UNIVERSIDADE DA CORUÑA

PROGRAMACIÓN ORIENTADA A OBJETOS. CURSO 2007-2008

Boletín de Ejercicios 1

NOTA: Lee las instrucciones de los ejercicios antes de empezar a realizarlos

1. Escribe un programa que muestre por pantalla la fecha y la hora actuales correctamente formateadas.

   Tema: Iniciación a Java y a su API
   De utilidad: Método `println` de `System.out` y la clase `Date`

2. Crea un método que dado un `String` pasado por parámetro devuelva un valor booleano que indique si el `String` es un palíndromo. Un palíndromo es una palabra que se lee igual tanto si empiezas por el principio como por el final (ignorando las diferencias entre mayúsculas y minúsculas). Por sencillez supondremos que la palabra sólo contiene las letras del abecedario inglés, sin caracteres acentuados o especiales (como espacios).

   Ejemplo: La palabra "Radar" es un palíndromo

   Tema: Bucles y manejo de Strings
   De utilidad: Métodos de la clase `String`

3. Crea un método que imprima los números primos entre 1 y 100 (ambos incluidos) mediante el método de la criba de Eratostenes. Este método consisten en crear un array con los 100 primeros números e ir marcando progresivamente los que son divisibles por 2 (recorriendo el array saltando de dos en dos), por 3, por 4, etc. Los númeroa que queden sin marcar serán primos, por lo que se puede recorrer el array imprimendo los no marcados ¿Es necesario llegar hasta 100 cubrir toda la criba?

   Tema: Bucle for, Arrays

4. Todos los libros aparecen identificados mediante un ISBN (International Standard Book Number). Actualmente existen dos modelos de ISBN, de 10 cifras y de 13 cifras. En el ISBN de 10 cifras el último valor es un dígito de control que puede calcularse a partir de los otros 9 dígitos, esto permite controlar errores en la introducción de ISBNs. El algoritmo de control es como sigue, se calcula un valor *x* calculado como:

   $$x = 10x_1 + 9x_2 + 8x_3 + 7x_4 + 6x_5 + 5x_6 + 4x_7 + 3x_8 + 2x_9$$

   siendo los x_n los dígitos del código ISBN. La última cifra deberá calcularse como:

$x_{10} = 11 - (x \text{ modulo } 11)$ si el resultado del módulo es distinto de cero

$x_{10} = 0$ si el resultado del módulo es cero

Como es posible que el último dígito sea 10 este se representará con el carácter X.

Ejemplo: Calcular el dígito de control para el número ISBN 030640615 (los guiones se ignoran).

$$x = (10x0) + (9x3) + (8x0) + (7x6) + (6x4) + (5x0) + (4x6) + (3x1) + (2x5) =$$

$$= 27 + 42 + 24 + 24 + 3 + 10 = 130$$

$$x_{10} = 11 - (130 \text{ modulo } 11) = 11 - 9 = 2$$

Por lo tanto el ISBN resultante sería: 0306406152

El objetivo del ejercicio es crear un método que acepte un `String` con los 9 primeros números de un ISBN y devuelva otro `String` que represente dicho ISBN pero con el número de control añadido. Si el `String` inicial no tiene 9 caracteres o alguno de los caracteres no es un número se debe lanzar la excepción `IllegalArgumentException`.

Tema: Bucles, cálculos numéricos, manejo de Strings

5. Un array desigual (ragged array) es un array bidimensional en el cual la segunda dimensión no tiene la misma longitud (recordemos que un array bidimensional en Java es en realidad un array de arrays). El objetivo del ejercicio es crear un método `longitudMedia` que acepte como parámetro un array desigual de enteros y que devuelva a la salida la longitud media de los distintos subarrays.

   Por ejemplo, el siguiente array tiene 5 subarrays de longitudes 3, 5, 2, 1 y 4 respectivamente. La longitud media de los subarrays por tanto es (3+5+2+1+4)/5 = 3

2	4	5		
4	8	9	6	3
2	3			
8				
9	7	5	3	

   Añadir tambien un método `hacerMatriz` que dado un array desigual devuelva una matriz cuadrada, rellenando los huecos con casillas inicializadas a cero, tal y como se muestra en el ejemplo.

2	4	5	0	0
4	8	9	6	3
2	3	0	0	0
8	0	0	0	0
9	7	5	3	0

   Tema: Creación, inicialización y manejo de Arrays multidimensionales

6. Crea una clase `Persona` que tenga un atributo `nombre` de tipo `String` y visibilidad pública. También tendrá un atributo `id` de tipo `long`, carácter `final` y de visibilidad privada. Este atributo representa un identificador único para cada persona, por lo que será generado en el constructor de la clase de forma secuencial (la primera persona creada tendrá el `id` 0, la siguiente el 1, etc.), para ello definiremos un atributo estático y privado denominado `siguienteId` que almacenará el siguiente `id` a ser usado (y que se irá incrementando a medida que se creen personas).

