

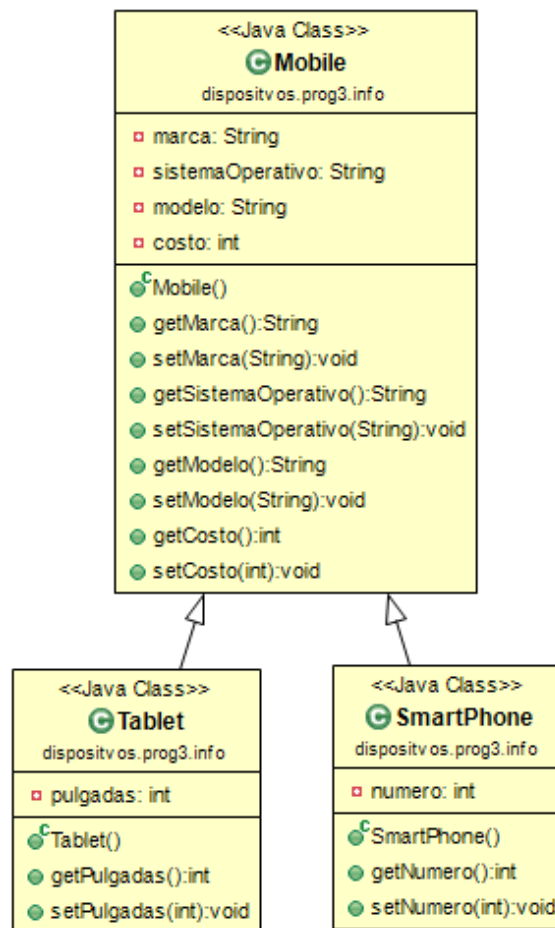
Programación III

TEMA 2: Conceptos Básicos - Herencia

Práctica nº 2 - B

1. Cree un proyecto llamado **DispositivosMoviles** y escriba la siguiente jerarquía de clases en JAVA.

NOTA: reutilice la clase Tablet de la práctica anterior, renombrando a "Mobile", y eliminando la variable de instancia pulgadas.



- a. En las clases **Tablet** y **SmartPhone** sobreescriba el método **public boolean equals(Object)** heredado de la clase **Object**. El método `equals` permite comparar 2 instancias (defina ud. cuando 2 objetos se consideran iguales).
- b. Sobreescriba también en ambas clases el método **public String toString()** heredado de la clase **Object**, de manera que retorne los datos de esos objetos de manera **legible**.

- c. Escriba una clase dispositivos.prog3.info.EjercicioTestSobreescritura y pruebe los métodos sobreescritos
- Defina dos objetos de tipo **SmartPhone**, setee el mismo valor en la variable de instancia **numero**. ¿Cuál es el resultado de invocar al método equals?.
 - Imprima el contenido de ambos objetos (use System.out.println y envíe directamente la instancia creada)

2. Jerarquía de animales

- a. Cree un proyecto llamado **Animales**.
- b. Escriba el siguiente código en Eclipse (cada clase debería ubicarse en su propio archivo y dentro del paquete **animales.prog3.info**).

NOTA: en caso de no quedar correctamente indentado, use Ctrl + Shift + f para formatear su código.

<pre>public abstract class Animal { public abstract void saludo(); }</pre>	<pre>public class Gato extends Animal { @Override public void saludo() { System.out.println("Miau!"); } }</pre>
<pre>public class Perro extends Animal { @Override public void saludo() { System.out.println("Guau!"); } public void saludo(Perro otro) { System.out.println("Guau! Guau!"); } }</pre>	<pre>public class PerroGrande extends Perro { @Override public void saludo() { System.out.println("Guauuuuuu!"); } @Override public void saludo(Perro otro) { System.out.println("Guauuuuuu! Guauuuuuu!"); } public boolean esMasBuenoQue(Animal otro){ return true; } }</pre>

```
public class TestAnimal1 {  
    public static void main(String[] args) {  
        Gato donGato = new Gato();  
        donGato.saludo();  
        Perro benji = new Perro();  
    }  
}
```

```
        benji.saludo();  
        PerroGrande lassie = new PerroGrande();  
        lassie.saludo();  
    }  
}
```

