Virrey Morcillo
Instituto de Educación Secundaria

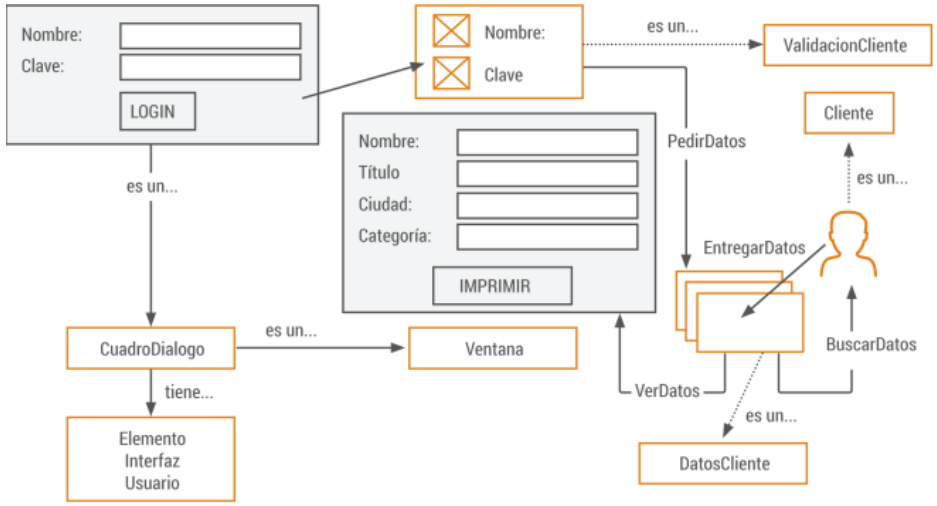
UT 1. POO, EDA y API JAVA.

Programación Orientada a Objetos

Virrey Morcillo
Instituto de Educación Secundaria

Orientación a Objetos

- Java es un lenguaje basado en la programación orientada a objetos.
- Un programa orientado a objetos es el resultado de la colaboración de varios tipos de objetos entre sí.
- Los objetos tiene sus funcionalidades y se las prestan los unos a los otros.



Virrey Morcillo
Instituto de Educación Secundaria

Orientación a Objetos

- La POO se basa en cuatro principios:
  - **Abstracción**: mecanismo que permite concentrarse en los aspectos necesarios. Una clase es una abstracción para un tipo concreto de objetos.
  - **Encapsulación:** mecanismo por el cual los datos y los códigos están protegidos de su acceso desde el exterior.
  - **Herencia:** mecanismos por el cual un objeto puede adquirir propiedades y funcionalidades de otro tipo de objetos. (superclases y subclases).
  - **Polimorfismo:** mecanismo por el cual con la misma forma se ejecutan códigos diferentes.

Virrey Morcillo
Instituto de Educación Secundaria

Clases, datos y métodos.

- Clases:
  - Definen la estructura y el comportamiento de los objetos de un tipo particular.
  - Son como modelos de objetos del mismo tipo.
  - Se implementan con la palabra reservada **class**.
  - En ella se definen las propiedades o atributos y los métodos.

```
<modificador_acceso> class <identificador>{
  <definicion_variables>
  <definicion_metodos>
```

Instituto de Educación Secundaria Virrey Morcillo

Clases, datos y métodos.

- Variables.
  - Las variables miembro dentro de la definición de una clase son de dos tipos:
    - Instancia:
      - Atributos que tiene cada objeto.
      - Creados cuando se crea el objeto y liberado cuando se liberan los objetos.
      - Espacios de memoria donde se almacenan los datos de cada objeto
    - Clase:
      - Atributos que comparten todos los objetos de una clase.
      - No pertenecen a ningún objeto, son de la clase.
      - Definidos utilizando la palabra **static**.
  - Se definen de manera similar a una variable estándar, añadiendo el modificador de acceso: **private, protected, public**.

```
<modificador_acceso> [static] <tipo> <identificador>;
o
<modificador_acceso> [static] <tipo> <identificador> =
<expresion_incializacion>
```

Virrey Morcillo
Instituto de Educación Secundaria

Clases, datos y métodos.

