UNIVERSIDADE DA CORUÑA

PROGRAMACIÓN ORIENTADA A OBJETOS. CURSO 2008-2009

Boletín de Ejercicios 1

NOTAS:

- Lee las instrucciones de los ejercicios antes de empezar a realizarlos.
- La fecha límite para la entrega del boletín es el **21 de Noviembre de 2008**

1. Crea una clase que tenga un método al que se le pasa el número correspondiente a un mes y el año al que pertenece y que devuelve el número de días que tiene dicho mes. Deberás tener en cuenta los años bisiestos para el mes de Febrero (ver apuntes sobre JUnit)

   Tema: Cláusula condicional switch

2. Escribiremos una función que simule la tirada repetida de un par de dados hasta que se obtenga un valor predeterminado en la suma de dichos dados. La función sería "`int tirarPara(int N)`" donde N es la cantidad que pretendemos obtener (como precondición podemos suponer que $2 \leq N \leq 12$) y el tipo de retorno representa el número de veces que hemos tenido que tirar los dos dados para obtener dicho valor N. Vamos a determinar ahora empíricamente cual es el número medio de tiradas que tenemos que hacer para obtener *snake eyes* (dos ases, cuya suma es dos). Para ello tiraremos el dado hasta obtener M veces *snake eyes* usando la función "`int snakeEyes(int M)`". El tipo de retorno será el número medio de tiradas que hemos necesitado (usar la función `tirarPara`). Como es complicado hacer una pruebas de métodos que dependen de la aleatoriedad, las pruebas JUnit para estos dos métodos simplemente incluirán sentencias que impriman en consola el resultado de la ejecución de los mismos (no se llamará a ningún método *assert*).

   Tema: Bucles while y for.

   De utilidad: Para simular los dados usar la función `random()` de la clase `Math`.

3. Define una clase que tenga los siguientes métodos estáticos para la conversión de temperaturas: CelsiusAFahrenheit y FahrenheitACelsius. Recuerda que la conversión entre ambas temperaturas se hace según la siguiente ecuación: ºF = (°C × 9/5) + 32 o, alternativamente, ºC = (ºF - 32) × (5/9). Incluye también un método estático formateaTemperatura que, dada una expresión en doble precisión (tipo double) devuelve un String formateado a un máximo de dos decimales

   Tema: Métodos estáticos y formateo de valores en punto flotante

   De utilidad: Método printf de la clase System.out

4. Calcula 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 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 número que queden sin marcar serán primos ¿Sería necesario llegar hasta mirar si los números son divisible por 100?

   Tema: Bucle for, Arrays

5. Crea una clase que tenga un método al cual se le pasa un array bidimensional de 9x9 y comprueba si el array pasado es una solución de sudoku (suponer como precondición que el tamaño del array es el correcto y que sólo contiene valores del 1 al 9). Una solución de sudoku es correcta si todas las filas, las columnas y los cuadrados internos 3x3 no contienen números repetidos. Por ejemplo una posible solución es:

```
+-------+-------+-------+
| 2 1 9 | 6 4 7 | 8 5 3 |
| 6 4 8 | 9 3 5 | 7 2 1 |
| 7 3 5 | 1 2 8 | 4 9 6 |
+-------+-------+-------+
| 3 2 1 | 8 6 9 | 5 7 4 |
| 9 5 4 | 3 7 2 | 6 1 8 |
| 8 7 6 | 4 5 1 | 9 3 2 |
+-------+-------+-------+
| 1 6 3 | 7 9 4 | 2 8 5 |
| 4 9 2 | 5 8 3 | 1 6 7 |
| 5 8 7 | 2 1 6 | 3 4 9 |
+-------+-------+-------+
```

   Tema: Bucles, Arrays bidimensionales

6. El formato de fecha ISO es un formato estándar que representa las fechas de la siguiente forma: “AAAA/MM/DD”. En dicho formato, por ejemplo, el “28 de julio de 2006” se representaría como 2006-07-28. Crea una clase que contenga los siguientes métodos: En primer lugar un método para convertir una fecha en formato habitual de "DD de MM de AAAA" a su representación en formato ISO. Para simplificar las cosas podemos suponer que el formato habitual que le vamos a pasar es correcto, aunque la fecha puede ser incorrecta, por ejemplo, podemos pasarle el 31 de Febrero. En segundo lugar crearemos un método en el que dada una fecha en formato ISO compruebe que es una fecha correcta. Por fecha correcta entendemos que no existan letras ni caracteres extraños en la fecha y que esta sea correcta, es decir, que se tenga en cuenta el número máximo de días en cada mes (ejercicio 1 de este boletín).

