EXAMEN PARCIAL ESTRUCTURA DE DATOS Y ALGORITMOS
25 FEBRERO 2013

Nombre y Apellidos: **Grupo:**

Problema 2 (5 puntos)

Cocinilla S.A, una empresa de electrodomésticos para el hogar, quiere desarrollar tres nuevos productos:

- BatiCocinilla: una batidora que además de batir, permite mezclar y montar.
- PicaCocinilla: una picadora que pica y muele.
- ThermoCocinilla: un robot de cocina que incorpora las funcionalidades de la batidora y la picadora (aunque su implementación es distinta porqueutilizanlos últimos avances tecnológicos en robots de cocina). ThermoCocinilla también deberá permitir amasar, pesar y calentar. Para calentar un alimento siempre es necesario indicar el tiempo en segundos. Además se quiere desarrollar dos modelos distintos: el básico y el superCocinilla. Este último mantiene las mismas opciones que el modelo básico pero mejora la precisión de la función de pesado.

Cualquiera de estos electrodomésticos tiene asociado una potencia, una tensión (ambos pueden ser valores enteros) y un estado que indique si el electrodoméstico está encendido o apagado. Además, todos los electrodomésticos deben poder encenderse y apagarse (ambas funcionalidades se implementan de la misma forma en todos los electrodomésticos).

Se pide:

- Crea las interfaces y clases necesarias para representar la situación descrita. **La solución propuesta debe aplicarlas bases del paradigma de POO para conseguir sw robusto, adaptable y reutilizable. Razona cada una de tus decisiones en relación con la elección de interfaces, clases abstractas y clases no abstractas.**

Nota: Para simular cada una de las funcionalidades, bastará con mostrar por pantalla un texto que indique el tipo de operación que se realiza. Por ejemplo:

System.out.println("apagando…").

Cuando sea necesario distinguir la funcionalidad de algún electrodoméstico se deberá especificar en el mensaje el nombre del electrodoméstico. Por ejemplo, el enunciado indica que el robot debe picar, sin embargo lo hará de forma distinta a una picadora. Para distinguir la implementación bastará con escribir el mensaje:

System.out.prinltn("ThermoCocinilla picando…");

Nota: No es necesario escribir el código de los constructores ni de métodos getters y setters.

En primer lugar, creamos la clase Electrodoméstico en la que se representarán los atributos y métodos comunes a todos los electrodomésticos:

```
package cocinilla;

public class Electrodomestico {
    boolean estado;//true si está encendido y false si está apagado.
    int potencia;
    int tension;

    public void apagar() {
        System.out.println("apagando");
        estado=false;
    }

    public void encender() {
        System.out.println("encender");
        estado=false;
    }

}
```

Figura 1 Clase Electrodoméstico

Como los electrodomésticos comparten algunas de las funcionalidades aunque con implementaciones distintas, una buena decisión de diseño siguiendo las recomendaciones de POO es utilizar interfaces:

```
package cocinilla;

public interface IBatidora {
    void batir();
    void mezclar();
    void montar();
}
```

Figura 2 Interfaz IBatidora define las operaciones que debe proporcionar una batidora

```
package cocinilla;

public interface IPicadora {
    void picar();
    void moler();
}
```

Figura 3 Interfaz IPicadora las operaciones que debe proporcionar una picadora

Ya podemos definir las clases BatiCocinilla y PicaCocinilla que implementen respetivamente las interfaces anteriores y además heredan de la clase Electrodoméstico:

```java
package cocinilla;

public class BatiCocinilla extends Electrodomestico implements IBatidora {

    @Override
    public void batir() {
        System.out.println("batiendo...");
    }

    @Override
    public void mezclar() {
        System.out.println("mezclando...");
    }

    @Override
    public void montar() {
        System.out.println("montando...");
    }



}
```

Figura 4 Clase BatiCocinilla implementa la interfaz IBatidora y hereda de la clase Electrodoméstico

```java
package cocinilla;

public class PicaCocinilla extends Electrodomestico implements IPicadora {

    @Override
    public void picar() {
        System.out.println("picando...");

    }

    @Override
    public void moler() {
        System.out.println("moliendo...");

    }

}
```

Figura 5 Clase PicaCocinilla implementa la interfaz IPicadora y hereda de la clase Electrodoméstico

Finalmente debemos crear la clase para representar la ThermoCocinilla. En este caso, es posible distinguir entre dos modelos: básico y super. Como dice el enunciado la única diferencia entre ambos modelos es que la SuperCocinilla mejora la función de pesado (es decir, debemos implementarlo como que su implementación de dicha funcionalidad es distinta).

```java
package cocinilla;

public class ThermoCocinillaBasico extends Electrodomestico implements IBatidora, IPicadora {
    @Override
    public void picar() {
        System.out.println("picando thermoCocinilla...");
    }
    @Override
    public void moler() {
        System.out.println("moliendo thermoCocinilla...");

    }
    @Override
    public void batir() {
        System.out.println("batiendo thermoCocinilla...");
    }
    @Override
    public void mezclar() {
        System.out.println("mezclando thermoCocinilla...");
    }
    @Override
    public void montar() {
        System.out.println("picando thermoCocinilla...");
    }
    //M@todos nuevos
    public void amasar() {
        System.out.println("amasando thermoCocinilla...");
    }
    public void pesar() {
        System.out.println("pesando thermoCocinilla...");
    }
    public void calentar(int seg) {
        System.out.println("calentando thermoCocinilla durante " + seg);
    }
}
```

Figura 6 Clase ThermoCocinillaBasico implementa ambos interfaces y hereda de la clase Electrodoméstico.

```java
public class ThermoCocinillaSuper extends ThermoCocinillaBasico{
    //reescribrimos el método heredado
    public void pesar() {
        System.out.println("pensado con ThermoCocinillaSuper (más precisión)...");
    }

}
```

Figura 7 Ejemplo de Polimorfismo: la clase ThermoCocinillaSuper es hija de la clase ThermoCocinillaBasico porque realiza las mismas operaciones y mejora (reescribe) la función de pesado.