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

```java
package Programacion_objetos;

public class animal {

	//TEMA 7

	// -Tipo
	// -nombre
	//metodos: duerme (), come()
	//gettery 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;
	}
}
```

```java
package pruebasPOO;

public class Animal {

    private String tipo;
    private String nombre;

    public Animal(String ti, String n) {
        this.tipo= ti;
        this.nombre=n;
    }

    public String getTipo() {
        return tipo;
    }

    public String getNombre() {
        return nombre;
    }

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

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

```java
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.


	}

}
```

```java
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

```java
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**

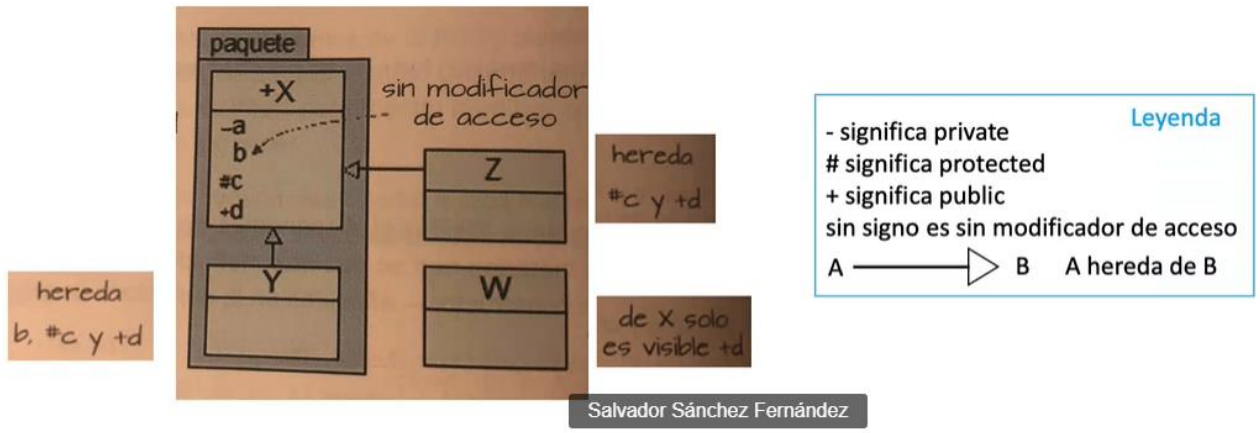
```
package Programacion_objetos;

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;
	}


}
```

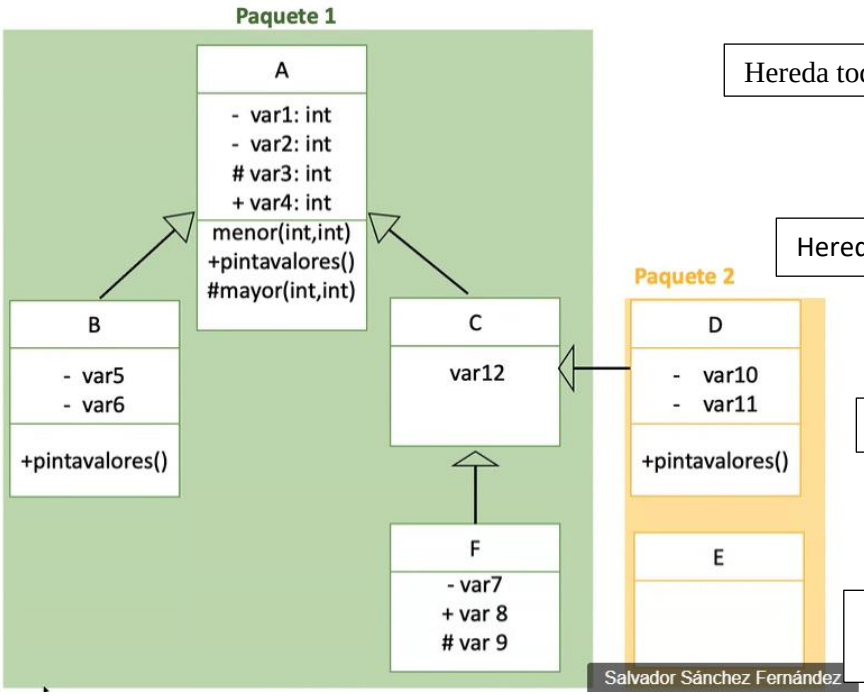
MAIN

```
/*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");