- c. Indique qué obtuvo como salida luego de la ejecución de TestAnimal1
- d. Ahora escriba el siguiente código:

```
public class TestAnimal2 {  
    public static void main(String[] args) {  
        Animal donGato = new Gato();  
        donGato.saludo();  
        Animal benji = new Perro();  
        benji.saludo();  
        Animal lassie = new PerroGrande();  
        lassie.saludo();  
    }  
}
```

- e. Analice en el código la diferencia entre TestAnimal1 y TestAnimal2 e indique qué obtuvo como salida luego de la ejecución de TestAnimal2.

i. Responda.

1. Considerando TestAnimal2, puede enviarle a lassie el mensaje "esMasBuenoQue(...)"? **JUSTIFIQUE**

- f. Escriba el siguiente código en eclipse:

```
public class TestAnimal3 {  
    public static void main(String[] args) {  
  
        Gato gato1 = new Gato();  
        gato1.saludo();  
        Perro perro1 = new Perro();  
        perro1.saludo();  
        PerroGrande perroGrande1 = new PerroGrande();  
        perroGrande1.saludo();  
  
        Animal animal1 = new Gato();  
        animal1.saludo();  
        Animal animal2 = new Perro();  
        animal2.saludo();  
        Animal animal3 = new PerroGrande();  
        animal3.saludo();  
    }  
}
```

```
Perro perro2 = animal2;  
PerroGrande perroGrande2 = animal3;  
Perro perro3 = animal3;  
Gato gato2 = animal2;  
perro2.saludo(perro3);  
perro3.saludo(perro2);  
perro2.saludo(perroGrande2);  
perroGrande2.saludo(perro2);  
perroGrande2.saludo(perroGrande1);  
}  
}
```

g. Corrija los errores en compilación y **JUSTIFIQUE**. ¿cómo se llama el mecanismo aplicado?

h. **Responda:** ¿es posible crear una instancia de la clase Animal? **JUSTIFIQUE**