- Métodos.
  - Los métodos son las funciones miembro de la clase que definen el comportamiento de los objetos.
  - Los métodos pueden ser de dos tipos:
    - Instancia:
      - Funciones miembro que definen el comportamiento de los objetos.
      - Son llamadas para un objeto concreto.
      - Pueden acceder a los atributos del objeto (según la visibilidad del mismo).
    - Clase:
      - Funciones de la clase, no de los objetos.
      - Solamente pueden acceder a sus variables de clase.
      - Se definen usando la palabra **static.**

```
<modificador_acceso> <tipo> <identificador> ([<lista_
parametros>]){
 <sentencias>
}
```

Virrey Morcillo
Instituto de Educación Secundaria

Clases, datos y métodos.

- Modificadores de acceso.
  - Existen 4 modificadores de acceso, desde los cuales se define el ámbito desde el que se puede usar las clases, las variables y los métodos.

Modificadores de acceso	Accesibilidad		
	Clase	Variable	Método
private	Accesible sólo dentro del archivo o clase donde se define	Accesible dentro de la clase	
protected	No se aplica	Accesible dentro de la propia clase y funciones de clases que hereden (subclases)	
sin modificador	Accesible dentro de la propia clase y las clases que estén dentro de su mismo paquete (package)		
public	Accesible dentro de la propia clase y de las que en su archivo la importen (import)		

Virrey Morcillo
Instituto de Educación Secundaria

Creación de objetos.

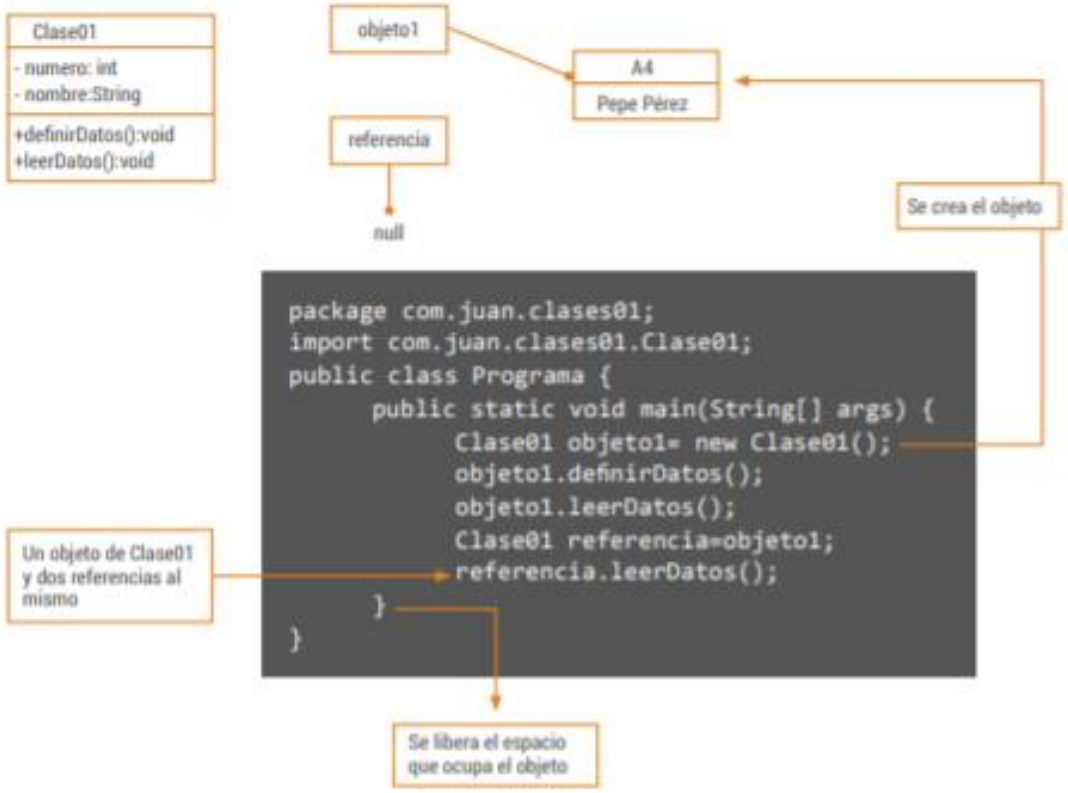
- Para crear objetos usamos el operador **new.**
- Nos devuelve una ***referencia*** al objeto instanciado.
- No es obligatorio inicializar las variables de instancia.
- Al crear el objeto se invoca el **método constructor.**
- Si no se define un método constructor, Java usa un constructor por defecto.

```
<nombre_clase> <identificador > = new <nombre_clase>
(<lista_parametros);
```