}
}
```

VISIBILIDIDAD + HERENCIA

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





¿qué atributos tiene B?

Hereda todos, solo se ven var3 y var 4

¿qué atributos puede usar D?

Var4, var10, var11

¿qué métodos tiene C?

Hereda de A, pintavalores y mayor

¿qué métodos puede usar D?

Pintavalores

¿qué atributos podría usar E?

Var4, var8, son públicos

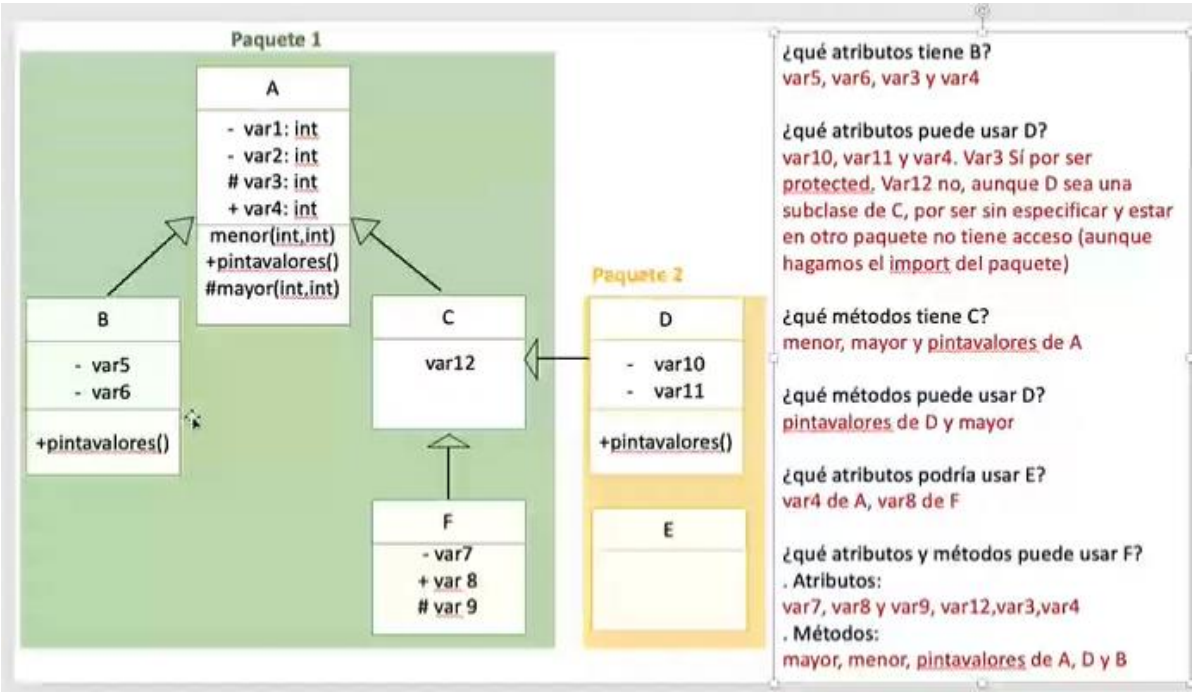
¿qué atributos y métodos puede usar F?

. Atributos:

Var7, var8, var9, va12, var3, var4 **(ATB)**

. Métodos:

Mayor, menor, pintavalores de A, D y B **(métodos)**



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 dara el valor true si son iguales y false si no.

Podemos sobreescribir 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 sobreescribe.

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

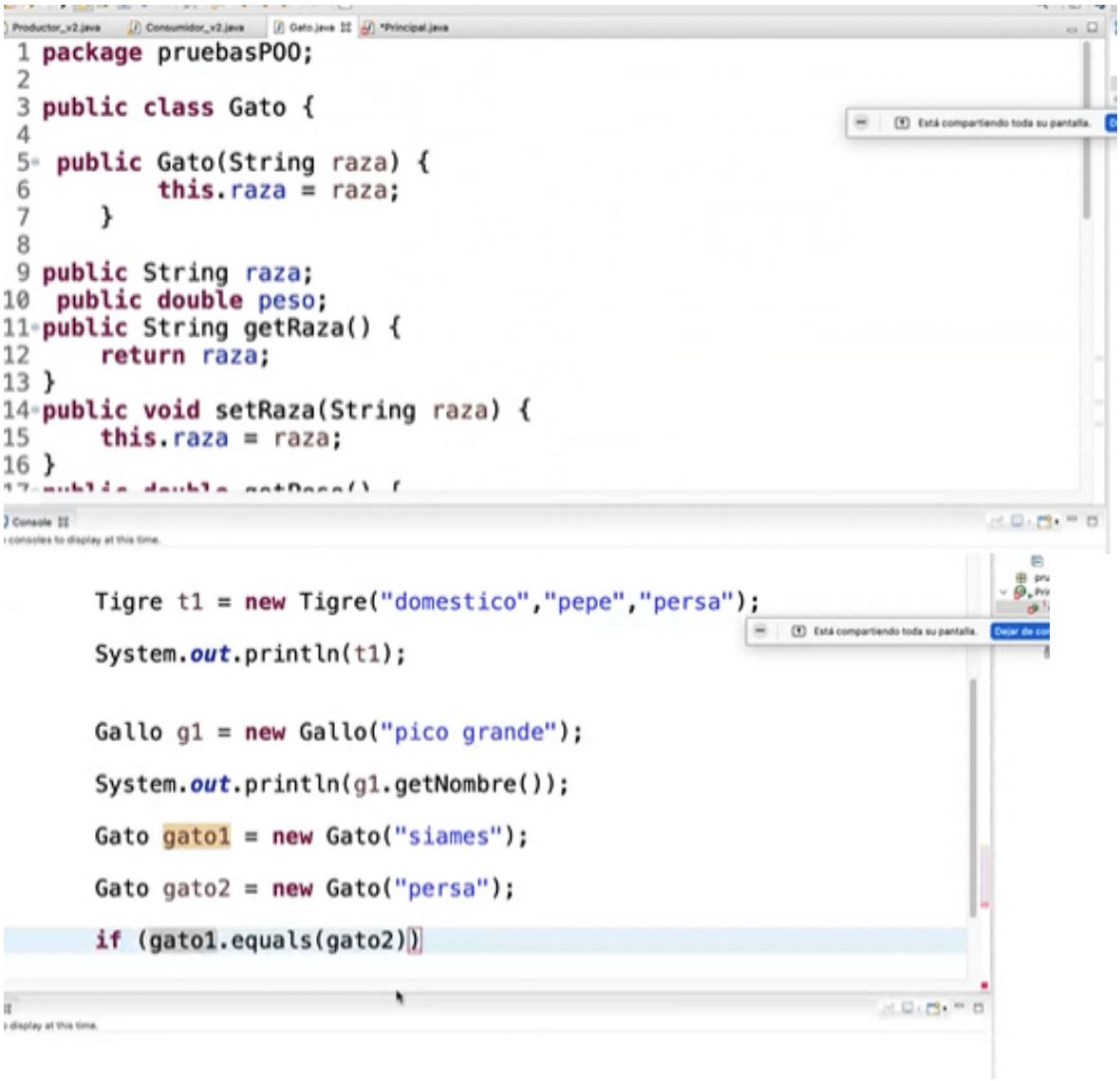
```
@Override
      public boolean equals(Object obj) {

