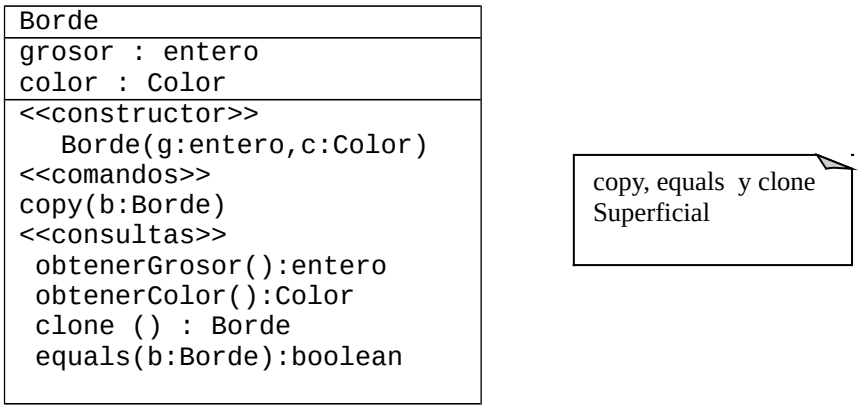
PRACTICO N° 6

EJERCICIO 1. Dada la clase Color implementada en el práctico 4, implemente la clase Borde de acuerdo al siguiente diagrama.



EJERCICIO 2. Dada la implementación de las clases Color y Borde y el siguiente segmento de código:

```
Color C1,C2;
Borde B1,B2;

C1 = new Color(110, 110, 110);
C2 = new Color(110, 110, 110);
B1 = new Borde(1,C1);
B2 = new Borde(1,C2);

System.out.println( C1 == C2);
System.out.println( B1 == B2);
System.out.println( C1.equals(C2));
System.out.println(B1.equals(B2));
```

a) Dibuje el diagrama de objetos al terminar el bloque de asignaciones.
b) Muestre la salida.

EJERCICIO 3. Dada la implementación de las clases Color y Borde y el siguiente segmento de código:

```
Color C1,C2;
Borde B1,B2,B3,B4;

C1 = new Color(110, 110, 110);                                  //1
C2 = new Color(110, 110, 110);                                  //2
B1 = new Borde(1,C1);                                           //3
B2 = new Borde(1,C2);                                           //4
B3 = B2.clone();                                                //5
B4 = new Borde (B2.obtenerGrosor(),B2.obtenerColor());          //6
```

```
System.out.println( B2 == B3);
System.out.println( B2 == B4);
System.out.println(B2.equals(B1));
System.out.println(B2.equals(B3));
System.out.println(B2.obtenerGrosor() == B1.obtenerGrosor() &
                   B2.obtenerColor() == B1.obtenerColor());
System.out.println(B2.obtenerGrosor() == B3.obtenerGrosor() &
                   B2.obtenerColor() == B3.obtenerColor());
System.out.println(B2.obtenerGrosor() == B1.obtenerGrosor() &
                   B2.obtenerColor().equals(B1.obtenerColor()));
System.out.println(B2.obtenerGrosor() == B4.obtenerGrosor() &
                   B2.obtenerColor().equals(B4.obtenerColor()));
```

a) Dibuje el diagrama de objetos cada vez que se establece o modifica una referencia.
b) Muestre la salida.

EJERCICIO 4. Muestre la evolución del diagrama de objetos y la salida del ejercicio anterior considerando que la clase Borde implementa:

a) clonación en profundidad, igualdad superficial
b) clonación superficial, igualdad en profundidad
c) clonación en profundidad, igualdad en profundidad

EJERCICIO 5 Muestre la salida que resulta al ejecutar main, dadas las clases A, B y C definidas como siguen:

```
class A {
   private int a1;
   private String a2;

public A (int at1,String at2)
{  a1 = at1; a2 = at2;}

public void establecerA1(int at1)
{ a1 = at1;}

public void establecerA2(String at2)
{ a2 = at2;}

public int obtenerA1()
{ return a1;}

public String obtenerA2()
{ return a2;} }
```

```
class B{

public void p (A x){
   x.establecerA1 (10);
   x.establecerA2 (new
String("azul"));}

public void q(A x){
   x=new A(11,
              new String("blanco"));}
}
```

```
class C {

public static void main (String arg[]){

  A a  = new A(1,new String("rojo");

  B b  = new B();

  b.p(a);

  System.out.println (a.obtenerA1()+" "+
a.obtenerA2());

  b.q(a);

  System.out.println (a.obtenerA1()+" "+
a.obtenerA2());
} }
```

EJERCICIO 6 Dadas las clases A, B y C definidas como siguen:

```
class A{
     private B b1;
     private int b2;

public A (B p1, int p2)
{    b1 = p1;
     b2 = p2;
 }
public B obtenerB1()
{ return b1;
}
public int obtenerB2()
{ return b2;
}
public A clone ()
{    A r = new A(b1.clone(),b2);
     return r;
}
public boolean equals(A p)
{  return (b1.equals(p.obtenerB1())
            && b2 == p.obtenerB2());
}
}
```

```
class B{
     private C c1;
     private char c2;

public B (C p1, char p2)
{  c1 = p1;
   c2 = p2;
}
public C obtenerC1()
{   return c1;
}
public char obtenerC2()
{    return c2;
}
public B clone ()
{  B r = new B(c1,c2);
  return r;
}
public boolean equals(B p)
{  return (c1 == p.obtenerC1()
       && c2 == p.obtenerC2());
}
}
```

```
class C{
   private int d1;
   private char d2;
public C (int p1, char p2)
{   d1 = p1;
   d2 = p2;
}
public int obtenerD1()
{ return d1;
}
public char obtenerD2()
{  return d2;
}
public C clone ()
{  return new C (d1,d2);
}
public boolean equals(C p)
{   return (d1 == p.obtenerD1()
          && d2 == p.obtenerD2());
}
}
```

Y las instrucciones:

```
C vc = new C (10,´a´);
B vb1 = new B (vc,'5');
B vb2 = new B (vc.clone(),'5');
A va1 = new A (vb1.clone(), 1);
A va2 = new A (vb1.clone(), 1);
A va3 = new A (vb2, 1);
A va4 = new A(new B(new C(11,'b'),'c'),-1);
```

a) Elaborar el diagrama de objetos luego de ejecutarse las instrucciones anteriores.
b) Agregar e implementar el método copy en las clases A, B y C, en profundidad en la clase A y superficial en B.
c) Indicar los valores de verdad de las siguientes expresiones:

```
va1.equals(va2)
va1.equals(va3)
```