

19/12/2022

Cubo

Capacidad contenido



atributos, características

Cubo (capacidad)

Vaciar ()

Llenar()

Volcarth(cubo)

Pinta()

Métodos
(comportamiento)

PROGRAMA 1. CUBO

```
package Programacion_objetos;
```

```
public class cubo {
```

```
    int capacidad, contenido;
```

```
    cubo(int capa){  
        capacidad = capa;  
    }
```

```
    void vaciar () {  
        contenido = 0;  
    }
```

```
    void vaciar (int c) {  
        if (c>contenido || contenido - c<0) {  
            contenido =0;  
        }  
        else {  
            contenido -= c;  
        }  
    }
```

```
    void llenar () {  
        contenido = capacidad;  
    }
```

```
    int getCapacidad () {  
        return this.capacidad;  
    }
```

```
    int getContenido() {  
        return this.contenido;  
    }
```

```
    void setContenido(int c) {  
        this.contenido = c;  
    }
```

```

    void volcar (cubo destino, int cantidad) {
        this.contenido = this.contenido-cantidad;
        destino.contenido = destino.contenido + cantidad;
    }

    int pinta(int x) {
        return x;
    }
}
}PROGRAMA 2. PRUEBA_CUBO
package Programacion_objetos;

public class prueba_cubo {

    public static void main(String[] args) {

        cubo cubo1 = new cubo(5);

        System.out.println(cubo1.capacidad);
        System.out.println(cubo1.getCapacidad());

        cubo cubo3=new cubo(3);
        System.out.println("el contenido es " + cubo3.getContenido());

        cubo3.llenar();

        cubo3.vaciar();

        System.out.println("el contenido es " + cubo3.getContenido());

        cubo1.llenar();
        cubo3.volcar(cubo1, 2);

        System.out.println("el contenido de cubo 3 es " +
cubo3.getContenido());
        System.out.println("el contenido de cubo 1 es " +
cubo1.getContenido());

    }

}

```

EJERCICIO 7.15

7.15. Diseña la clase `Calendario` que representa una fecha concreta (año, mes y día). La clase debe disponer de los métodos:

- `Calendario(int año, int mes, int dia)`: que crea un objeto con los datos pasados como parámetros, siempre y cuando, la fecha que representen sea correcta.
- `void incrementarDia()`: que incrementa en un día la fecha del calendario.
- `void incrementarMes()`: que incrementa en un mes la fecha del calendario.
- `void incrementarAño(int cantidad)`: que incrementa la fecha del calendario en el número de años especificados. Ten en cuenta que el año 0 no existió.
- `void mostrar()`: muestra la fecha por consola.
- `boolean iguales(Calendario otraFecha)`: que determina si la fecha invocante y la que se pasa como parámetro son iguales o distintas.

Por simplicidad, solo tendremos en consideración que existen meses con distinto número de días, pero no tendremos en cuenta los años bisiestos.

ANIMAL 09/01/2023

```
package Programacion_objetos;

public class animal {

    //TEMA 7

    // -Tipo
    // -nombre
    //metodos: duerme (), come()
    //getter y soltery

    private String tipo;
    public String nombre;

    public animal(String ti, String no) {
        this.tipo = ti;
        this.nombre= no;
    }

    public String getTipo() {
        return tipo;
    }

    public String getNombre() {
        return nombre;
    }

    public String setNombre() {
        return nombre;
    }

    public void come() {
        System.out.println("ñam ñam");
    }

    public void duerme() {
        System.out.println("Zzzzzz");
    }

    public String toString() {
        return "Animal de tipo " + tipo + " y con nombre " + nombre;
    }
}
```



```
package Programacion_objetos;
```

```
public class animal_prueba {
```

```
    public static void main(String[] args) {
```

```
        animal a1 = new animal("domestico", "Paco");
```

```
        animal a2= new animal("salvaje", "gato");
```

```
        System.out.println(a1);
```

```
        a1.nombre = "pepe";
        System.out.println(a1);
```

```
        a1.setNombre("maria");
        System.out.println(a1);
```

```
        tigre t1 = new tigre("domestico", "Ana", "Persa");
        t1.come();
        t1.
```

```
    }
```

```
}
```

```
package Programacion_objetos;
```

```
public class animal_prueba {
```

```
    public static void main(String[] args) {
```

```
        animal a1 = new animal("domestico", "Paco");
```

```
        animal a2= new animal("salvaje", "gato");
```

```
        tigre t1 = new tigre("domestico", "Ana", "Persa");
```

```

        System.out.println(t1);

    }

}

package Programacion_objetos;

public class animal_prueba {

    public static void main(String[] args) {

        animal a1 = new animal("domestico", "Paco");

        animal a2= new animal("salvaje", "gato");

        tigre t1 = new tigre("domestico", "Ana", "Persa");

        System.out.println(t1);

        tigre t2 = new tigre();
        System.out.println(t2);

        ave ave1 = new ave("rojo");
        System.out.println(ave1);

        gallo g1 = new gallo ("grande");
        System.out.println(g1);

    }

}