            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
```



```java
package pruebasPOO;

public class Gato {

 public Gato(String raza) {
        this.raza = raza;
    }

public String raza;
 public double peso;
public String getRaza() {
    return raza;
}
public void setRaza(String raza) {
    this.raza = raza;
}
```

```java
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))
```

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 16/01/2022

HUEVO

```
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();



	}

}
```

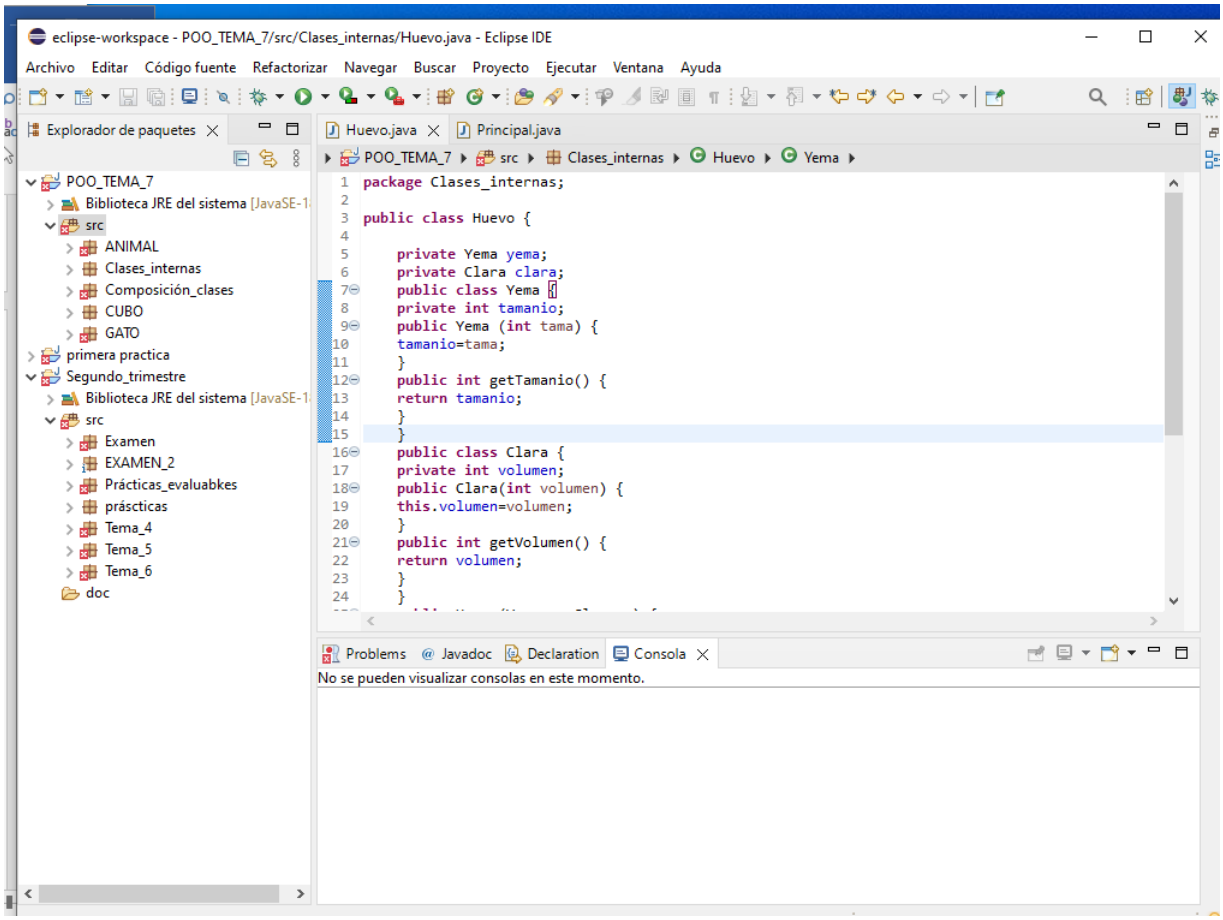
Clases internas Ejemplo: Huevo, Yema y Clara 23/01/2023

HUEVO

