POO

Eduardo Sánchez

Luis García

Objetivo

- Reforzar los conocimientos adquiridos en la materia
- Adquirir los conocimientos necesarios para materias posteriores

Programa

1. POO
2. Java
3. Datos
4. Clases
5. Objetos
6. Métodos
7. Encapsulamiento
8. Constructores
9. Herencia
10. Polimorfismo

Evaluación

- Mínimo 80% asistencia
- Trabajos 40%
- Examen 60%

Calendario

Lunes	Martes	Miércoles	Jueves	Viernes
POO	Objetos	Herencia	Aplicaciones	Aplicaciones
Java	Métodos	Polimorfismo	Aplicaciones	Aplicaciones
Tipos de Datos	Encapsulamiento	Aplicaciones	Aplicaciones	Examen
Clases	Constructores	Aplicaciones	Aplicaciones	Examen

Paradigma

- Conjunto de conocimientos científicos
- Imperantes en una época
- Formas de pensar y sentir de las personas
- Lugar y momento histórico

Paradigma

- Forma aceptada para resolver problemas
- Modelo de investigación y formación teórica
- Conjunto de métodos, reglas y generalizaciones
- Utilizado por científicos

Paradigma de programación

- Diversas formas de programar
- Estilos para programar
- Para resolver problemas mediante una computadora

Paradigmas de programación

- Imperativa
- Orientada a objetos
- Dinámica
- Dirigida por eventos
- Declarativa
- Funcional
- Lógica
- Con restricciones
- Multi-paradigma
- Leguaje específico

Paradigma orientado a objetos

- Abstracción de la realidad
- Implementar aplicaciones
- Resolver problemas usando lenguajes de programación

Características del POO

- Abstracción
- Encapsulamiento
- Herencia
- Polimorfismo
- Recolección de basura*

Lenguajes Orientados a Objetos

- ABAP
- ABL
- ActionScript
- ActionScript
- C Sharp (C#)
- Clarion
- Clipper
- D
- Object Pascal (Embarcadero Delphi)
- Gambas
- GObject
- Genie
- Harbour
- Eiffel
- Fortran 90/95
- Java
- JavaScript
- Lexico
- Objective-C
- Ocaml
- Oz
- R
- Pascal
- Perl
- PowerBuilder
- Processing
- Python
- Ruby
- Self
- Smalltalk
- Magik
- Vala
- VB.NET
- Visual FoxPro
- Visual Basic 6.0
- Visual DataFlex
- Visual Objects
- XBase++
- DRP
- Scala

Java

- Simple
- Sintaxis basada en C++
- *Garbage Collector*
- JVM
  - Lenguaje WORA
- Diversidad de bibliotecas
- Robusto

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	extends	false
final	finally	float	for	if
implements	import	instanceof	int	interface
long	native	new	null	package
private	protected	public	return	short
static	super	switch	syncronized	this
throw	throws	transient	true	try
void	volatile	while	**var**	**rest**
byvalue	**cast**	**const**	**future**	**generic**
goto	**inner**	**operator**	**outer**	

Tipo de operadores	Operadores
Operadores posfijos	`[ ] . ( parámetros) expr++ expr--`
Operadores unarios	`++expr --expr +expr -expr ~ !`
Creación o conversión	`New (tipo) expr`
Multiplicación	`* / %`
Suma	`+ -`
Desplazamiento*	`<< >> >>>`
Comparación	`< > <= >= instanceof`
Igualdad	`== !=`
AND a nivel bit	`&`
OR a nivel bit	`|`
XOR a nivel bit	`^`
AND lógico	`&&`
OR lógico	`||`
Condicional*	`? :`
Asignación	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Tipos de Datos

- Primitivos
- Objeto

Tipos Primitivos

Nombre	Tipo
`byte`	**Entero**
`short`	**Entero**
`int`	**Entero**
`long`	**Entero**
`float`	**Decimal simple**
`double`	**Decimal doble**
`char`	**Carácter**
`boolean`	**True/False**

Tipos Objeto

- Biblioteca estándar de Java (String, ArrayList, Scanner…)
- Definidos por el programador*
- Arreglos
- Envoltorio**

