IES ILIBERIS
Avda. Andalucía
Atarfe - Granada

Departamento de Informática
1º DAW
Programación

RELACIÓN DE EJERCICIOS 11 - Herencia en Java, clases abstractas e interfaces

Resolver los siguientes problemas escribiendo el algoritmo con lenguaje Java.

Crear en GitHub un repositorio llamado **UD6-Herencia en Java** para subir cada uno de los ejercicios de la relación.

1. Se desea almacenar cierta información sobre personas. En concreto, interesa saber:
    a. NIF: Número de Identificación Fiscal
    b. Altura: Expresada en centímetros
    c. Edad: Expresada en años
    d. Por defecto, una persona para la que se no especifique dicha información se asume que tendrá 25 años, NIF: 11111111A y 175 cm de altura

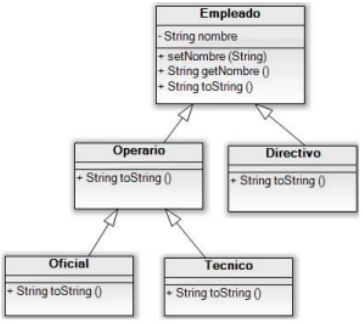
    Diseña la clase Persona con sus atributos, constructores, getters y setters. Diseña la clase TestPersona que instancie dos personas, una con la información por defecto y otra con la que tú mismo inventes, y que muestre la información por pantalla.

2. Utilizando la clase persona del ejercicio anterior y teniendo en cuenta que:
    a. Una persona puede hablar y comer.
    b. Un ingeniero, además, puede razonar y trabajarEnGrupo
    c. Un ingeniero informático, además, puede crearPrograma

    Diseña una jerarquía de clases apropiada para diferenciar entre una Persona, un Ingeniero y un IngenieroInformatico. Modifica la clase TestPersona del ejercicio anterior para probar la jerarquía creada.

3. Crea utilizando herencia las clases: **Yogur** y **YogurDesnatado**. En primer lugar, diseña la clase Yogur sabiendo que un yogur siempre tiene 120.5 calorías. Se requiere implementar un método consultor para obtener sus calorías. A continuación, diseña la clase YogurDesnatado sabiendo que siempre tiene la mitad de calorías que un Yogur normal. Finalmente, construye una clase TestYogures con un método principal que cree un objeto Yogur y un YogurDesnatado y muestre sus calorías

4. Implementa la siguiente estructura de clases:



La clase base es la clase Empleado. Esta clase contiene:

- Un atributo privado **nombre** de tipo String que heredan el resto de clases.
- Un constructor por defecto.
- Un constructor con parámetros que inicializa el nombre con el String que recibe.
- Método set y get para el atributo nombre.
- Un método toString() que devuelve el String: "Empleado " + nombre.

El resto de clases solo deben sobrescribir el método **toString()** en cada una de ellas y declarar el constructor adecuado de forma que cuando la ejecución de las siguientes instrucciones:

RELACIÓN DE EJERCICIOS 11 - Herencia en Java, clases abstractas e interfaces

INSTRUCCIONES	SALIDA
Empleado E1 = new Empleado("Rafa"); Directivo D1 = new Directivo("Mario"); Operario OP1 = new Operario("Alfonso"); Oficial OF1 = new Oficial("Luis"); Tecnico T1 = new Tecnico("Pablo"); System.out.println(E1); System.out.println(D1); System.out.println(OP1); System.out.println(OF1); System.out.println(T1);	Empleado Rafa Empleado Mario -> Directivo Empleado Alfonso -> Operario Empleado Luis -> Operario -> Oficial Empleado Pablo -> Operario -> Técnico

5. (Clases Abstractas). Modificar el ejercicio anterior incorporando **clases abstractas** donde sea necesario suponiendo que todo empleado de la empresa debe estar categorizado en alguna de las subclases. También se deben modificar los métodos necesarios para conseguir la funcionalidad del problema.