```
package Clases_internas;

public class Huevo {

	private Yema yema;
	private Clara clara;
	public class Yema {
	private int tamanio;
	public Yema (int tama) {
	tamanio=tama;
	}
	public int getTamanio() {
	return tamanio;
	}
	}
	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 tamanioYema, int volumenClara) {
	 this.yema = new Yema(tamanioYema);
	 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);

	}

}
```

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

NODO

NODO 25/01/2023

```
package NODO;

public class Nodo {

	public int info;
	public Nodo nodo_siguiente;

}
```

LISTA

```
package NODO;

public class Lista {

	public Nodo raiz;
	public Lista() {
		raiz = null;
	}

	//insertar en la primera posición
	public  void insertaPrimero (int x) {

		Nodo nuevo = new Nodo();
		nuevo.info = x;

		if(raiz == null) {
			//nuevo.info = 4;
			raiz = nuevo;
		}
		else {
			//nuevo.info=7;
			nuevo.nodo_siguiente = raiz;
			raiz = nuevo;
		}

	}

	//insertar en la última posición
	public void insertarUltimo(int x) {
		Nodo nuevo = new Nodo();
		nuevo.info = x;
		Nodo busca = raiz;

		while (busca.nodo_siguiente != null) {
			busca = busca.nodo_siguiente;
		}

		busca.nodo_siguiente = nuevo;
	}

	//mostar lista
	public void mostrarLista(int x) {
		Nodo uno = raiz;
	while (uno!=null) {
		System.out.print(uno.info + "->");
		uno = uno.nodo_siguiente;
```

```
	}
	System.out.println("");
	}

	//Mirar si está o no un nº en la lista
	//meter un if en el while anterior

	/*public boolean comprobarLista(int x) {
		Nodo aux = raiz;
		return  aux.equals(raiz.getClass()) &&
raiz.equals(aux.getClass());
		while(aux !=null ) {
			if (aux.info == x) {
				num = true;
			}
			else {
				num = false;
			}
		}
	}*/


	//método borrar
	public void borrarNumero(int x) {
		Nodo aux = raiz;
		Nodo previo = raiz;
		while (aux!=null) {
			if (aux.info == x) {
				previo.nodo_siguiente = aux.nodo_siguiente;
			}
			previo = aux;
			aux = aux.nodo_siguiente;
			raiz = raiz.nodo_siguiente;
		}

	}
	//metodo primero: muestra el valor del primer elemento de la lista
(nodo raíz)
	public void primero() {
		Nodo aux = raiz;
		aux.info = raiz.info;
	}

	//muestra el valor del último elemento de la lista (que tenga como
siguiente null)
	public void ultimo() {
		Nodo aux = raiz;
		Nodo ult = raiz;
		if(aux != null) {
			aux.nodo_siguiente = ult;
		}
		aux = ult;
		ult.nodo_siguiente = raiz;
	}
	//borra el primero
	public void borrarPrimero(int x) {
		Nodo aux = raiz;
		Nodo prim = raiz;
		while (aux != null) {
```

```
                    if(aux.info == x) {
                          prim.nodo_siguiente = aux;
                    }
                    prim = aux;
                    raiz = aux;
              }
        }
        //Borra el último
        public void borrarUltimo(int x) {
              Nodo aux = raiz;
              Nodo previo = raiz;
              while(aux != null) {
                    if(aux.info == x) {
                          previo.nodo_siguiente = aux.nodo_siguiente;
                    }

                    previo = aux;
                    aux = aux.nodo_siguiente;
              }
        }
      //inserta un nodo con el valor x en la posicion dada por parámetro
        public void insertaPos (int posicion, int x){

