Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Informe práctica 1 parte 1

Diseño arquitectónico y Patrones

Francesco Marelli
(alu0101161730@ull.edu.es)

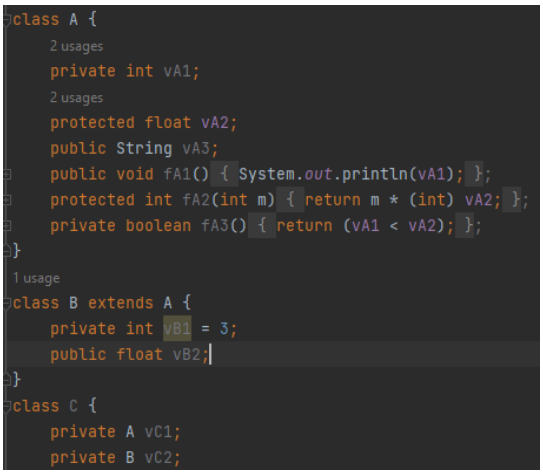
Índice:

1. Introducción.

El objetivo de esta práctica es introducir el entorno de desarrollo IntelliJ IDEA (versión community) y hacer un repaso sobre conceptos de Programación Orientada a Objetos utilizando el lenguaje de programación Java. Se hará especial hincapié en el uso de polimorfismo.

2. Parte 1.

2.1. Pregunta 1



```
class A {
    2 usages
    private int vA1;
    2 usages
    protected float vA2;
    public String vA3;
    public void fA1() { System.out.println(vA1); };
    protected int fA2(int m) { return m * (int) vA2; };
    private boolean fA3() { return (vA1 < vA2); };
}
1 usage
class B extends A {
    private int vB1 = 3;
    public float vB2;
}
class C {
    private A vC1;
    private B vC2;
```

¿Podríamos añadir el siguiente método a la definición de la clase C? public float fC4() { return vC1.vA2; }

No, porqué **vC1** apunta a null, no está inicializado. Habría que inicializarlo y se podría ya que aunque C no herede de ninguna de ellas, está en el mismo paquete.

2.2. Pregunta 2

¿Podríamos añadir el siguiente método a la definición de la clase B? public String fB1() { return vA3; }

Si se podría porqué B hereda de A además que es un atributo público, si fuese protected también se podría.

2.3. Pregunta 3

¿Podríamos añadir el siguiente método a la definición de la clase B? public float fB2() { return vA2; }
Por supuesto. Es un método de la clase B que llama a un atributo público de su propia clase. Se podría en todo caso (private, protected)

2.4. Pregunta 4

¿Podríamos añadir el siguiente método a la definición de la clase B? public int fB3() { return vA1; }
No, porque vA1 es un atributo privado de A, la clase padre de B, por lo que es accesible solo por dentro de la clase A.

2.5. Pregunta 5

¿Podríamos añadir el siguiente método a la definición de la clase C? public int fC5(int a) { return vC2.fA2(a); }
Solo si inicializamos correctamente en el constructor de C el objeto de la clase B que es vC2, para evitar errores en tiempo de ejecución (tiene valor null) En el método fC5 de la clase C se intenta llamar a fA2 de la instancia vC2 que es tipo B, que hereda de A. Dado que fA2 en A está protegido, se puede.

2.6. Pregunta 6

¿Podríamos añadir el siguiente método a la definición de la clase B? public int fB4(int a) { return fA2(a); }
Si por qué estamos llamando a la función protected fA2 de la clase A desde la clase B, que hereda de A.

2.7. Pregunta 7

¿Podríamos añadir el siguiente método a la definición de la clase B? public boolean fB5() { return fA3 (); }
No porqué la función fA3 de la clase A tiene modificador private, por ende se puede acceder a esta solo desde dentro de la clase A.

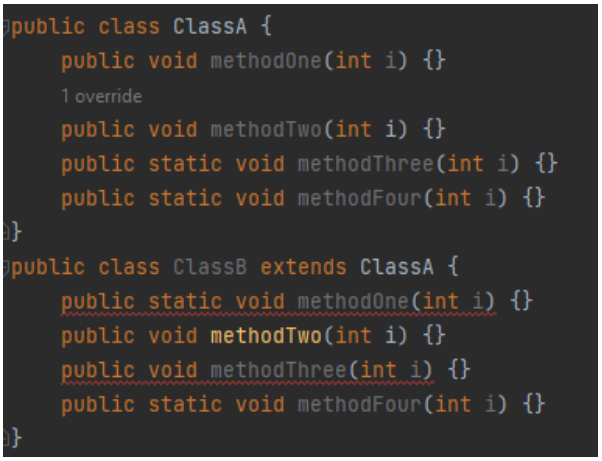
2.8. Pregunta 8

¿Podríamos añadir el siguiente método a la definición de la clase B? public B fB6() { return this; }
Si se puede. Retorna la misma instancia que está invocando el método.

