Programación

Bloque 06 - Estructuras de almacenamiento

Arrays Unidimensionales

Todos las clases correspondientes a los ejercicios de esta relación se deben crear dentro del paquete `prog.unidad06.arraysuni`. Cada ejercicio se deberá almacenar en su propio paquete bajo éste, llamando a cada paquete `ejercicio01`, `ejercicio02`, etc.

1. Crea un programa en la clase `Array8App`, que lea 8 números reales desde teclado y los almacene en un array. A continuación debe calcular:
   - Suma total de todos los elementos
   - Suma de los elementos del array mayores a 36
   - Cantidad de valores mayores a 50
   - Media de los valores entre 10 y 30, ambos incluidos

Ejemplo de uso:

```
OPERACIONES VARIAS SOBRE UN ARRAY DE 8 ELEMENTOS
A continuación se van a solicitar 8 números reales
Introduce el número 1: 10
Introduce el número 2: 2.3
Introduce el número 3: 89
Introduce el número 4: -1
Introduce el número 5: 87
Introduce el número 6: 3
Introduce el número 7: 45
Introduce el número 8: -10
La suma de todos los elementos vale 225.3
La suma de los elementos mayores de 36 vale 221.0
La cantidad de elementos mayores a 50 es de 2
La media de los 1 elementos comprendidos entre 10 y 30 vale 10.0
Fin del programa
```

2. Crea una clase, llamada `UtilidadesArrays` que tenga el método estático `int[] generaArrayEnteroAleatorio(int longitud, int valorMinimo, int valorMaximo)`
   Este método crea un array con `longitud` números enteros tomados aleatoriamente del intervalo entre `valorMinimo` y `valorMaximo`.

   Se lanza `IllegalArgumentException` si la `longitud` es menor que 1 o si el `valorMinimo` es mayor que `valorMaximo`.

   Usa las pruebas JUnit en el archivo `actividad_6.01b_pruebas.zip` para comprobar la corrección de la clase
3. Crea un programa, llamado `BusquedaNotasApp`, que ayude a un profesor a buscar cuantos alumnos han sacado una determinada calificación (entera). Las

calificaciones de los 30 alumnos se almacenarán en un array de valores enteros aleatorios. El programa pedirá una nota por teclado y el programa contestará con el número de alumnos con dicha nota y se volverá a pedir otra nota. Se terminará cuando se introduzca una nota negativa.

Ejemplo de uso:

```
BÚSQUEDA DE CALIFICACIONES
Introduzca la calificación a buscar (0 a 10, negativa para
terminar): 0
Con la calificacion 0 se han encontrado 2 alumnos
Introduzca la calificación a buscar (0 a 10, negativa para
terminar): 10
Con la calificacion 10 se han encontrado 3 alumnos
Introduzca la calificación a buscar (0 a 10, negativa para
terminar): 5
Con la calificacion 5 se han encontrado 2 alumnos
Introduzca la calificación a buscar (0 a 10, negativa para
terminar): a
Entrada incorrecta. Debe introducir un número entero entre 0 y 10,
ambos incluidos
Introduzca la calificación a buscar (0 a 10, negativa para
terminar): -1
FIN DEL PROGRAMA
```

4. Crea una nueva clase `UtilidadesArrays` en otro paquete (ejercicio04) que añada a lo que ya tenía, los siguientes métodos estáticos nuevos:
   - `int getMinimoArrayEntero(int[] array)`. Obtiene el valor mínimo contenido en el array. Si el array está vacío (longitud cero) o es `null` se lanza `IllegalArgumentException`.
   - `int getMaximoArrayEntero(int[] array)`. Obtiene el valor máximo contenido en el array. Se lanza `IllegalArgumentException` en las mismas condiciones que el método anterior.
   - `int enteroApareceEnArrayEntero(int[] array, int valor)`. Devuelve el número de veces que aparece valor en el array. Si se pasa un array `null` o vacío devuelve cero.