        }
        //size:tamaño lista
        public void size() {

        }

        //MAIN
        public static void main(String[] args) {
               Lista lista = new Lista();

               lista.insertaPrimero(4);
               lista.insertaPrimero(7);
               lista.insertaPrimero(6);


               lista.borrarNumero(4);
               //lista.comprobarLista(7);
              lista.primero();
              lista.ultimo();
               lista.borrarPrimero(6);
               lista.borrarUltimo(6);
               lista.mostrarLista(0);
               System.out.println();
        }

}
```

PRINCIPAL

```
package NODO;

public class Prueba {

      public static void main(String[] args) {


              Nodo a = new Nodo();
              a.info = 4;
```

```
            System.out.println(a.info);


            Nodo b = new Nodo();
            b.info = 7;

            Nodo c = new Nodo();
            c.info = 6;

            a.nodo_siguiente = b;
            b.nodo_siguiente = c;


            //insertarPrimero(4);
            //insertarPrimero(7);
            //insertarUltimo(6);



      }


}
```

INSTANCEOF

CRICULO

```
package Instanceof;

public class Circulo extends Figura {

}
```

CUADRADO

```
package Instanceof;

public class Cuadrado extends Figura {

                                    }
```

FIGURA

```
package Instanceof;

public class Figura {

}
```

PRINCIPAL

```
package Instanceof;

public class Principal {

      public static void main(String[] args) {

            Figura[] figuras = new Figura [3];
```

```
		figuras[0] = new Circulo ();
		figuras[1] = new Cuadrado();
		figuras[2] = new Circulo();

		for(int i=0; i<figuras.length; i++) {
			if(figuras[i] instanceof Circulo) {
				System.out.println("Figura en el indice: " + i + "
es un circulo");
			}
			else if (figuras[i] instanceof Cuadrado) {
				System.out.println("Figura en el indice: " + i + "
es uncuadrado");
			}
		}
	}

}
```

WRAPPER
EJEMPLO

```
package Wrappers;

public class ejemplo {

	public static void main(String[] args) {

		int a = 4;
		Integer b=5;

		String num = " 6 ";
		System.out.println(num.trim());
		String num2 = num.trim();

		int c = Integer.parseInt(num2);
		//Integer c = Integer.parseInt(num2);
		System.out.println(c+3);

		//System.out.println(c);

		Integer d = Integer.valueOf(num2);
		System.out.println(d);

		int e = Integer.valueOf(num2).intValue();
		System.out.println(e);

		Integer.max(a, b);

	}

}
```

ABSTRACT

AUTOMOVIL

```
package Clase_abstracta_abstract;

public abstract class Automovil {

	String marca;

	public abstract void eslogan();
	public String getMarca() {
		return marca;
	}
}
```

DEPORTIVO

```
package Clase_abstracta_abstract;

public class Deportivo extends Automovil {

	public void eslogan() {
		System.out.println ("Veloz como el rayo");
	}
}
```

FAMILIAR

```
package Clase_abstracta_abstract;

public class Familiar extends Automovil{

	public void eslogan() {
		System.out.println ("Cabe hasta el gato");
	}
}
```

TURISMO

```
package Clase_abstracta_abstract;

public class Turismo extends Automovil {

	public void eslogan() {
		System.out.println ("Para el uso diario");
	}
}
```

PRINCIPAL

```
package Clase_abstracta_abstract;
```

```
public class Principal {

	public static void main(String[] args) {


		Automovil [] auto = new Automovil [3];
		auto [0] = new Deportivo();
		auto [1] = new Turismo();
		auto[2] = new Familiar();

		imprime_eslogan(auto);

