Universidad Carlos III de Madrid

Programación de Sistemas
Grado en Ingeniería de Sistemas de Comunicaciones y Grado en Ingeniería Telemática

Leganés, 21 de marzo de 2014 — Examen parcial 1 (problemas)
Duración de la prueba: 75 min — Puntuación: 5 puntos sobre 10 del examen

Problema 1 (2 puntos)

Se tiene la siguiente interfaz que representa una propiedad inmobiliaria. Dispone de dos métodos, uno que devuelve el área de la propiedad en metros cuadrados y otro que devuelve su precio en euros:

```
public interface Estate {
    double area();
    double price();
}
```

Apartado 1 (0,75 puntos)

Indica qué debe aparecer en cada hueco de la clase siguiente, que implementa la interfaz *Estate*. Cada hueco puede contener uno o más términos. Esta clase programa el método que calcula el precio (como el producto del área por el precio por metro cuadrado) pero deja sin programar el método que calcula el área:

```
public _______(1)_______ class BaseEstate _______(2)_______ {
    private double eurosPerSquaredMeter;

    public BaseEstate(double eurosPerSquaredMeter) {
        this.eurosPerSquaredMeter = eurosPerSquaredMeter;
    }

    public double price() {
        _______(3)_______
    }
}
```

Apartado 2 (1,25 puntos)

Programa la clase *RectangularEstate*, que representa una propiedad de planta rectangular. Esta clase hereda de *BaseEstate* y debe ser instanciable. Además, debe proporcionar un constructor que reciba la longitud en metros de cada lado y el precio por metro cuadrado.

Problema 2 (1,5 puntos)

Las etiquetas *JLabel* permiten mostrar un texto, una imagen o ambos. **Programa** una clase llamada *RestrictedJLabel* que tenga exactamente el mismo comportamiento que *JLabel*, pero sin la capacidad de mostrar imágenes. Para ello:

- No debe proporcionar ninguno de los constructores de *JLabel* que reciben una imagen. Por tanto, debe proporcionar únicamente tres constructores: uno que no reciba parámetros, otro que reciba una cadena de texto con el texto de la etiqueta, y otro que reciba dicha cadena de texto y un entero que indique la alineación horizontal. Cada uno de los tres invocará al constructor de *JLabel* equivalente (el que recibe el mismo número y tipo de parámetros).
- Debe disponer de un método *void setIcon(Icon icon)* que, en vez de hacer lo que hace el método equivalente en *JLabel*, no haga nada.
- El resto de sus métodos deben ser exactamente los mismos que los de *JLabel*.

Tu solución debe ser compatible con la siguiente asignación:

```
JLabel label = new RestrictedJLabel("This is a label");
```

Problema 3 (1,5 puntos)

Programa una clase que herede de *JFrame*. El título de la ventana debe ser "Random frame". Debe mostrar mediante un *FlowLayout* un botón y una etiqueta. En la etiqueta se verá inicialmente un número aleatorio. Cada vez que el usuario pulse el botón debe cambiarse el texto de la etiqueta para que muestre un nuevo número aleatorio.

Recuerda que por defecto el contenedor principal de un objeto *JFrame* no está gestionado con *FlowLayout*.

Para generar números aleatorios puedes añadir el siguiente método a tu clase:

```
private String randomNumber() {
    return Integer.toString((new Random()).nextInt(1000));
}
```

Problema 1

Apartado 1

1. `abstract`
2. `implements Estate`
3. `return area() * eurosPerSquaredMeter;`

La clase debe ser abstracta porque no implementa el método *area*. Pese a ello, sí se puede utilizar *area()* para calcular el precio de la propiedad. El código completo de la clase sería:

```
public abstract class BaseEstate implements Estate {
    private double eurosPerSquaredMeter;

    public BaseEstate(double eurosPerSquaredMeter) {
        this.eurosPerSquaredMeter = eurosPerSquaredMeter;
    }

    public double price() {
        return area() * eurosPerSquaredMeter;
    }
}
```

Apartado 2

```
public class RectangularEstate extends BaseEstate {
    private double side1;
    private double side2;

    public RectangularEstate(double side1, double side2,
                             double eurosPerSquaredMeter) {
        super(eurosPerSquaredMeter);
        this.side1 = side1;
        this.side2 = side2;
    }

    public double area() {
        return side1 * side2;
    }
}
```

Para ser instanciable, esta clase está obligada a implementar el método *area*. Los lados del rectángulo deben ser declarados como atributos para poder ser utilizados desde dicho método. Además, debe invocarse al constructor de la superclase pasando el precio por metro cuadrado.

Problema 2

```
import javax.swing.JLabel;
import javax.swing.Icon;

public class RestrictedJLabel extends JLabel {

    public RestrictedJLabel() {
        super();
    }

    public RestrictedJLabel(String text) {
        super(text);
    }

    public RestrictedJLabel(String text, int horizontalAlignment) {
        super(text, horizontalAlignment);
    }

    public void setIcon(Icon icon) {
        // do nothing
    }
}
```

No se tendrán en cuenta los *import* en la corrección del problema.

Problema 3

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;

public class RandomFrame extends JFrame implements ActionListener {
    private JLabel label;
    public RandomFrame() {
        super("Random frame");
        JButton button = new JButton("Press me");
        button.addActionListener(this);
        label = new JLabel(randomNumber());
        getContentPane().setLayout(new FlowLayout());
        getContentPane().add(button);
        getContentPane().add(label);
        pack();
        setVisible(true);
    }

    private String randomNumber() {
        return Integer.toString((new Random()).nextInt(1000));
    }

    public void actionPerformed(ActionEvent e) {
        label.setText(randomNumber());
    }
}
```

No se tendrán en cuenta los *import* ni el uso de *pack()* en la corrección del apartado.