2.9. Pregunta 9

¿Podríamos añadir el siguiente método a la definición de la clase B? public A fB6() { return super; }
No. super se utiliza para acceder a miembros de la clase padre desde la subclase pero no como valor de retorno, por ejemplo llamar al constructor de la superclase desde la clase hija. No es una instancia que puedo retornar.

2.10. Pregunta 10



```
public class ClassA {
    public void methodOne(int i) {}
    1 override
    public void methodTwo(int i) {}
    public static void methodThree(int i) {}
    public static void methodFour(int i) {}
}
public class ClassB extends ClassA {
    public static void methodOne(int i) {}
    public void methodTwo(int i) {}
    public void methodThree(int i) {}
    public static void methodFour(int i) {}
}
```

¿Qué método sobrescribe un método en la superclase?
methodTwo() de la clase hija sobreescribe el método **methodTwo()** de la clase padre porque tiene la misma firma (el mismo nombre del método y el tipo de parámetros y mismo tipo de retorno). Además es no estático, es decir, de instancia.

¿Qué método oculta un método en la superclase?
methodOne() porque tienen el mismo nombre pero modificador de acceso más restrictivo. El método padre tiene acceso público mientras que el método hijo tiene acceso estático.

¿Qué hacen los otros dos métodos?

methodFour() y **methodThree()** porqué en el caso del **methodFour()** son ambos estáticos y en el caso del **methodThree()** tienen misma firma y modificador de acceso más permisivo.

2.11. Pregunta 11

Indique cual sería la salida tras ejecutar el siguiente código

```
public class BaseClass
{
   public void methodToOverride() //Base class method
   {
       System.out.println ("I'm the method of BaseClass");
   }
}

public class DerivedClass extends BaseClass
{
   public void methodToOverride() //Derived Class method
   {
       System.out.println ("I'm the method of DerivedClass");
   }
}

public class TestMethod
{
    public static void main (String args []) {
          BaseClass obj1 = new BaseClass();
          BaseClass obj2 = new DerivedClass();
           obj1.methodToOverride();
           obj2.methodToOverride();
}
}
```

Output:
I'm the method of BaseClass
I'm the method of DerivedClass

2.12. Pregunta 12

¿Cuál sería la salida del siguiente código?

```
abstract public class PuebaAbstract{

        public void miMetodo(){
                System.out.println("Hola");
        }
        abstract public void otroMetodo();
}

public class PruebaConcreta extends PuebaAbstract{

        public void otroMetodo() {
                System.out.print("Metodo Abstracto");
         }
        public static void main(String args[])
        {
                PuebaAbstract obj = new PuebaAbstract();
                obj.miMetodo();
        }
}
```

Se está instanciando un objeto de una clase abstracta, por definición una clase abstracta no puede instanciar objetos:

java: p1.PruebaAbstracta is abstract; cannot be instantiated

2.13 Pregunta 13

¿Cuál sería la salida del siguiente código?

```
public abstract class Animal{
        protected string tipoAnimal; //vaca, perro gato

        public Animal(){}
        public string toString(){
                return "Soy un "+ tipoAnimal+ " y hago" + hablar();
        }
        public abstract string hablar();

}

public class Gato extends Animal{
        public Gato(){
                tipoAnimal="gato";
        }
        public string hablar(){
                return "miau";
}
}

public class Vaca extends Animal{
        public Vaca(){
                tipoAnimal="vaca";
        }
        public string hablar(){
                return "muu";
}
}
```

```
Animal animal= new Vaca();
System.out.println(animal.toString());
animal= new Gato();
System.out.println(animal.toString());
```

Soy un vaca y hagomuu
Soy un gato y hagomiau

2.14 Pregunta 14

¿Qué sucede cuando se intenta compilar y ejecutar el código?

```
public class SubClass extends SuperClass {
    private static void TestMethod(SuperClass c) {
        if (c instanceof SubClass) {
            System.out.println("C belongs to SubClass!");
        } else if (c instanceof SuperClass) {
            System.out.println("C belongs to SuperClass!");
        } else {
            System.out.println("C is alone");
        }
    }

    public static void main(String... args) {
        SuperClass m = new SubClass();
        SubClass.TestMethod(m);
    }
}
```

Después de haber creado la SuperClass para que el compilador la encuentra, el resultado es:

> C belongs to SubClass!

