



APRENDERAPROGRAMAR.COM

EJERCICIO RESUELTO
EJEMPLO DE HERENCIA
SIMPLE EN JAVA. USO DE
EXTENDS Y SUPER.
CONSTRUCTORES CON
HERENCIA. (CU00687B)

Sección: Cursos

Categoría: Curso “Aprender programación Java desde cero”

Fecha revisión: 2029

Resumen: Entrega nº87 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

EJERCICIO RESUELTO DE HERENCIA SIMPLE EN JAVA

En apartados anteriores del curso ya hemos comenzado con el estudio de la herencia en programación orientada a objetos. La herencia puede comprender numerosas clases, es decir, una clase puede heredar de otra clase que a su vez herede de otra clase. Pensemos ahora que un Profesor hereda de Persona, y que a su vez pueda ser ProfesorInterino o ProfesorTitular.



Trata de escribir el código de las clases ProfesorInterino y ProfesorTitular, de forma que hereden de profesor y que a su vez cada una tenga sus propios métodos. Una vez lo hayas hecho, escribe este código de ejemplo, la clase ProfesorInterino y una clase de test TestHerencia2 y estudia cómo hemos desarrollado la herencia.

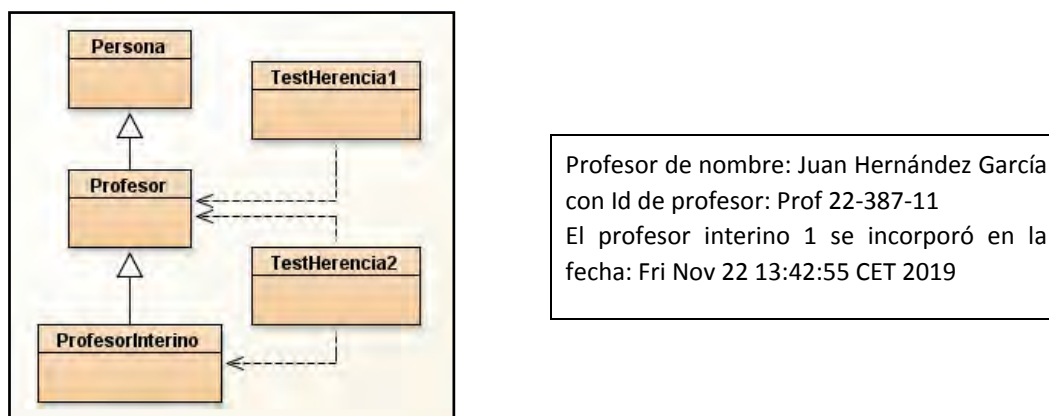
```
//Código de la clase ProfesorInterino ejemplo aprenderaprogramar.com
import java.util.Calendar;
public class ProfesorInterino extends Profesor {
    private Calendar fechaComienzoInterinidad;
    public ProfesorInterino(Calendar fechaInicioInterinidad) {
        super();
        fechaComienzoInterinidad = fechaInicioInterinidad;}

    public ProfesorInterino(String nombre, String apellidos, int edad, Calendar fechaInicioInterinidad) {
        super(nombre, apellidos, edad);
        fechaComienzoInterinidad = fechaInicioInterinidad;}

    public Calendar getFechaComienzoInterinidad () { return fechaComienzoInterinidad;}
} //Cierre de la clase
```

```
import java.util.Calendar;
public class TestHerencia2 {
    public static void main (String [ ] Args) {
        Profesor profesor1 = new Profesor ("Juan", "Hernández García", 33);
        profesor1.setIdProfesor("Prof 22-387-11");
        profesor1.mostrarNombreApellidosYCarnet();
        Calendar fecha1 = Calendar.getInstance();
        fecha1.set(2019,10,22); //Los meses van de 0 a 11, luego 10 representa noviembre
        ProfesorInterino interino1 = new ProfesorInterino("José Luis", "Morales Pérez", 54, fecha1);
        System.out.println("El profesor interino 1 se incorporó en la fecha: " + fecha1.getTime().toString() );}
} //Cierre de la clase ejemplo aprenderaprogramar.com
```

El diagrama de clases y el resultado del test son:



Los aspectos a destacar del código son:

- a) Hemos usado la clase Calendar del API de Java. La importamos, declaramos objetos de tipo Calendar y usamos algunos de sus métodos. No vamos a profundizar en esta clase porque no es ese nuestro objetivo: en este código es sólo una clase auxiliar para desarrollar este ejemplo.
- b) La clase ProfesorInterino tiene dos constructores y ambos invocan a la superclase Profesor mediante la palabra clave *super*. La superclase habrá de tener también dos constructores, ya que en caso contrario saltará un error.
- c) En el test creamos un ProfesorInterino pasando 4 parámetros al constructor: tres de esos parámetros son gestionados por el constructor de la superclase, y el restante es gestionado por el constructor de la subclase.

EJERCICIO

Se plantea desarrollar un programa Java que permita la gestión de una empresa agroalimentaria que trabaja con tres tipos de productos: productos frescos, productos refrigerados y productos congelados. Todos los productos llevan esta información común: fecha de caducidad y número de lote. A su vez, cada tipo de producto lleva alguna información específica. Los productos frescos deben llevar la fecha de envasado y el país de origen. Los productos refrigerados deben llevar el código del organismo de supervisión alimentaria, la fecha de envasado, la temperatura de mantenimiento recomendada y el país de origen. Los productos congelados deben llevar la fecha de envasado, el país de origen y la temperatura de mantenimiento recomendada.

Hay tres tipos de productos congelados: congelados por aire, congelados por agua y congelados por nitrógeno. Los productos congelados por aire deben llevar la información de la composición del aire con que fue congelado (% de nitrógeno, % de oxígeno, % de dióxido de carbono y % de vapor de agua). Los productos congelados por agua deben llevar la información de la salinidad del agua con que se realizó la congelación en gramos de sal por litro de agua. Los productos congelados por nitrógeno deben llevar la información del método de congelación empleado y del tiempo de exposición al nitrógeno expresada en segundos.

Crear el código de las clases Java implementando una relación de herencia siguiendo estas indicaciones:

a) En primer lugar realizar un esquema con papel y bolígrafo donde se represente cómo se van a organizar las clases cuando escribamos el código. Estudiar los atributos de las clases y trasladar a la superclase todo atributo que pueda ser trasladado.

b) Crear superclases intermedias (aunque no se correspondan con la descripción dada de la empresa) para agrupar atributos y métodos cuando sea posible. Esto corresponde a “realizar abstracciones” en el ámbito de la programación, que pueden o no corresponderse con el mundo real.

c) Cada clase debe disponer de constructor y permitir establecer (set) y recuperar (get) el valor de sus atributos y tener un método que permita mostrar la información del objeto cuando sea procedente.

Crear una clase testHerencia3 con el método main donde se creen: dos productos frescos, tres productos refrigerados y cinco productos congelados (2 de ellos congelados por agua, otros 2 por agua y 1 por nitrógeno). Mostrar la información de cada producto por pantalla.

Puedes comprobar si tu respuesta es correcta consultando en los foros [aprenderaprogramar.com](http://www.aprenderaprogramar.com).

Próxima entrega: CU00688B

Acceso al curso completo en [aprenderaprogramar.com](http://www.aprenderaprogramar.com) -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188