```

```

package Programacion_objetos;

public class animal_prueba {

    public static void main(String[] args) {

        animal a1 = new animal("domestico", "Paco");

        animal a2= new animal("salvaje", "gato");

        tigre t1 = new tigre("domestico", "Ana", "Persa");

        System.out.println(t1);

        tigre t2 = new tigre();
        System.out.println(t2);

        ave ave1 = new ave("rojo");
        System.out.println(ave1);

    }

}

```

```
        gallo g1 = new gallo ("grande");  
        System.out.println(g1);  
        g1.getColor("verde");  
    }  
}
```

TIGRE

```
package Programacion_objetos;

public class tigre extends animal{

    //super hace referencia a animal, a la super clase, llama al
    //superconstructor de la clase padre = animal

    private String raza;
    public tigre (String ti, String n, String raza) {
        super(ti, n);
    }

    public void come() {
        System.out.println("Estoy hambriento, soy un tigre");
    }

    public void come(String comida) {
        System.out.println(comida);
    }

    public String toString() {
        return "Tigre " + super.toString() + " con raza " + this.raza;
    }

    public tigre () {
        super("no se sabe", "no se sabe");
        raza = "sin especificar";
    }
}
```


AVE

```
package Programacion_objetos;

public class ave extends animal {
    //herencia
    public ave() {
        super();
    }

    public ave (String color) {
        this.color = color;
    }

    private String color;

    public String toString()
    {
        return "este es el toString de ave " + this.color;
    }

    public String getColor() {
        return color;
    }
    public void setcolor(String color) {
        this.color = color;
    }
}
```

GALLO

```
package Programacion_objetos;

public class gallo extends ave {

    private String tipoPico;

    public gallo (String tipoPico) {
        this.tipoPico = tipoPico;
    }

    public String gettipoPico() {
        return tipoPico;
    }

    public void settipoPico (String tipoPico) {
        this.tipoPico = tipoPico;
    }

    public void vuela () {
        System.out.println("Vuela hasta el cielo");
    }

}
```

ATRIBUTOS Y MÉTODOS DE CLASE 11/01/2023

COCHE

- **ATRIBUTO DE INSTANCIA** EJ. Raza
- **Métodos instancia.** Ej devolver
- **Atributos de clase**

```
package Programacion_objetos;

public class coche {

    private String marca;
    private String modelo;
    private int kilometraje;

    //atributo clase
    private static int kilometrajeTotal = 0;

    public coche(String ma, String mo) {
        marca = ma;
        modelo = mo;
        kilometraje = 0;
    }

    public int getkilometraje() {
        return kilometraje;
    }

    //recorre una distancia
    public void recorre (int km) {
        kilometraje += km;
        kilometrajeTotal += km;
    }
}
```

```

    }

    //metodo clase
    public static int getkilometrajeTotal() {
        return kilometrajeTotal;
    }
}