   Tema: Manejo de Strings
   De utilidad: Métodos indexOf, substring, trim, etc. de String

7. Crea una clase `Termometro` que almacene una temperatura en grados Kelvin. Provee un constructor para termómetro que permita pasarle una temperatura y la escala de dicha temperatura (Celsius, Fahrenheit o Kelvin) representada esta última mediante un tipo enumerado. La representación interna de la temperatura se hará siempre a través de grados Kelvin. El termómetro debe proveer de métodos de lectura y escritura de su valor de temperatura interna. Los métodos de lectura serán `getKelvin`, `getCelsius` y `getFahrenheit` que devolverán la temperatura en la escala que se le indica, los métodos de escritura serán `setKelvin`, `setCelsius` y `setFahrenheit` (aunque recordar que la temperatura interna se almacenará siempre en Kelvin). Crea también un método `equals` que permita comparar dos termómetros por su temperatura interna. Como el contrato de `equals` dice que si dos números son “equals” deben devolver el mismo hash code es necesario que redefinas también el método `hashCode` de `Object` (para ello puedes utilizar las facilidades de NetBeans pulsando sobre el icono contextual en forma de bombilla que aparece a la derecha del método y seleccionando “Generate missing hashCode()”). Os adjunto la tabla de conversión de temperaturas.

	Desde Kelvin	Hacia Kelvin
Celsius	[°C] = [K] − 273.15	[K] = [°C] + 273.15
Fahrenheit	[°F] = [K] × 9⁄5 − 459.67	[K] = ([°F] + 459.67) × 5⁄9

Tema: Encapsulación de objetos, métodos equals y hashCode.

De utilidad: Recuerda que no puedes comparar dos números en coma flotante directamente, habrá que restarlos y comprobar que el valor absoluto es menor que una determinada cantidad

8. Crea una clase `Carta` inmutable que incluya las características básicas de una carta (como el número y el palo). Crea una clase `Baraja` inmutable que se encargue de mantener una baraja de cartas. Entre las funciones de la baraja será la de incluir un método constructor que genere todas las cartas y un método que genere un objeto de la clase `Mazo`. Un mazo de cartas será prácticamente igual que una baraja pero no será inmutable, por lo que podrán incluirse o eliminarse cartas del mazo. Además el orden de las cartas al generarse el mazo siempre será aleatorio.

   Tema: Composición de objetos

9. Crea un tipo enumerado para representar las constantes de las monedas de Euro. Estas constantes deberán ser: EURO2, EURO1, CENT50, CENT20, CENT10, CENT5, CENT2, CENT1. Cada constante debe además almacenar el valor numérico que le pertenece y deberá incluir un método de lectura de dicho valor. Crea una clase monedero que permita contener un número indeterminado de monedas de euro. La clase monedero debe incluir métodos para introducir o extraer monedas del monedero y un método que devuelva la cantidad numérica total de euros almacenados en el monedero.

   Tema: Tipos enumerados, arrays como colecciones

10. Crea una clase `Dinero` que representa a una cantidad monetaria expresada en euros. Como no queremos usar una representación en punto flotante para el dinero (ni queremos usar clases como `BigDecimal` de Java) la clase `Dinero` tendrá dos campos enteros que representarán la cantidad en euros y la cantidad en céntimos del dinero que estamos representado. Como constructores incluiremos un constructor de copia, un constructor al que le pasamos los euros y los centimos, otro al que no le pasamos nada y lo inicializa a cero y alguno más que consideres adecuado. Los constructores se deben enlazar usando la sentencia `this`. Se pueden introducir euros como una cantidad negativa, pero no céntimos, por lo que el sistema deberá lanzar una excepción si se hace eso. También si el número de céntimos es mayor o igual a 100 la cantidad almacenada debe ajustarse correctamente (5 euros más 120 céntimos son 6 euros y 20 céntimos). La clase debe sobreescribir los métodos `equals` y `toString` (este último debe representar la cantidad correctamente formateada como por ejemplo "5,02€"). También debe incluir métodos de lectura de los atributos y métodos para sumar dos clases `Dinero` y para restar dos clases `Dinero` (recuerda que hay que ajustar el valor de los centimos si son mayores o iguales a 100). En este caso debes decidir qué estrategia te parece más correcta (por ejempo, que sean métodos estáticos, que sean métodos que sólo aceptan un parámetro y devuelven el resultado de sumar dicho parámetro al objeto actual, o que sean métodos que modifican el objeto actual al realizar la operación). ¿Debería ser `Dinero` una clase inmutable?

    Tema: Clases, Encapsulación de atributos, constructores, método `equals`, método `toString`