Al quitarle el static a TestMethod y al compilar y ejecutar sale este mensaje de error:

> java: non-static method TestMethod(p1.Subclass.SuperClass) cannot be referenced from a static context

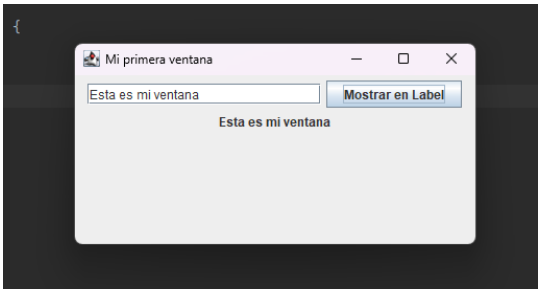
Para que el método funcione non-static habría que instanciar el objeto a partir de una Subclase y llamar al método non-static desde la misma instancia de la clase.

2.16. Pregunta 16

Cree una ventana que contenga una caja de texto, un botón y un label. Cuando escribamos algo en la caja de texto y le demos al botón, se escriba en el label el contenido de la caja de texto en el label.

```java
public class Ventana extends JFrame {
    private JTextField cajaTexto;
    private JLabel label;

    public Ventana(){
        //Configuramos la ventana
        super("Mi primera ventana");
        setSize(400,200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Creamos panel y agregamos componentes
        JPanel panel = new JPanel();
        add(panel);

        //Crear una caja de texto
        cajaTexto = new JTextField(20);
        panel.add(cajaTexto);

        //Crear un boton
        JButton boton = new JButton("Mostrar en Label");
        panel.add(boton);

        //Crear un label
        label = new JLabel("Texto");
        panel.add(label);

        //Crear un evento para el boton
        boton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                //Obtener el texto de la caja de texto
                String texto = cajaTexto.getText();
                //Mostrar el texto en el label
                label.setText(texto);
            }
        });
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        //instanciar un objeto Ventana
        Ventana ventana = new Ventana();
        //hacer visible la ventana
        ventana.setVisible(true);

    }
}
```



2.17. Pregunta 17

Cree una clase denominada miClase donde al menos uno de sus atributos sea privado. Intente modificar este atributo desde una objeto de esa clase, del tipo.

```java
public class miClase {
    private int x;
    public int y;

    public miClase(int x, int y){
        this.x = x;
        this.y = y;
    }

    void setX(int x){
        this.x = x;
    }

    int getX(){
        return this.x;
    }

}
```

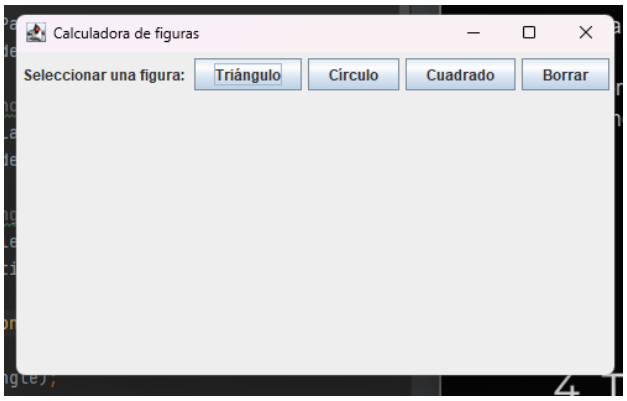
```java
public static void main(String[] args) {
    miClase obj1 = new miClase(1,2);
    obj1.setX(3);
    System.out.println(obj1.getX());
}
```

```
C:\Users\crerr\.jdks
3

Process finished wit
```

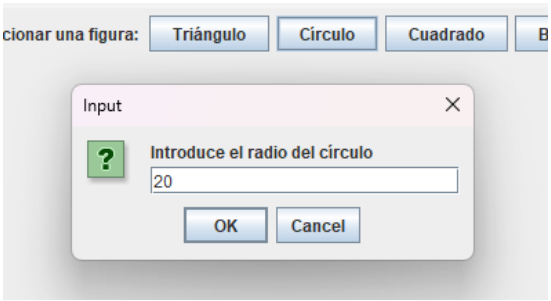
3. Parte 2

Flujo del programa

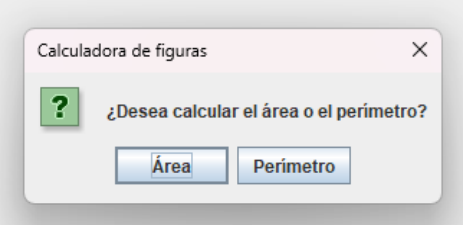
El programa principal empieza lanzando una ventana con varios botones con los que podemos seleccionar la figura para trabajar.