3. Pasaje de parámetros.

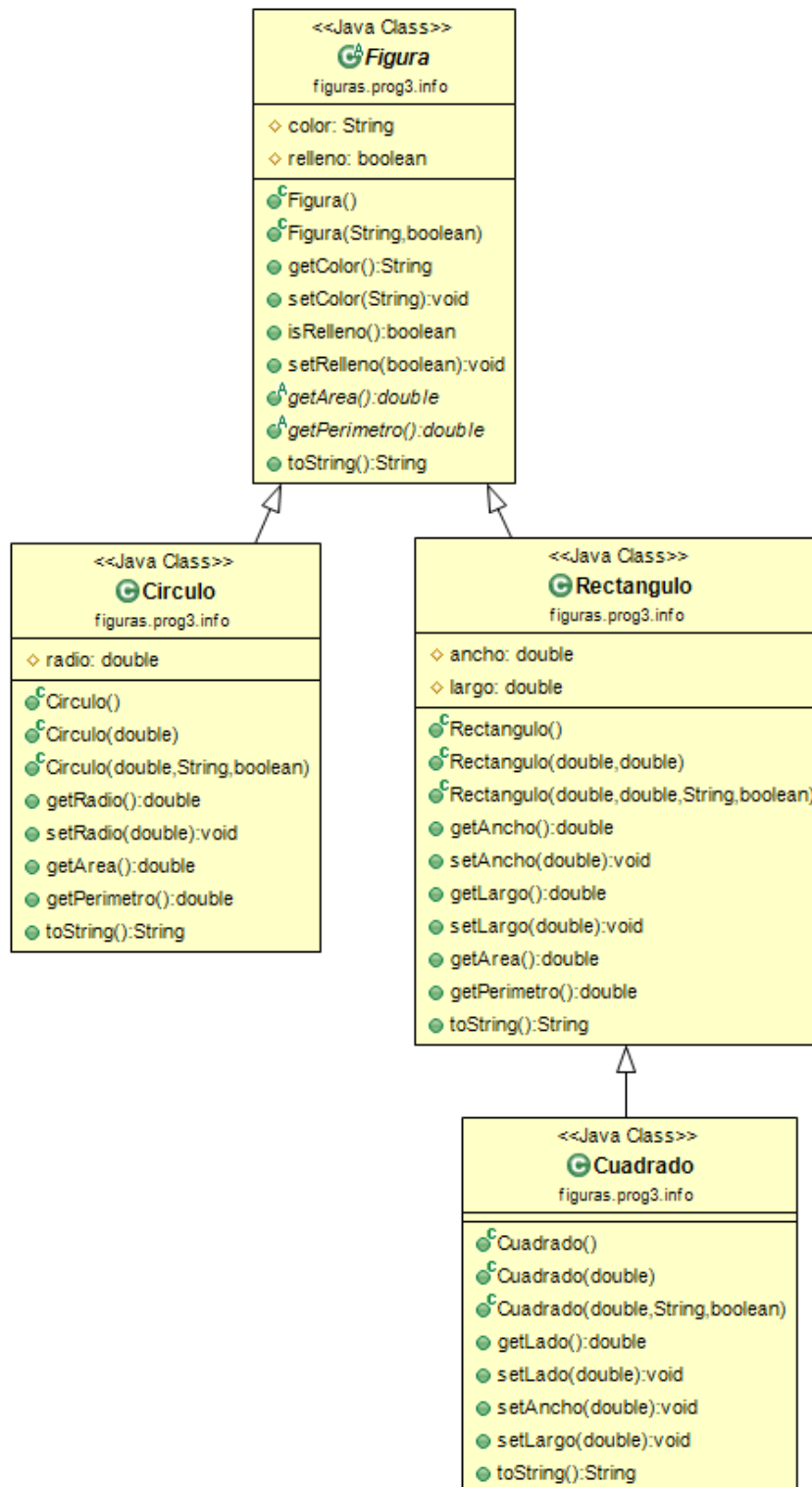
1.1. Analice el siguiente código **sin copiarlo en el Eclipse**. A su criterio ¿qué imprime?

```
public class SwapValores {  
    public static void swap1 (int x, int y) {  
        if (x < y) {  
            int tmp = x ;  
            x = y ;  
            y = tmp;  
        }  
    }  
    public static void swap2 (Integer x, Integer y) {  
        if (x < y) {  
            int tmp = x ;  
            x = y ;  
            y = tmp;  
        }  
    }  
    public static void main(String[] args) {  
        int a = 1, b = 2;  
        Integer c = 3, d = 4;  
        swap1(a,b);  
        swap2(c,d);  
        System.out.println("a=" + a + " b=" + b) ;  
        System.out.println("c=" + c + " d=" + d) ;  
    }  
}
```

- 1.2. Ahora copie el código en Eclipse
 - 1.3. Agregue un breakpoint en cada línea: `y = tmp;` ejecute en modo Debug y analice. ¿es correcta su respuesta?.
4. Cree una clase llamada **PruebaRetorno** con un método llamado `procesarArreglo`. Éste método recibe como argumento un arreglo de enteros y debe devolver el máximo, el mínimo y el promedio del mismo. Proponga 2 implementaciones **para la forma de devolver** el resultado de `procesarArreglo`. **Nota:** el método no puede devolver un arreglo.

5. Jerarquía de formas

a. Implemente en Java la siguiente jerarquía de clases



Nota: En este ejercicio, **Figura** está definida como una clase abstracta, la cual contiene:

- Dos variables de instancia privadas : color (String) y relleno (boolean).
- Getter y setter para todas las variables de instancia y el método toString().
- Dos métodos abstractos getArea() y getPerimetro().

Las subclases **Circulo** y **Rectangulo** deben sobrescribir los métodos abstractos getArea() y getPerimetro() y proveer implementación propia. También sobrescriben el método toString().

- b. Escriba una clase llamada TestDeFigurasGeometricas.
- c. Defina en el método "main" de la clase TestDeFigurasGeometricas un arreglo de 3 posiciones, donde almacenará objetos **de tipo Figura**.
- d. Agregue al arreglo 1 Círculo, 1 Rectangulo y 1 Cuadrado.
- e. Itere sobre el arreglo con una estructura de control de tipo "foreach" de modo que cada Figura imprima su información. **Responda:** ¿Qué método invocará?