EJERCICIOS PROGRAMACION III
Práctica SEMANA 3

CONTENIDOS: Wrapper Classes y primeros ejercicios de POO

Ejercicios obligatorios:

1. Deberás crear una clase Person.java con atributos nombre altura y peso, sus correspondientes getters y setters. En el método main deberás instanciar 3 objetos de dicha clase y completar su información solicitándola al usuario por consola para cada objeto. Finalmente, deberás mostrar por consola quién es el más alto y quién es el que más pesa. Deberás utilizar siempre los *getters* y *setters* para obtener la información y establecerla en cada instancia de cada objeto.
2. Mejora el ejercicio 1 para que exista un constructor sin parámetros que inicialice los valores de nombre, altura y peso a valores por defecto que estimes oportunos.
3. Mejora el ejercicio 1 de tal forma que la clase Person ofrezca un método que calcule el IMC de la persona a través de los valores actuales de los atributos del objeto (su estado). Se deberá contemplar la posibilidad de que los valores no se hayan inicializado a valores válidos.
4. Analiza el flujo de ejecución del programa colocando puntos de interrupción en el constructor y en las distintas partes del código del ejercicio 1. Establece valores a los atributos directamente en su declaración y establece un punto de parada para comprobar qué se ejecuta antes: el constructor o la declaración del atributo.

Ejercicios de extensión:

5. Empleando los métodos parseXXX de las clases de envoltorio, construir una clase dotada de métodos adecuados para leer todos los tipos de datos numéricos (int, float, double) en forma de cadenas. Si el usuario no inserta un valor correcto, pedir de nuevo el valor. (Largo). Ver biblioteca.jar y en particular Esdia.java
6. Modifica el ejercicio 1 para que la clase Person.java esté dentro de un paquete llamado model en lugar del mismo paquete que la clase App.java. ¿Que has necesitado cambiar para poder utilizarla en App.java?
7. ¿Qué diferencia existe entre el constructor por defecto de una clase en Java y un constructor sin parámetros?
8. ¿Por qué utilizar getters y setters y no acceder directamente a los atributos a través del punto? Encapsulación. Chapter: The meaning of encapsulation. ¿Es una cuestión de seguridad o de diseño?

9. ¿Qué diferencia hay entre definir un atributo como public, private o no poner modificador de visibilidad? ¿En qué caso es posible acceder a ellos mediante el punto (.) con una referencia de un objeto?
10. ¿Para qué se utiliza la keyword **this**? ¿Y la keyword **new**?
11. Es posible definir clases que a su vez tengan atributos de otras clases (denominado **Composición** y como ya sucede en el ejercicio 1 con la clase String que en Java es un tipo de referencia). Debes ahora modificar el ejercicio 1 para que la clase Person tenga un atributo de tipo DNI (otra clase que deberéis crear) que tendrá como atributos número y letra. Al constructor de la clase Person se le deberá pasar como parámetro un número de DNI y se creará un objeto de tipo DNI, generando la letra correspondiente a partir del algoritmo para obtener la letra del DNI a partir del número. Consulta el operador `%` (módulo, resto tras la división entera) en Java.
12. Cree una clase llamada Rectangulo.java con los atributos longitud y anchura, cada uno con un valor predeterminado de 1. Debe tener métodos para calcular el perímetro y el área del rectángulo. Debe tener métodos getter y setter para longitud y anchura. Los métodos setter deben verificar que longitud y anchura sean números de punto flotante mayores de 0.0, y menores de 20.0. Escriba un programa para probar la clase Rectangulo.
13. Cree un nuevo proyecto con la clase Person.java que creó inicialmente. Realice las siguientes instrucciones y responda a las preguntas planteadas.

```
    public static void main(String[] args) {

        // Creación de un objeto de la clase Person

        Person paco = new Person("Paco",22,177.3f,82.3f);

        // Copiamos la referencia del objeto anterior a otra referencia

        Person refPaco2 = paco;

        // Utilizamos la segunda referencia para establecer un valor en el objeto al que
hace referencia

        refPaco2.setAgeYears(77);

        // ¿Resultado?

        System.out.printf("La edad de %s es:%d%n",paco.getName(),paco.getAgeYears());

// ¿Cuál es la edad que se muestra?
//  ¿Cuántos objetos hay en memoria? ¿Cuántas referencias? Revisa el Debugger

    }
```