6. (Clases Abstractas). Basándote en la jerarquía de clases **Persona**, **Alumno**, **Profesor** explicada en los apuntes, crea un método abstracto llamado **mostrar** para la clase **Persona**. Dependiendo del tipo de persona (alumno o profesor) el método **mostrar** tendrá que mostrar unos u otros datos personales (habrá que hacer implementaciones específicas en cada clase derivada). Una vez hecho esto, implementa completamente las tres clases (con todos sus atributos y métodos) y utilízalas en un pequeño programa de ejemplo que cree un objeto de tipo Alumno y otro de tipo Profesor, los rellene con información y muestre esa información en la pantalla a través del método **mostrar**.

7. (Clases Abstractas). Crea la clase **Vehiculo** (de la que no se podrán crear instancias), así como las clases **Bicicleta** y **Coche** como subclases de la primera. Para la clase **Vehiculo**, crea los atributos de clase **vehiculosCreados** y **kilometrosTotales**, así como el atributo de instancia **kilometrosRecorridos**. Crea también algún método específico para cada una de las subclases. Prueba las clases creadas en una clase denominada **testVehiculo** mediante un programa con un menú como el que se muestra a continuación:

```
VEHÍCULOS
=========
1. Anda con la bicicleta
2. Haz el caballito con la bicicleta
3. Anda con el coche
4. Quema rueda con el coche
5. Ver kilometraje de la bicicleta
6. Ver kilometraje del coche
7. Ver kilometraje total
8. Salir
Elige una opción (1-8):
```

8. (Interfaces). Se quiere informatizar una biblioteca. Crea las clases **Publicacion**, **Libro** y **Revista**. Las clases deben estar implementadas con la jerarquía correcta. Las características comunes de las revistas y de los libros son el código **ISBN**, el **título**, y el **año de publicación**. Los libros tienen además un atributo **prestado**. Cuando se crean los libros, no están prestados. La clase **Libro** debe

implementar la **interfaz Prestable** que tiene los métodos **presta**, **devuelve** y **estaPrestado**. Las revistas tienen un **número**.

Programa principal:

```
Libro libro1 = new Libro("123456", "La Ruta Prohibida", 2007);
Libro libro2 = new Libro("112233", "Los Otros", 2016);
Libro libro3 = new Libro("456789", "La rosa del mundo", 1995);
Revista revista1 = new Revista("444555", "Año Cero", 2019, 344);
Revista revista2 = new Revista("002244", "National Geographic", 2003, 255);
System.out.println(libro1);
System.out.println(libro2);
System.out.println(libro3);
System.out.println(revista1);
System.out.println(revista2);
libro2.presta();
if (libro2.estaPrestado()) {
  System.out.println("El libro está prestado");
}
libro2.presta();
libro2.devuelve();
if (libro2.estaPrestado()) {
  System.out.println("El libro está prestado");
}
libro3.presta();
System.out.println(libro2);
System.out.println(libro3);
```

Salida:

```
ISBN: 123456, título: La Ruta Prohibida, año de publicación: 2007 (no prestado)
ISBN: 112233, título: Los Otros, año de publicación: 2016 (no prestado)
ISBN: 456789, título: La rosa del mundo, año de publicación: 1995 (no prestado)
ISBN: 444555, título: Año Cero, año de publicación: 2019
ISBN: 002244, título: National Geographic, año de publicación: 2003
El libro está prestado
Lo siento, ese libro ya está prestado.
ISBN: 112233, título: Los Otros, año de publicación: 2016 (no prestado)
ISBN: 456789, título: La rosa del mundo, año de publicación: 1995 (prestado)
```

9. (Interfaces). Basándote en la jerarquía de clases del ejercicio 7, crear una interfaz llamada **Arrancable** que proporcione los métodos típicos de objetos que tengan un motor (no necesariamente vehículos), como por ejemplo **arrancar**, **detener** y **estaArrancado**. Deberás implementar esta interfaz en la clase **Coche** ya que los objetos de esta clase sí son arrancables. Modifica el menú de opciones del ejercicio 7 para que incluya nuevas opciones para arrancar el coche, etc. Hacer las comprobaciones necesarias para por ejemplo, no arrancar un coche que ya está parado, o no andar con el coche si éste no está arrancado.