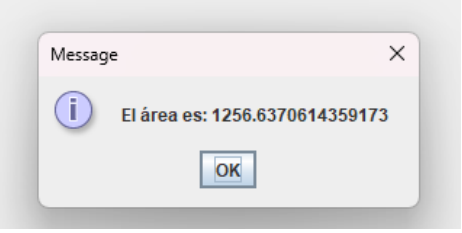


Cuando seleccionamos la figura aparecerá otra ventana con la solicitud de entrada de los datos:

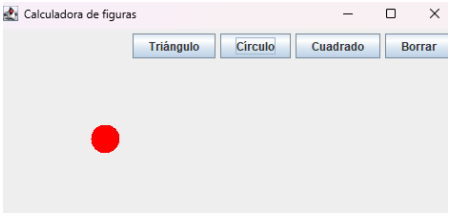


El programa preguntará si queremos calcular el área o el perímetro y mostrará la figura enseguida.



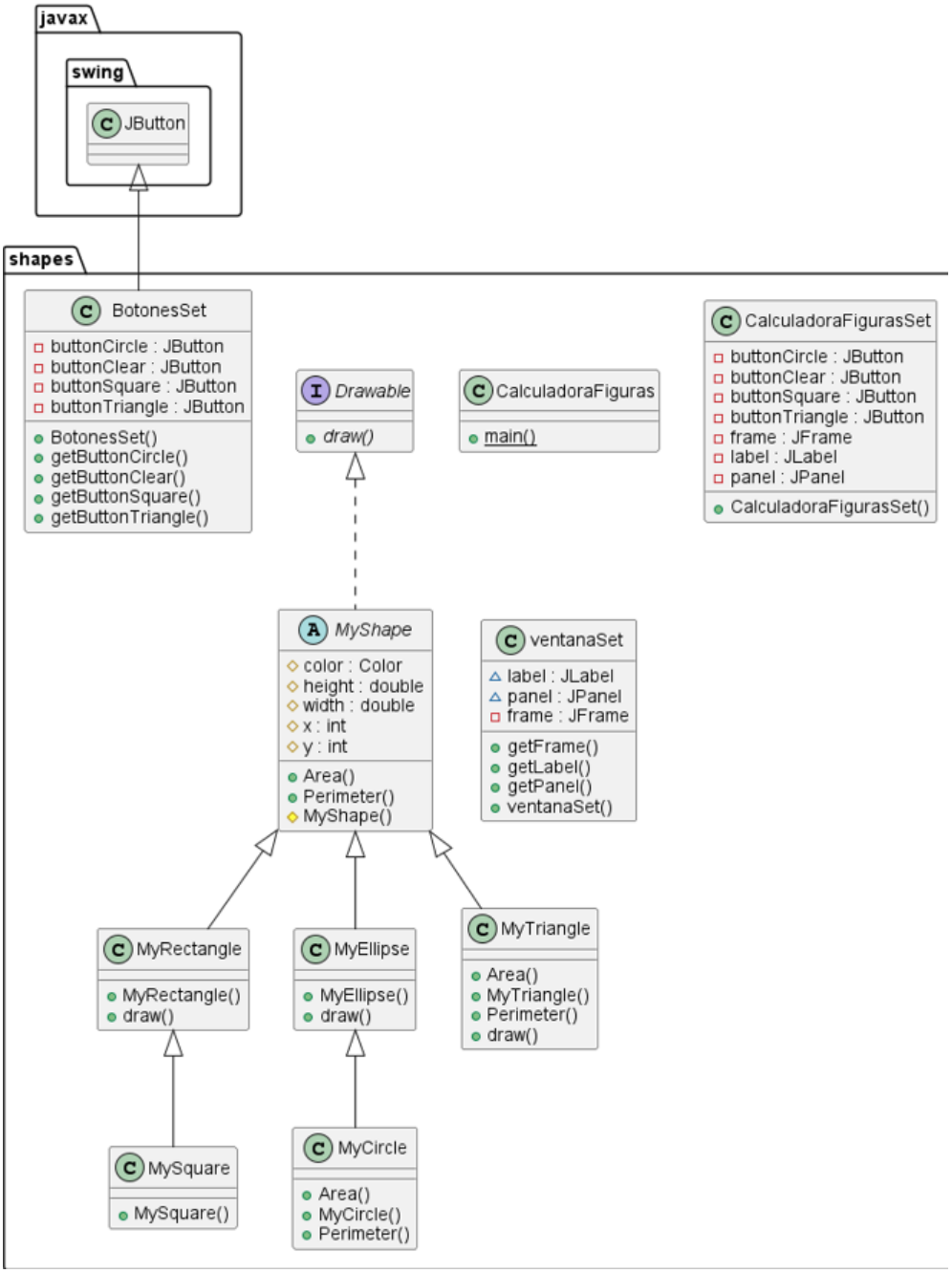


Y enseguida nos mostrará la imagen



Estructura

En el segundo ejercicio se ha desarrollado un programa que trabaja con figuras. Se ha creado una jerarquía de clases a partir de MyShape (Figura) de la que heredan las formas geométricas a representar:



MyShapes es la clase abstracta padre, MyTriangle es su hija, mientras que MySquare y MyCircle descienden de MyRectangle y MyEllipse. CalculadorFigurasSet es la clase que maneja la ventana y las figuras.

MyShapes es una clase abstracta que contiene solo constructor y los dos métodos para calcular Area y Perimetro de una figura, métodos que cada clase hija sobrecarga para poder adaptarlos a sus necesidades.

Cada clase hija aparte de sobrecargar el constructor, tiene su propio método

para calcular P erimetro, Area y el método draw que hace uso de Drawable para representar las figuras geométricas en pantalla.

En el programa principal se hace una llamada al constructor de la clase CalculadoraFigurasSet:

```
public class CalculadoraFiguras {
    public static void main(String[] args) {
        CalculadoraFigurasSet Frame = new CalculadoraFigurasSet();
    }
}
```

Esta, instancia una ventana con las clases ventanaSet y BotonesSet, que se encargan de inicializar la ventana con sus elementos base.

Por pantalla se muestra entonces la ventana con su estructura principal: el panel, la etiqueta y sus botones.

Cada vez que se pulse un botón se ejecutará el método actionPerformed, desde

```
buttonTriangle.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Solicitar la base y la altura del triángulo