- El espacio que ocupa un objeto es liberado por el recolector de basura (Garbage Collector)
  - El ámbito del objeto acaba.
  - El objeto deja de estar referenciado. (**NullPointerException**)

Virrey Morcillo
Instituto de Educación Secundaria

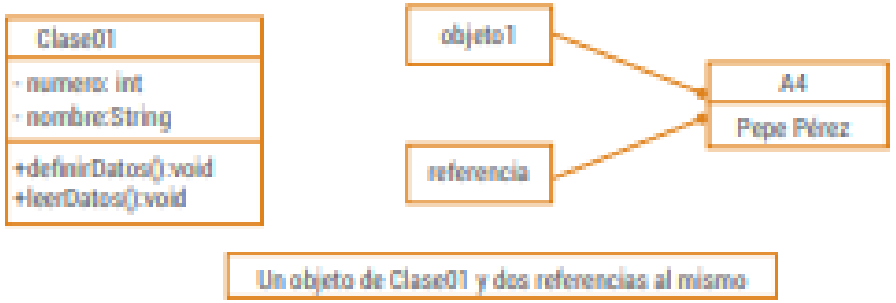
Creación de objetos.



Virrey Morcillo
Instituto de Educación Secundaria

Ejemplo de clase

```
package com.juan.clases01;        //paquete donde se
encuentra Clase01.java
import java.util.Scanner;     //para utilizar Scanner
public class Clase01 {
  private int numero; //variable de instancia privada
  private String nombre;      //variable de instancia
privada
  public void definirDatos(){ //método de acceso publico
          Scanner teclado = new Scanner(System.in);
          System.out.println("Teclea un nombre: ");
          nombre = teclado.nextLine();
          System.out.println("Teclea un numero: ");
          numero = teclado.nextInt();
  }
          public void leerDatos(){     //método de acceso
publico
          System.out.println("Nombre: " + nombre);
          System.out.println("Numero: " + numero);
  }
}
```



Un objeto de Clase01 y dos referencias al mismo

```
package com.juan.clases01;
import com.juan.clases01.Clase01;
public class Programa {
      public static void main(String[] args) {
            Clase01 objeto1= new Clase01();
            objeto1.definirDatos();
            objeto1.leerDatos();
            Clase01 referencia=objeto1;
            referencia.leerDatos();
      }
}
```

Virrey Morcillo
Instituto de Educación Secundaria

Constructores

- Los constructores son funciones que se ejecutan cuando se crean objetos.
- Los constructores tienen el mismo nombre que la clase, una lista de parámetros y no devuelven nada (**ni void**).
- En una clase se pueden definir tantos constructores como se desee.
- Si existen varios constructores se dice que los constructores están **sobrecargados**