```

MAIN

```

/*Programa que prueba la clase Coche*/
public class Prueba_Coche {
    public static void main(String[] args) {
        Coche cocheDeLuis = new Coche("Renault", "Megane");
        Coche cocheDeJuan = new Coche("Toyota", "Avensis");
        cocheDeLuis.recorre(30);
        cocheDeLuis.recorre(40);
        cocheDeLuis.recorre(230);

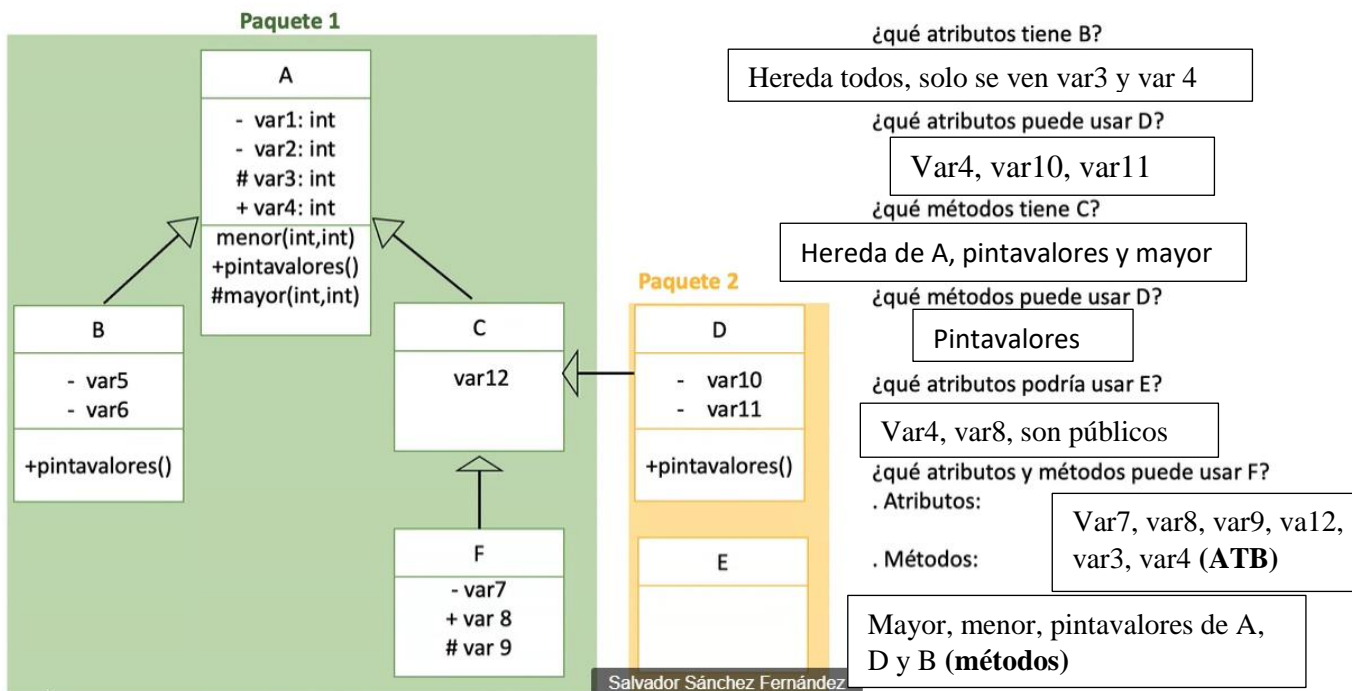