        double base = Double.parseDouble(JOptionPane.showInputDialog("Introduce la base del triángulo"));
        double height = Double.parseDouble(JOptionPane.showInputDialog("Introduce la altura del triángulo"));

        // Crear un triángulo
        MyTriangle triangle = new MyTriangle(100, 100, base, height, Color.GREEN);

        // Preguntar si quiere calcular el área o el perímetro
        int opcion = JOptionPane.showOptionDialog(frame, "¿Desea calcular el área o el perímetro?", "Calculadora

        // Calcular el área o el perímetro según la opción seleccionada
        if (opcion == JOptionPane.YES_OPTION) {
            double area = triangle.Area();
            JOptionPane.showMessageDialog(frame, "El área es: " + area);
        } else {
            double perimeter = triangle.Perimeter();
            JOptionPane.showMessageDialog(frame, "El perímetro es: " + perimeter);
        }

        // Dibujar la figura
        Graphics g = panel.getGraphics();
        triangle.draw(g);
```

la interfaz ActionListener, sobrecargado para botón de cada figura:

Cuando se entra en el método se solicitarán los datos de entrada al usuario en una ventana emergente o “Input Dialog” , un método de swing.JOptionPane. Una vez seleccionada se creará la figura.

El objeto figura, una vez que se haya instanciado usará sus métodos sobrecargados para calcular el Área o el Perímetro.

Se lanzará luego, una ventana emergente que nos dirá el valor de estos y en seguida pintará la figura con la ayuda de la librería Graphics.

El Botón de borrar simplemente hace una limpieza de la ventana.

4. Tabla de tiempos

Todas las tareas se han ido realizando paso a paso con el informe, por lo que el tiempo de realización con el informe está incluído en los tiempos de cada tarea.

Tarea	Tiempo estimado	Tiempo Efectivo
Configuración IDE	30 min	30 min
Parte 1	1h	1h 30 min
Parte 2	2h	5h
Total	3h 30 min	7h