Virrey Morcillo
Instituto de Educación Secundaria

Constructores

```
package com.juan.clases;

public class Clase02 {
	private int numero;
	private String nombre;

	//Constructor sin parámetros
	public Clase02 (){
		numero = 99;
		nombre ="anonimo";
	}
	//Constructor con dos parámetros
	public Clase02(int num, String cad){
		numero = num;
		nombre = cad;
	}
	//Constructor que recibe referencia objeto de
Clase02
	public Clase02(Clase02 o){
		numero= o.numero;
		nombre= o.nombre;
	}

	public void leerDatos(){
		System.out.println("Nombre: " +
```

```
nombre);
		System.out.println("Numero: " +
numero);
	}
```

- El objetivo principal de los constructores es inicializar de un modo determinado las variables de instancia para el objeto que se crea.
- El constructor por defecto solamente es efectivo mientras no se haya definido ningún otro.
- Los constructores tiene que estar en ámbito como public para que se pueda acceder a ellos.

Virrey Morcillo
Instituto de Educación Secundaria

La referencia this

- La palabra reservada **this** es una referencia al propio objeto para el que se esta ejecutando el código del método miembro o constructor.
- Solo puede usarse dentro de las funciones de instancia y constructores de la clase.
- Por omisión es como si todo el acceso a todas las variables de instancia dentro de las funciones de la clase fueran precedidas por this y el .

```
//Constructor con dos parámetros
	public Clase03(int numero, String nombre){
		this.numero = numero;
		this.nombre = nombre;
	}

	//Función que accede a las variables de instancia y cambia su
valor
	public void modificaDatos(String nombre, int numero){
		this.numero = numero;
		this.nombre = nombre;
	}
```

Instituto de Educación Secundaria Virrey Morcillo

Variables y métodos de clase: miembros static

- Los miembros de clase vienen precedidos del modificador **static.**
- Las variables static son espacios compartidos por todos los objetos de la clase. Pueden ser accedidos desde cualquier función miembro.
- Las funciones static son código de la clase. Solo pueden acceder a los miembros static de la clase .

```
package com.juan.clases;

public class Clase04 {
	private static int numPersonas; //variable de
clase

	private int numero;
	private String nombre;

	public static int cuentaPersonas(){
		return numPersonas;
	}
```

```
	//Constructor sin parámetros
	public Clase04 (){
		numPersonas++;
		numero = 99;
		nombre =”anonimo”;
	}
	//Constructor que recibe referencia objeto de
Clase02
	public Clase04(Clase04 o){
		numPersonas++;
		numero= o.numero;
		nombre= o.nombre;
	}
```

Virrey Morcillo
Instituto de Educación Secundaria

Sobrecarga de métodos.

- Se puede sobrecargar cualquier método.
- Podemos tener varias funciones que tengan el mismo nombre y que devolviendo el mismo tipo, se diferencien la lista de parámetros, en su número o en sus tipos.

```
public Clase06{
	private static int numPersonas; //variable de
clase
	final public static int EDAD_MAX= 65;
	private int numero;
	private String nombre;
	//Constructores

	//Función que accede a las variables de
instancia y cambia su valor
	//sobrecargada de 4 formas
```

```
	public void modificaDatos(String nombre, int
numero){
		this.numero = numero > EDAD_MAX ? EDAD_
MAX : numero;
		this.nombre = nombre;
	}
	public void modificaDatos(String nombre){

		this.nombre = nombre;
	}
	public void modificaDatos(int numero){
		this.numero = numero > EDAD_MAX ? EDAD_
MAX : numero;
	}
	public void modificaDatos(){
		this.nombre = "anonimo";
		this.numero = EDAD_MAX;
	}
	//Otras funciones
}
```

```
public class Programa06 {
	public static void main(String[] args) {
		Clase06 objeto1= new Clase06();
		objeto1.modificaDatos("Luis Ruiz", 72);

		Clase06 objeto2= new Clase06(18, "Pepe Perez");
		objeto2.modificaDatos(Clase06.EDAD_MAX);

		Clase06 objeto3= new Clase06(objeto2);
		objeto3.modificaDatos("Antonio Gil");

		Clase06 objeto4 = new Clase06(objeto1);
		objeto4.modificaDatos();
	}
}
```