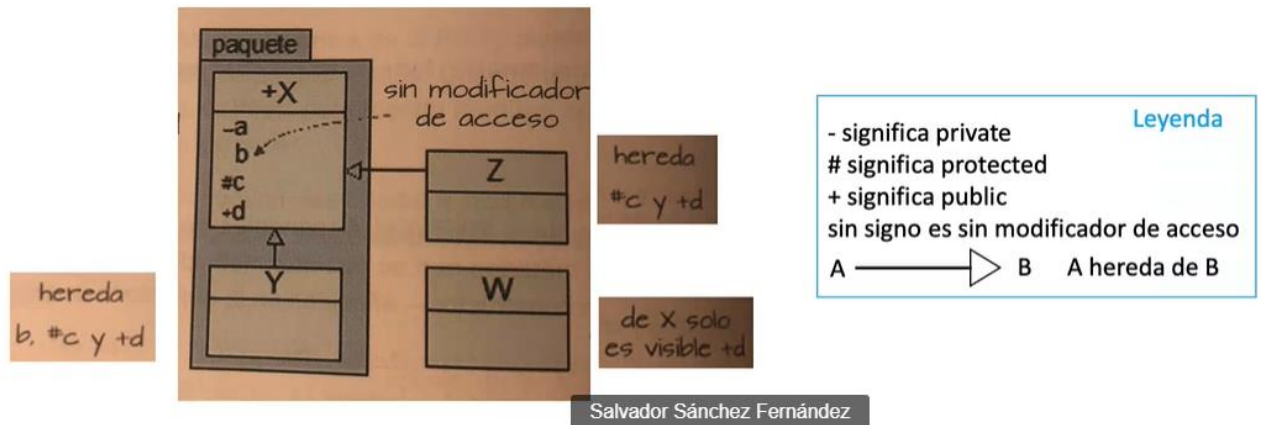
        cocheDeJuan.recorre(60);
        cocheDeJuan.recorre(140);

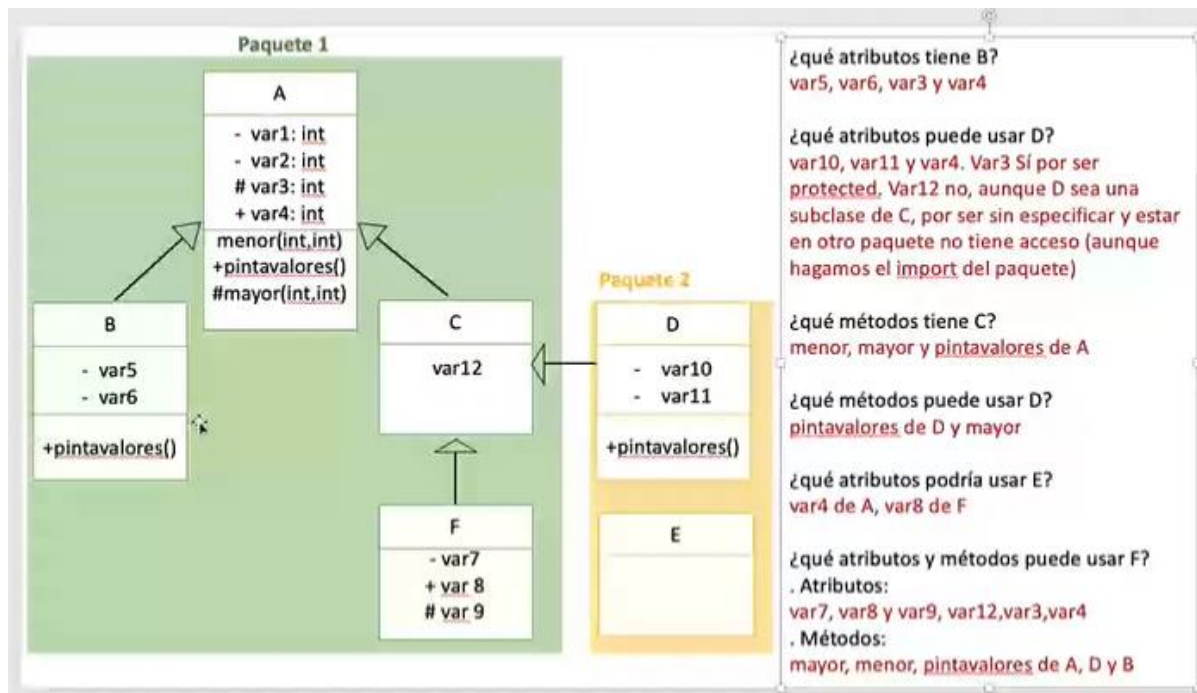
        System.out.println("El coche de Luis:"+cocheDeLuis.getKilometraje()+"Km");
        System.out.println("El coche de Juan:"+cocheDeJuan.getKilometraje()+"Km");
        System.out.println("Km totales:"+Coche.getKilometrajeTotal()+ "Km");
    }
}

