UT 03 – Utilización de objetos – Ejercicios arrays

Ejercicio 11 – Arrays y métodos

Crea un programa que

- Cree un array de 100 números aleatorios entre 1000 y 2000.
- Pregunte al usuario un número entre 1000 y 2000.
- Cuente cuántas veces aparece en el array el número que ha introducido el usuario.

Utiliza métodos en tu programa. Al menos:

- Un método para generar el array de números aleatorios. Recibe la cantidad de números a generar, y devuelve un nuevo array con los números generados.
- Un método para buscar en el array el número que ha introducido el usuario.

Ejercicio 12 – Arrays y métodos

Crea un programa que:

- Pregunte al usuario cuántos números desea generar, que tendrá que ser un valor entre 10 y 100 ambos incluidos. Si el usuario introduce un número menor que 10 o mayor que 100 se le volverá a preguntar.
- Cree un array con tantos números aleatorios como haya indicado el usuario. Cada número tendrá que ser un valor entre 0 y 1000 ambos incluidos, y no podrán repetirse, no podrá aparecer el mismo número en el array dos veces.
- Muestre el array en pantalla.

A tener en cuenta:

- El array se debe crear con un método que:
    - Recibe la cantidad de números a generar.
    - Recibe el valor mínimo y el valor máximo para generar los números aleatorios
    - Devuelve el array con los números creados.
- El array se debe mostrar con un método que muestra los números del array separados por guiones (-).

Ejercicio 13 – Arrays y métodos – Distribución uniforme en Math.random()

Crea un programa que sirva para comprobar si los números generados con Math.random siguen una distribución uniforme, es decir, que, si generamos muchos números, todos los números se generar más o menos las mismas veces. Para ello, vamos a:

- Generar un número importante de números aleatorios, entre 0 y 99. Podemos empezar con 10000. Usa una constante para este número, así podremos cambiarlo fácilmente, para probar con más o con menos.
- Contar cuantas veces aparece cada uno de los números.
- Obtener qué número ha aparecido más veces.
- Obtener qué número ha aparecido menos veces.

Para contar las apariciones usa un array de enteros de 100 posiciones, inicializado a cero (todas sus posiciones tienen valor cero). Cada vez que se genera un número incrementamos la posición correspondiente del array en uno, para contar cuantas veces ha aparecido ese

número. Así, en la posición 0 tendremos cuantas veces ha aparecido el cero, en la uno, cuantas veces el uno, y así sucesivamente.

Una vez generados y contabilizados, vamos a mostrar los valores del array, y contar cuántas veces aparece el que más veces aparece, y cuantas el que menos.

¿Estos valores se separan mucho de lo esperado? Lo esperado sería que cada número apareciera la misma cantidad de veces, que se obtiene al dividir la cantidad total de números generados por la cantidad de números diferentes que generamos.

Ejemplo: generamos 50000 números aleatorios. Cada número está entre 0 y 99 ambos incluidos. En total tenemos 100 números distintos. Si todos aparecen la misma cantidad de veces cada uno debería aparecer 50000/100 veces = 500 veces.

Ejercicio 14 – Arrays y métodos

Crear un programa que:

- Pida al usuario 5 números enteros, entre 1 y 100, y los almacene en un array.
- Genere un array de 25 números aleatorios entre 1 y 100.
- Calcule qué números de los que ha introducido el usuario están en el array de números aleatorios.
- Muestre en pantalla los números que sí estaban en el array de aleatorios.

Para hacerlo:

- Crear un método “pedirNumero” que:
    - Recibe dos parámetros “minimo” y “máximo”, y un scanner ya creado, para poder preguntar al usuario.
    - Pide al usuario un número entre esos dos valores, ambos incluidos.
    - Si el usuario no introduce un valor válido, vuelve a preguntar hasta que el usuario lo introduzca correctamente.
    - Devuelve el valor introducido por el usuario.
- Crear un método “pedirNumerosAUsuario” que:
    - Recibe la cantidad de números que tiene que pedir al usuario.
    - Recibe el máximo y el mínimo (ambos incluidos) que puede escribir el usuario.
    - Crea un array del tamaño adecuado para guardar los números.
    - Usando el método “pedirNumero”, pregunta al usuario todos los números necesarios para llenar el array.
    - Devuelve el array de números.
- Crear un método “generarNumerosAleatorios” que:
    - Recibe la cantidad de números a generar, y los valores máximo y mínimo (ambos incluidos) para los números generados.
    - Crea un array del tamaño adecuado.
    - Genera números aleatorios en el rango indicado y rellena con ellos el array.
    - Devuelve el array.
- Crear un método “buscarNumeros” que:
    - Recibe dos arrays, uno con el conjunto de números que buscamos (el array de números que ha cargado el usuario) y otro con el conjunto de números entre los que queremos buscar (el array de aleatorios).
    - Devuelve un array de boolean del mismo tamaño que el array de números buscados (cargados por el usuario), de forma que en cada posición del array

devuelto aparece true si el número en la misma posición del array de números del usuario aparece en alguna posición del array de aleatorios, y false en caso contrario.

- Crear un método “mostrarEncontrados” que:
    - Recibe dos arrays:
        - El array de números cargado por el usuario
        - El array de boolean indicando si los números del usuario se encontraron en el array de aleatorios.
    - Muestra, para cada número que introdujo el usuario, si apareció o no en el array de aleatorios.

Ejemplo de ejecución:

```
Introduce un número entre 1 y 100 a.i. 0
Número incorrecto. Introduce un número entre 1 y 100 a.i. 2
Introduce un número entre 1 y 100 a.i. 6
Introduce un número entre 1 y 100 a.i. 9
Introduce un número entre 1 y 100 a.i. 44
Introduce un número entre 1 y 100 a.i. 1110
Número incorrecto. Introduce un número entre 1 y 100 a.i. 19
2 - encontrado.
6 - no encontrado.
9 - no encontrado.
44 - no encontrado.
19 - encontrado.
```

Ejercicio 15 – Arrays – Inserción en arrays

Hacer un programa en Java que permita insertar valores en un array manteniéndolo ordenado. Para ellos vamos a:

- Crear un array de 10 posiciones, que inicializaremos todas con el valor cero.
- Pediremos al usuario que introduzca números mayores que cero. Seguiremos pidiendo números hasta que el usuario introduzca un número menor o igual a cero.
- Cada número que el usuario introduzca se incluirá en el array, pero manteniendo el array ordenado de menor a mayor.
- Los ceros en el array indican posiciones “vacías”, que no han sido utilizadas.
- Al insertar un número en el array, los números “a la derecha” del insertado deben desplazarse para dejarle sitio.
- Si el array está ya lleno (no quedan ceros al final):
    - Si el número a insertar es mayor que todos los del array, se descarta (no se inserta.
    - Si se inserta, al desplazar todos los números a la derecha, el que hasta entonces era el mayor número se descartará.
- Cada vez que el usuario indica un número se mostrará el array antes y después de la inserción.

Ejemplo de ejecución con un array de 5 posiciones en lugar de 10:

```
Introduce un número mayor que cero: 7
Antes  de  insertar 7: 0,0,0,0,0
Despues de insertar 7: 7,0,0,0,0
Introduce un número mayor que cero: 4
Antes  de  insertar 4: 7,0,0,0,0
Despues de insertar 4: 4,7,0,0,0
Introduce un número mayor que cero: 9
Antes  de  insertar 9: 4,7,0,0,0
Despues de insertar 9: 4,7,9,0,0
Introduce un número mayor que cero: 1
Antes  de  insertar 1: 4,7,9,0,0
Despues de insertar 1: 1,4,7,9,0
Introduce un número mayor que cero: 8
Antes  de  insertar 8: 1,4,7,9,0
Despues de insertar 8: 1,4,7,8,9
Introduce un número mayor que cero: 7
Antes  de  insertar 7: 1,4,7,8,9
Despues de insertar 7: 1,4,7,7,8
Introduce un número mayor que cero: 9
Antes  de  insertar 9: 1,4,7,7,8
Despues de insertar 9: 1,4,7,7,8
Introduce un número mayor que cero: 0
```