	}

	private static void imprime_eslogan(Automovil[] auto) {

		for (int i=0; i<=2; i++) {
			auto[i].eslogan();
		}

	}

}
```

INTERFACES
EMPLEADO

```
package Interfaces_2;

public class empleado implements Comparable {

/*Ordenar el array de empleados de 3 formas

* id de empleado

* nombre

* por id y en segundo lugar por nombre, si dos coinciden alfabeticamente*/

private int id_empleado;

private String nombre;

public empleado(int id_empleado, String nombre) {

super();

this.id_empleado = id_empleado;

this.nombre = nombre;

}

//getter y setter

public int getId_empleado() {

return id_empleado;

}

public void setId_empleado(int id_empleado) {

this.id_empleado = id_empleado;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;
```

```
}
//toString
@Override
public String toString() {
return "empleado [id_empleado=" + id_empleado + ", nombre=" + nombre + "]";
}
@Override
public int compareTo(Object jueves) {
empleado otro = (empleado) jueves;
if(this.id_empleado == otro.id_empleado) {
return nombre.compareTo(otro.nombre);
}
else if(this.id_empleado > otro.id_empleado) {
return 1;
}
else {
return -1;
}
//return this.id_empleado - otro.id_empleado;
//return nombre.compareTo(otro.nombre);
}
}
```

MAIN

```
package Interfaces_2;

import java.util.*;

public class Principal {

public static void main(String[] args) {

empleado e1 = new empleado (7, "Pepe");

empleado e2 = new empleado (2, "Paco");

empleado e3 = new empleado (3, "Maria");

empleado e4 = new empleado (3, "Ana");

empleado e5 = new empleado (5, "Susana");

empleado arrayEmpleados[] = {e1, e2, e3, e4, e5};

System.out.println("Antes de ordenar:");

System.out.println(Arrays.toString(arrayEmpleados));

//ordenaID(arrayEmpleados);

//ordenaNombre(arrayEmpleados);

Arrays.sort(arrayEmpleados);

System.out.println("Despues de ordenar:");

System.out.println(Arrays.toString(arrayEmpleados));

}

//ordenar por ID

static void ordenaID(empleado[] arrayAOrdenar) {

empleado aux;

for(int i = 1; i<arrayAOrdenar.length ; i++) {

for (int j = 1; j <arrayAOrdenar.length; j++) {

if(arrayAOrdenar[j].getId_empleado() > arrayAOrdenar[j+1].getId_empleado()) {

aux = arrayAOrdenar [j];

arrayAOrdenar [j] = arrayAOrdenar [j + 1];
```

```
arrayAOrdenar [j + 1] = aux;

}

}

}

}

/*generar variable auxiliar para intercambiar

* utilizar burbuja

* recorrer las posiciones del array excepto la última

* si el id es mayor que el anterior se intercambian las posiciones

*/

//ordenar por nombre

static void ordenaNombre(empleado[] arrayAOrdenar) {

empleado aux;

for(int i = 1; i<arrayAOrdenar.length ; i++) {

for (int j = 1; j <arrayAOrdenar.length; j++) {

if(arrayAOrdenar[j].getNombre().compareTo(arrayAOrdenar[j+1].getNombre()) >=
0) {

aux = arrayAOrdenar [j];

arrayAOrdenar [j] = arrayAOrdenar [j + 1];

arrayAOrdenar [j + 1] = aux;

}

}

}

}

//ordenar primero por id y en segundo lugar por nombre, si dos coinciden
alfabeticamente

//public static empleado[]

}
```

WRAPPER

Ejercicio con Wrappers

- Desarrolla un método leeNumero() para leer una cadena de texto por teclado y convertir dicha cadena a numero.
- Llamar al método getUltimaCifra indicando como parámetro el número que devuelva el método anterior.
  Este método:
  - pasa el número a String y obtiene la última cifra.
  - convierte esa cifra a entero y la multiplica por 2
- Desde el programa main se mostrará por consola el resultado de la operación con la siguiente instrucción

```
System.out.println(getUltimaCifra(leeNumero()));
```