```

VISIBILIDAD + HERENCIA

	En la misma clase	En el mismo paquete	En una subclase	Fuera del paquete
private	✓	✗	✗	✗
protected	✓	✓	✓	✗
public	✓	✓	✓	✓
sin especificar	✓	✓	✗	✗





MÉTODO EQUALS 12/01/2022

Para comparar

El método equals(), se utiliza para comparar dos objetos.

Ojo!, no confundir con el operador ==, que nos servía para comparar tipos de datos primitivos (int, double,...). Si lo usamos con objetos estaríamos comparando referencias de memoria.

equals() se usa para saber si dos objetos son del mismo tipo y tienen los mismos datos. Nos dará el valor true si son iguales y false si no.

Podemos sobrescribir el método equals() para hacer una comparación entre dos objetos, con el criterio o criterios de comparación que queramos.

En la lista de argumentos del método equals() hay que pasarle un argumento de tipo Object. No olvides poner el @Override, sino se sobrecarga el método, no se sobrescribe.

Supongamos que en la clase Gato redefino el método equals de la siguiente manera:

```
@Override
public boolean equals(Object obj) {
    Gato other = (Gato) obj;
    if (!color.equals(other.color))
        return false;
    else
        return true;
}
```

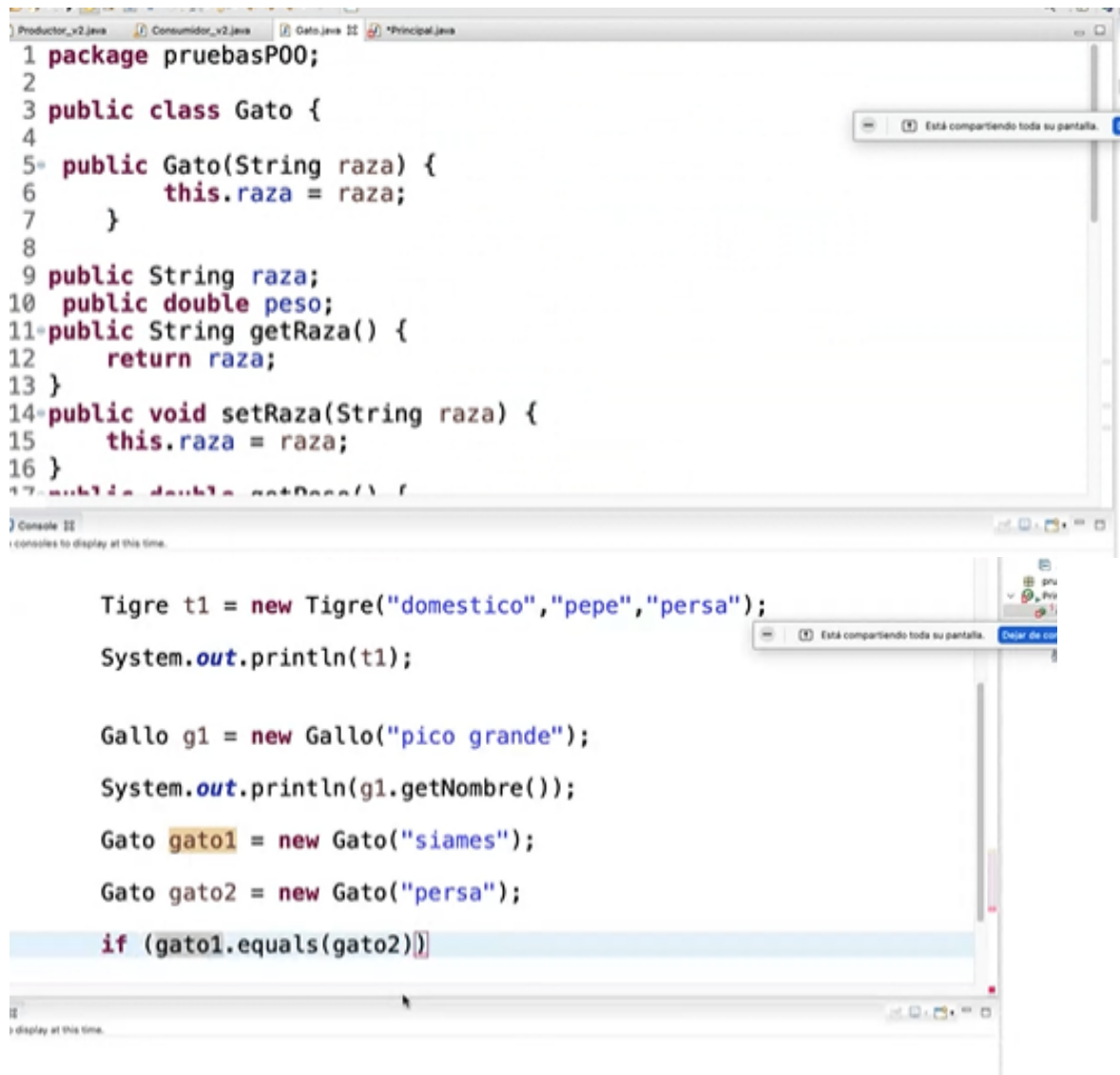
Así, en la otra clase donde tenga el main, puedo comparar objetos de tipo Gato, comprobando solo su color.

```
Gato garfield = new Gato("persa");
Gato isidoro = new Gato("persa");
```

Salvador Sánchez Fernández



```
23-@Override
24-public String toString() {
25-    return "Gato [raza=" + raza + ", peso=" + peso + "]";
26-}
27-
28-@Override
29-public boolean equals(Object obj) {
30-
31-    Gato otro = (Gato) obj;
32-    if (!raza.equals(otro.raza))
33-        return false;
34-    else
35-        return true;
36-}
37-
38-
```



The image shows a screenshot of an IDE with two open files. The top file, `Gato.java`, contains the following code:

```
1 package pruebasP00;
2
3 public class Gato {
4
5     public Gato(String raza) {
6         this.raza = raza;
7     }
8
9     public String raza;
10    public double peso;
11    public String getRaza() {
12        return raza;
13    }
14    public void setRaza(String raza) {
15        this.raza = raza;
16    }
17    public double getPeso() {
```

The bottom file, `Principal.java`, contains the following code:

```
Tigre t1 = new Tigre("domestico","pepe","persa");
System.out.println(t1);

Gallo g1 = new Gallo("pico grande");
System.out.println(g1.getNombre());

Gato gato1 = new Gato("siames");
Gato gato2 = new Gato("persa");

if (gato1.equals(gato2))
```

Both files have a console window at the bottom, which is currently empty. The IDE interface includes a toolbar at the top with icons for running, debugging, and other actions. A status bar at the bottom indicates the current line and column.

ARRAYS DE OBJETOS

Arrays de objetos

Del mismo modo que se pueden crear arrays de números enteros, decimales o cadenas de caracteres, también es posible crear arrays de objetos.

Vamos a definir la clase Alumno para luego crear un array de objetos de esta clase.

```
/*Definición de la clase Alumno*/
public class Alumno {

    private String nombre;
    private double notaMedia = 0.0;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public double getNotaMedia() {
        return notaMedia;
    }
    public void setNotaMedia(double notaMedia) {
        this.notaMedia = notaMedia;
    }
}
```

```
/*Definición de la clase Alumno*/
public class Alumno {

    private String nombre;
    private double notaMedia = 0.0;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public double getNotaMedia() {
        return notaMedia;
    }
    public void setNotaMedia(double notaMedia) {
        this.notaMedia = notaMedia;
    }
}
```

A continuación se define un array de cinco alumnos que posteriormente se rellena. Por último se muestran los datos de los alumnos por pantalla.

