma de los valores que se encuentran en las posiciones pares y la suma de los valores que se encuentran en las posiciones impares.