   Definir un método `getId` de visibilidad protegida que devuelva el valor del atributo `id`. Crear una subclase y comprobar que puede acceder sin problemas al método `getId`, pero que no puede acceder al atributo `siguienteId` (dejar este último código comentado para no romper la compilación del resto de ejercicios).

   Tema: Especificadores de visibilidad y atributos estáticos

7. Crea una clase `Temperatura` que tenga dos métodos estáticos `toCelsius` y `toFahrenheit`. Ambos métodos aceptarán un valor de tipo `double` como parámetro y devolverán otro valor de tipo `double`. El primer método convertirá temperaturas Fahrenheit en temperaturas Celsius, el segundo hará la transformación inversa. En la prueba de este ejercicio deberá comprobarse cómo los métodos estáticos pueden ser llamados usando el nombre de la clase Temperatura o una instancia de la misma. Los resultados de la prueba deben ser mostrados usando sólo dos cifras decimales.

   Tema: Métodos estáticos, manejo y formateo de valores flotantes
   De utilidad: Método `printf` de `System.out`

8. Crea una clase que represente a un círculo. El círculo debe tener tres propiedades, las coordenadas X e Y de su centro y su radio, todas ellas representadas por valores en coma flotante. Crea cuatro constructores para el círculo (sin parámetros, pasándole sólo las coordenadas X e Y, pasándole sólo el radio y pasándole el radio y las coordenadas X e Y). Los valores que no se le pasan al constructor se inicializan a cero. Crean en la clase círculo dos métodos de instancia que devuelvan (no por pantalla sino como tipo de retorno) el área (πr^2) y el perímetro ($2\pi r$) del círculo. Sobreescribe el método `equals` de la clase `Object` para que realiza una comparación de identidad basada en las tres propiedades del círculo.

   Tema: Inicialización y manejo de objetos. Metodos de instancia y método `equals`
   De utilidad: Haz que unos constructores se llamen a otros utilizando la palabra clave `this`. Utiliza la constante `PI` de la clase `Math`.

9. Crea un tipo enumerado `ColorPrimario` y otro `ColorSecundario`. Los colores secundarios pueden obtenerse como mezcla de los colores primarios. Los colores se encuentran en la siguiente tabla con sus respectivos valores:

COLORES PRIMARIOS	VALOR
Azul	1
Rojo	2
Amarillo	3
COLORES SECUNDARIOS	**VALOR**
Violeta	3
Verde	4
Naranja	5

Crear una clase `MezclarColores` que implemente un método que, a partir de dos colores primarios devuelva el color secundario asociado y muestre por pantalla un mensaje de la forma: "Color primario x" + "Color primario y" = "Color secundario z". El método debe aceptar y devolver tipos enumerados. El cálculo del color secundario se hará a través de los valores de los colores que deberán ser almacenados en el propio enumerado

Tema: Tipos enumerados

10. Escribe una clase `Corredor` que contenga un atributo de tipo entero denominado `energia` que indica el valor de la energía como un entero en el rango [0, 100]. Al crear un corredor se deberá indicar un valor para la energía. Si el valor estuviera fuera del rango [0, 100] se lanzaría la excepción `IllegalArgumentException`.

    El corredor tendrá un método `recargarEnergia` a la que se le pasará por parámetro la cantidad de energía que será sumada al atributo `energia`. De nuevo, si la cantidad pasada por parámetro no es un número entero mayor o igual a cero se lanzará la excepción `IllegalArgumentException`. Si la energía recargada pasa del valor 100 se deberá dejar a 100.

    La clase tendrá un método `correr` que cada vez que se llame restará 10 puntos de energía del corredor. Si la energía del corredor era menor que 10 entonces la energía quedará a cero y se lanzará la excepción personalizada `AgotadoException` indicando cuál era la energía del corredor antes de llamar al método `correr`.

    La clase también tendrá un método `entrenar` que permitirá al corredor recuperar energía. La primera vez que se llama la energía del corredor aumentará 50 puntos (teniendo en cuenta que nunca puede pasar de 100), la segunda vez aumentará 30 puntos, la tercera vez aumentará 10 puntos y el resto de veces que se llame al método el entrenamiento no variará la energía del corredor.

    Tema: Creación de clases y manejo de excepciones

Nota importante: En los métodos main de los ejercicios que manejan excepciones es necesario probar que las excepciones se lanzan correctamente, pero también es necesario capturar dichas excepciones para que no rompan la ejecución de los ejercicios. Si las excepciones no se capturan correctamente el ejercicio se dará como no válido.