La siguiente línea únicamente define la estructura del array pero no crea los objetos:

```
Alumno[] alum = new Alumno[5];
```

Cada objeto concreto se crea de forma individual mediante `alum[i] = new Alumno();`

Aquí tienes el ejemplo completo.

```
/*Programa que prueba un array de la clase Alumno*/
import java.util.Scanner;

public class PruebaAlumnos {
```



```

import java.util.Scanner;

public class PruebaAlumnos {

    public static void main(String[] args) {

        String nombre;
        double nota;
        double sumaDeMedias = 0;

        Scanner sc = new Scanner(System.in);

        // Define un array de 5 alumnos pero no crea los objetos
        Alumno[] alum = new Alumno[5];

        // Pide los datos de los alumnos
        System.out.println("Introduce nombre y nota media de 5 alumnos.");

        for(int i = 0; i < 5; i++) {

            i

            alum[i] = new Alumno();
            System.out.println("Alumno:" + i);

            System.out.print("Nombre: ");
            nombre = sc.next();
            alum[i].setNombre(nombre);

            System.out.print("Nota media: ");
            nota = sc.nextDouble();
            alum[i].setNotaMedia(nota);
        }

        // Muestra los datos de los alumnos
        System.out.println("Los datos introducidos son:");
        sc.close();

        for(int i = 0; i < 5; i++) {
            System.out.println("Alumno " + i);
            System.out.println("Nombre: " + alum[i].getNombre());
            System.out.println("Nota media: " + alum[i].getNotaMedia());
            System.out.println("-----");
            sumaDeMedias += alum[i].getNotaMedia();
        }
    }
}

```

```

        nota = sc.nextDouble();
        alum[i].setNotaMedia(nota);
    }

    // Muestra los datos de los alumnos
    System.out.println("Los datos introducidos son:");
    sc.close();

    for(int i = 0; i < 5; i++) {
        System.out.println("Alumno " + i);
        System.out.println("Nombre: " + alum[i].getNombre());
        System.out.println("Nota media: " + alum[i].getNotaMedia());
        System.out.println("-----");
        sumaDeMedias += alum[i].getNotaMedia();
    }
    System.out.println("La media de la clase es " + sumaDeMedias / 5);
}
}

```

ALUMNO

```

package Programacion_objetos;

public class ALUMNO {

    private String nombre;
    private double notaMedia = 0.0;

    public String getNOMBRE() {
        return nombre;
    }
    public void setNOMBRE(String nombre) {
        this.nombre=nombre;
    }
    public double getnotaMedia() {
        return notaMedia;
    }
    public void setnotaMedia(double NotaMedia) {
        this.notaMedia=NotaMedia;
    }
}

```

COMPOSICIÓN DE OTRAS CLASES

HUEVO

16/01/2022

```
package HUEVOS_COMPOSICION_DE_CLASES;

//array yema
//equals de huevo comparando objetos directamente

public class HUEVO {

    private Yema yema;
    private Clara clara;

    //constructor
    HUEVO(Yema y, Clara c){
        this.yema=y;
        this.clara=c;
    }

    //get yema
    public Yema getYema() {
        return this.yema;
    }

    //get clara
    public Clara getClara() {
        return this.clara;
    }

    //setter yema
    public void setYema(Yema y) {
        this.yema = y;
    }

    //setter clara
    public void setClara(Clara c) {
        this.clara=c;
    }

    //toString
    public String toString() {
        return "huevo con yema " + yema.getcolor() + " y clara con volumen " + clara.getvolumen();
    }

    //equals
    @Override
    public boolean equals(Object o) {
        HUEVO otro = (HUEVO) o;

        if (this.clara.getvolumen() == otro.clara.getvolumen() && this.yema.getcolor().equals(otro.yema.getcolor())) {
            return true;
        }
        else {
            return false;
        }
    }
}
```

```
}
```

YEMA

```
package HUEVOS_COMPOSICION_DE_CLASES;

public class Yema {

    private String color;

    //constructor
    public Yema(String color) {
        this.color=color;
    }

    public Yema() {
        this.color="amarillo";
    }

    //getter
    public String getcolor() {
        return color;
    }

    //setter
    public void setcolor(String color) {
        this.color=color;
    }

}

//equals
@Override
public boolean equals(Object x) {

    Yema otro = (Yema) x;

    if (this.equals(x) == otro.equals(x)) {
        return true;
    }
    else {
        return false;
    }
}
```

CLARA

```
package HUEVOS_COMPOSICION_DE_CLASES;

public class Clara {

    //atributo
    private double volumen;
    //constructor
    public Clara(double volumen) {
        this.volumen=volumen;
    }

    public Clara() {
        this.volumen=0.0;
    }

    //getter
    public double getvolumen() {
        return volumen;
    }

    //setter
    public void setcvolumen(double volumen) {
        this.volumen=volumen;
    }

    //equals
    @Override
    public boolean equals(Object a) {
        Clara otro = (Clara) a;

        if (this.equals(a) == otro.volumen) {
            return true;
        }
        else {
            return false;
        }
    }

}
```