Usa las pruebas en el fichero `actividad_6.01d_pruebas.zip` para verificar

5. Realiza un programa, en la clase `ProcesaAleatorios`, que cree un array de 150 elementos aleatorios con valores entre 1 y 300 y que muestre por pantalla cual es el valor mayor, cual el menor y cuantas veces aparecen ambos dentro del array (como mínimo deberán aparecer una vez cada uno).
6. Crea una clase, llamada `EstadisticasNotas`, que calcule estadísticas sobre las calificaciones de un grupo. La clase tendrá el interfaz descrito en el archivo `actividad_6.01f_doc.zip` y se podrá probar empleando las pruebas contenidas en el fichero `actividad_6.01f_pruebas.zip`.
7. Realiza un programa, en la clase `SuperArray`, que pida por teclado un número entero. A partir de él crea un array de 30 elementos. El primer elemento contendrá el número introducido y los siguientes el valor del elemento anterior mas 1 y multiplicado por 2. Una vez creado el array, muéstralo por pantalla en orden inverso.

Ejemplo de uso:

```
Super Array
Introduce un número entero cualquiera: 10
El array al revés es
```

```
6442450942, 3221225470, 1610612734, 805306366, 402653182,
201326590, 100663294, 50331646, 25165822, 12582910, 6291454,
3145726, 1572862, 786430, 393214, 196606, 98302, 49150, 24574,
12286, 6142, 3070, 1534, 766, 382, 190, 94, 46, 22, 10
```

8. Realiza un programa, `ArrayPares`, que cree un array de 20 elementos, cargado con los 20 primeros números pares (comenzando por 2). Muestra este array por pantalla.

Ejemplo de uso:

```
NÚMEROS PARES
Los primeros números pares son: 2, 4, 6, 8, 10, 12, 14, 16, 18,
20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40
```

9. Realiza un programa, `DividirPorLaMedia`, que cree un array entero con 50 posiciones a crear con números aleatorios enteros entre 300 y 800. A partir de él crea otro array de tipo real en el que todos los elementos resulten de dividir el elemento correspondiente del array original entre la media de los elementos del array original. Imprime ambos arrays por pantalla.

Ejemplo de uso:

```
DIVIDIR POR LA MEDIA
El array entero es: 561, 674, 468, 424, 438, 792, 786, 349, 487,
566, 479, 316, 336, 645, 773, 323, 427, 794, 512, 585, 500, 422,
392, 775, 529, 319, 361, 346, 602, 601, 441, 344, 599, 508, 711,
659, 502, 568, 628, 646, 680, 718, 529, 659, 747, 304, 610, 406,
694, 422
El array real es: 1.0405460548280596, 1.2501391104351374,
0.8680491152576325, 0.7864376599770004, 0.812404941202656,
1.4690061950513782, 1.457877360240383, 0.6473272248395593,
0.903290425492451, 1.0498200838372223, 0.8884519790777906,
0.5861186333790852, 0.6232147494157362, 1.1963497421819935,
1.4337648848165598, 0.5991022739919131, 0.7920020773824981,
1.4727158066550432, 0.9496605705382647, 1.0850613940720406,
0.9274029009162741, 0.7827280483733353, 0.7270838743183589,
1.4374744964202248, 0.981192269169418, 0.5916830507845828,
0.6695848944615499, 0.6417628074340617, 1.116593092703194,
1.1147382869013616, 0.8179693586081538, 0.6380531958303965,
1.1110286752976963, 0.9422413473309345, 1.3187669251029417,
1.2223170234076493, 0.9311125125199392, 1.0535296954408873,
1.1648180435508402, 1.1982045479838261, 1.2612679452461328,
1.3317505657157696, 0.981192269169418, 1.2223170234076493,
1.3855399339689134, 0.5638609637570946, 1.1314315391178544,
0.7530511555440146, 1.2872352264717883, 0.7827280483733353
```

10. Realiza un programa, `BuscarYReemplazar`, que genere un array con 30 números aleatorios enteros del 0 al 20 y que los muestre por pantalla. A continuación se pedirán por teclado dos valores enteros y el programa cambiará todas las ocurrencias en el array del primer valor por el segundo y volverá a imprimir el array modificado. **NOTA: Los números que se hayan cambiado deben aparecer entrecomillados con comillas simples (')**

Ejemplo de uso:

```
BUSCAR Y REEMPLAZAR EN ARRAY
Array original
16 7 20 2 10 11 19 9 8 11 5 2 10 14 18 9 11 1 4 17 1 10 15 7 11 3
1 7 15 11
```

```
Introduzca el valor a sustituir: 10
Introduzca el valor sustituto: 11
16 7 20 2 '11' 11 19 9 8 11 5 2 '11' 14 18 9 11 1 4 17 1 '11' 15 7
11 3 1 7 15 11
```