PRINCIPAL (MAIN)

```
package HUEVOS_COMPOSICION_DE_CLASES;

public class Principal {

    public static void main(String[] args) {

        //YEMA
        Yema y1 = new Yema("naranja");
        Yema y2 = new Yema("amarillo");
        System.out.println(y2.getcolor());

        //CLARA
        Clara c1 = new Clara(3.4);
        Clara c2 = new Clara(3.4);
    }

}
```

```

//HUEVO
HUEVO h1 = new HUEVO(y2,c2);
HUEVO h2 = new HUEVO (y1, c1);
System.out.println(h1);

if(h2.equals(h1)) {
    System.out.println("Los huevos son iguales");
}
else {
    System.out.println("Son distintos");
}

//ARRAY
Yema[] arraysYemas = new Yema[2];
arraysYemas[1]=y1;

System.out.println(y1.getcolor());

//MATRIZ YEMA

Yema matrizYemas[] = {y1,y2};
Clara miClara = new Clara (7);
HUEVO h = new HUEVO (matrizYemas, miClara);

//OTRA MATRIZ
Yema otraMatriz[] = h.getYema();

}

}

```

Clases internas Ejemplo: Huevo, Yema y Clara 23/01/2023

HUEVO

```
package Clases_internas;

public class Huevo {

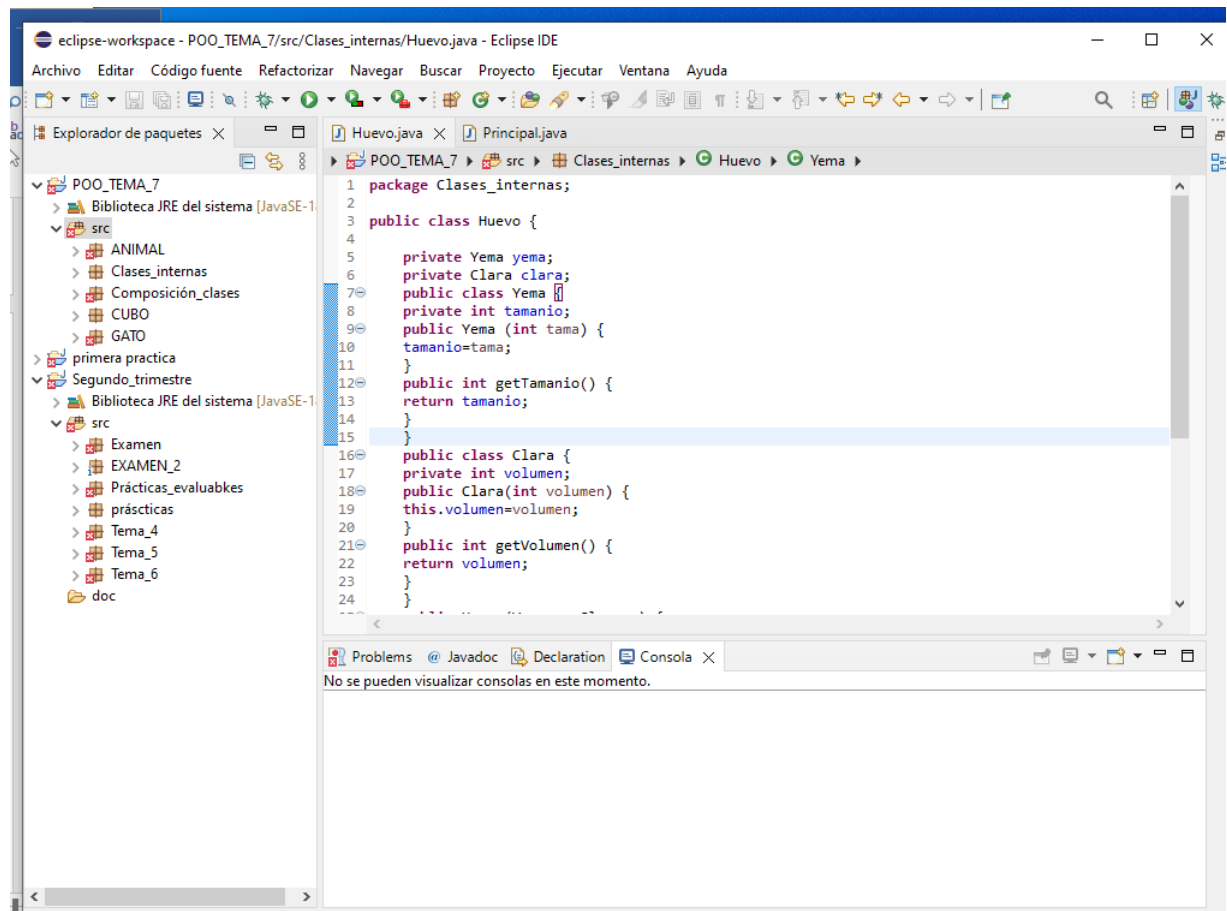
    private Yema yema;
    private Clara clara;
    public class Yema {
        private int tamano;
        public Yema (int tama) {
            tamano=tama;
        }
        public int getTamano() {
            return tamano;
        }
    }
    public class Clara {
        private int volumen;
        public Clara(int volumen) {
            this.volumen=volumen;
        }
        public int getVolumen() {
            return volumen;
        }
    }
    public Huevo(Yema y, Clara c) {
        this.yema = y;
        this.clara=c;
    }
    public Huevo(int tamanoYema, int volumenClara) {
        this.yema = new Yema(tamanoYema);
        this.clara = new Clara(volumenClara);
    }
    public Yema getYema() {
        return yema;
    }
    public Clara getClara() {
        return clara;
    }
}
```

MAIN

```
package Clases_internas;
```

```
public class Principal {
```

```
    public static void main(String[] args) {  
        Huevo h1 = new Huevo(10,20);  
        System.out.println(h1.getYema().getTamanio());  
    }  
}
```



TIPO NUMERADO FUTBOLISTA

```
package TIPO_ENUMERADO;

public class futbolista {

    private int dorsal;
    private String Nombre;
    private Demarcacion demarcacion;
    private Equipo equipo;

    public futbolista() {
    }

    public futbolista(int dorsal, String nombre, Demarcacion demarcacion,
Equipo equipo) {
        super();
        this.dorsal = dorsal;
        Nombre = nombre;
        this.demarcacion = demarcacion;
        this.equipo = equipo;
    }

    //getter y setter

    /**
     * @return el dorsal
     */
    public int getDorsal() {
        return dorsal;
    }

    /**
     * @param dorsal el dorsal a establecer
     */
    public void setDorsal(int dorsal) {
        this.dorsal = dorsal;
    }

    /**
     * @return el nombre
     */
    public String getNombre() {
        return Nombre;
    }

    /**
     * @param nombre el nombre a establecer
     */
    public void setNombre(String nombre) {
        Nombre = nombre;
    }

    /**
     * @return el demarcacion
     */
    public Demarcacion getDemarcacion() {
        return demarcacion;
    }
}
```

```

    }

    /**
     * @param demarcacion el demarcacion a establecer
     */
    public void setDemarcacion(Demarcacion demarcacion) {
        this.demarcacion = demarcacion;
    }

    /**
     * @return el equipo
     */
    public Equipo getEquipo() {
        return equipo;
    }

    /**
     * @param equipo el equipo a establecer
     */
    public void setEquipo(Equipo equipo) {
        this.equipo = equipo;
    }

    //getter y setter

```

```

}

```

EQUIPO

```

package TIPO_ENUMERADO;

public enum Equipo {

    REAL_MADRID, VILLAREAL, BARÇA, SEVILLA
}

```

DEMARCACION

```

package TIPO_ENUMERADO;

public enum Demarcacion {

    PORTERO, DEFENSA, CENTROCAMPISTA, DELANTERO
}

```

PRINCIPAL

```
package TIPO_ENUMERADO;

public class Principal {

    public static void main(String[] args) {

        futbolista casillas = new futbolista ("Casillas", 1,
Demarcacion.PORTERO, Equipo.REAL MADRID);
        futbolista capdevila = new futbolista ("Capdevila", 11,
Demarcacion.DEFENSA, Equipo.VILLAREAL);
        futbolista iniesta = new futbolista ("Iniesta", 6,
Demarcacion.DEFENSA, Equipo.BARÇA);
        futbolista navas = new futbolista ("Navas", 22,
Demarcacion.DELANTERO, Equipo.SEVILLA);

    }

}
```

```
package enumerados;

enum Colores { ROJO, VERDE, AZUL }

public class PruebaColores {

    public static final String BLACK = "\u001B[30m";
    public static final String RED = "\u001B[31m";
    public static final String GREEN = "\u001B[32m";
    public static final String YELLOW = "\u001B[33m";
    public static final String BLUE = "\u001B[34m";
    public static final String PURPLE = "\u001B[35m";
    public static final String CYAN = "\u001B[36m";

    public static final String RESET = "\u001B[0m";

    public static void pinta(Colores color) {
        switch (color) {
            case ROJO:
                System.out.println(RED+"El color es rojo"+RESET);
                break;
            case VERDE:
                System.out.println(GREEN+"El color es verde"+RESET);
                break;
            case AZUL:
                System.out.println(BLUE+"El color es azul"+RESET);
                break;
        }
    }

    public static void main(String[] args) {
        pinta(Colores.ROJO);
    }

}
```

LISTA ORDENADA

LISTA

```
package Lista_ordenada;
```

```
public class Lista {
```

```
    private Nodo primero;
```

```
    private Nodo ultimo;
```

```
    private int tamaño;
```

```
    private class Nodo{
```

```
        private Object elemento;
```

```
        private Nodo siguiente;
```

```
    }
```

```
}
```

PRINCIPAL