Clases

- Dato abstracto
- Atributos
- Métodos

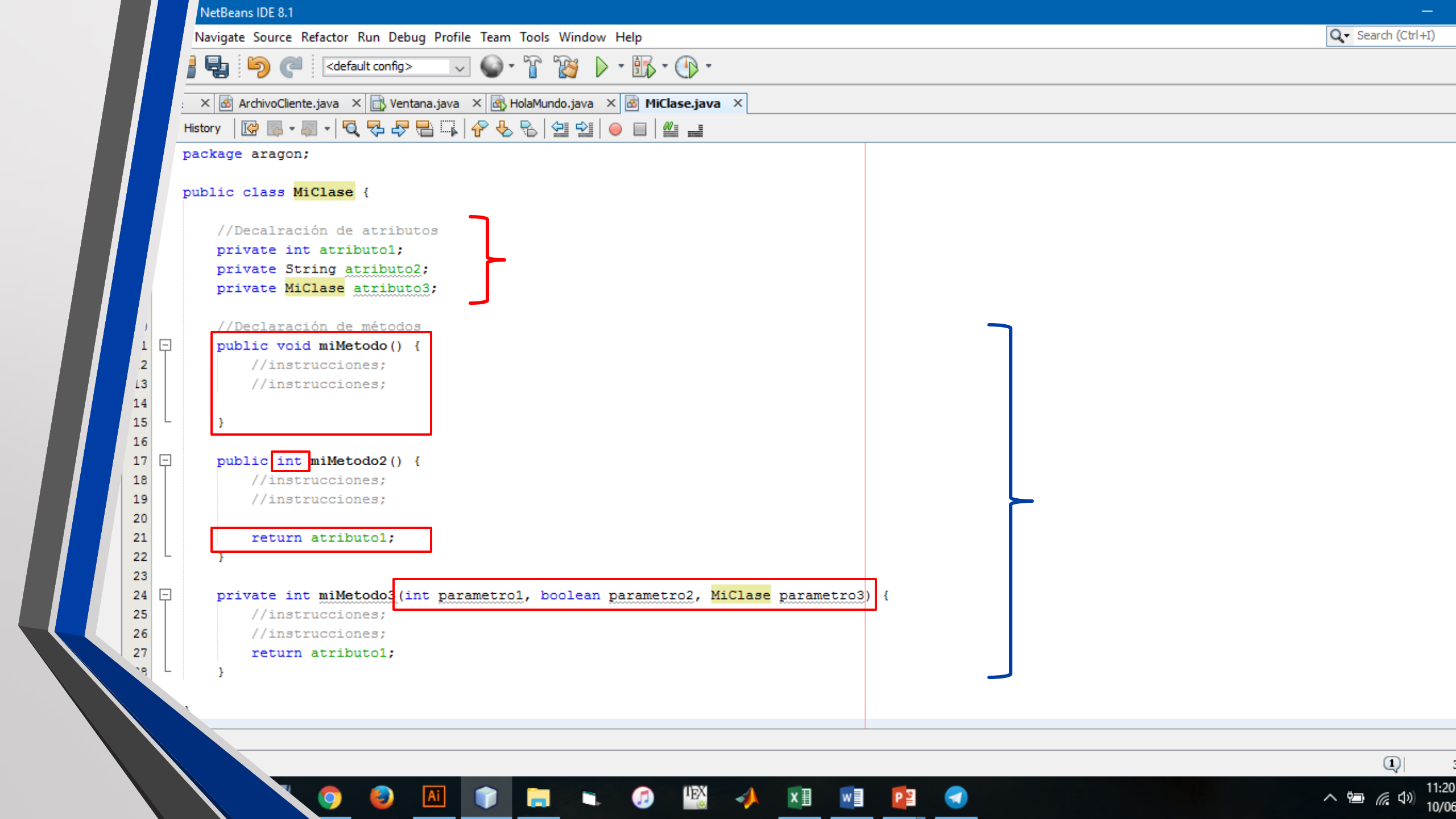
Clases

Atributos

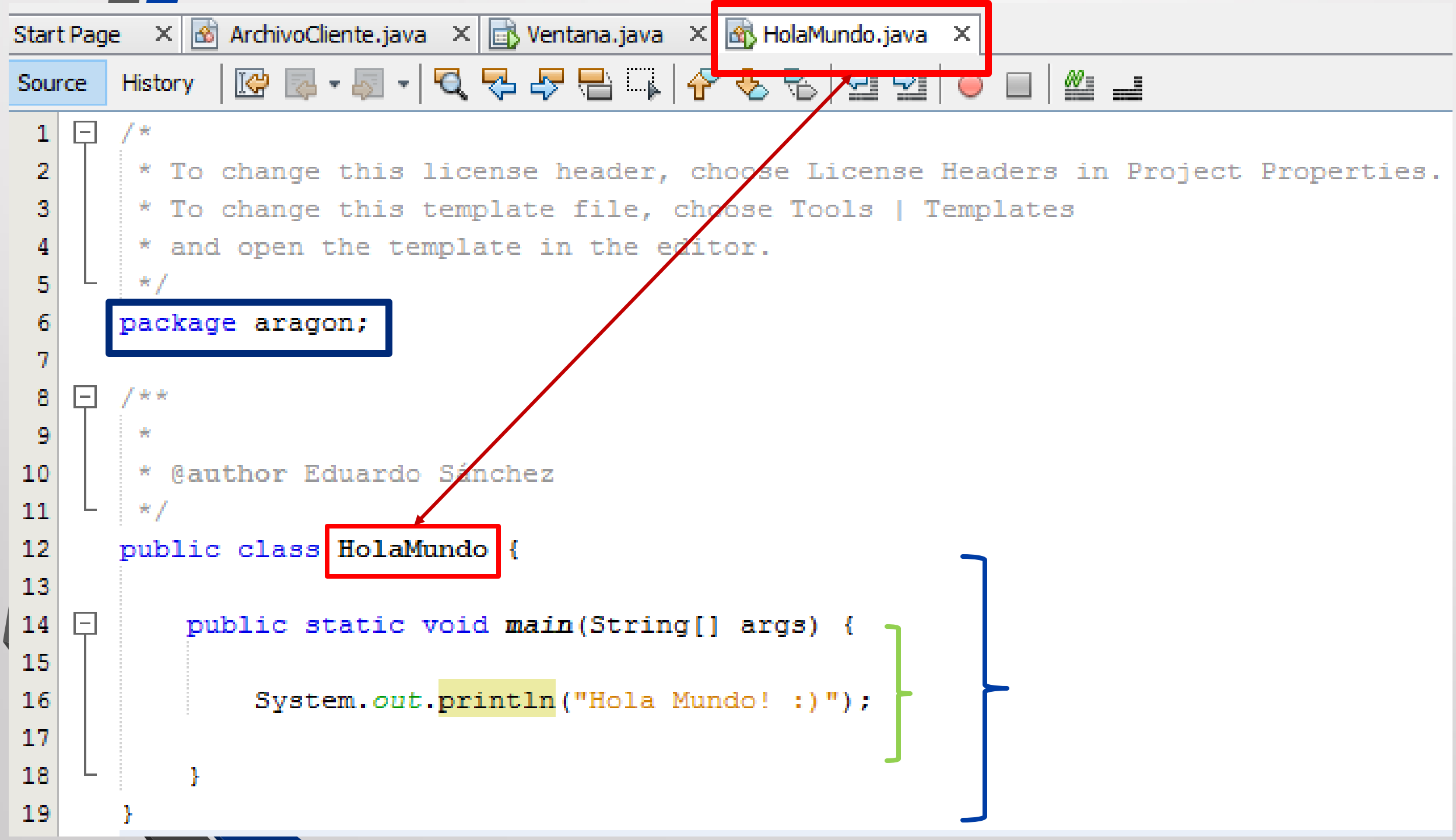
- Características de la clase
- Tipos primitivos
- Tipos objetos(clases)
- Visibilidad

Métodos

- Servicios de la clase
- Código necesario
- Visibilidad
- Retorno
- Nombre
- Parámetros


NetBeans IDE 8.1
Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
ArchivoCliente.java
Ventana.java
HolaMundo.java
MiClase.java
History
package aragon;
public class MiClase {
//Decalración de atributos
private int atributo1;
private String atributo2;
private MiClase atributo3;
//Declaración de métodos
public void miMetodo() {
//instrucciones;
//instrucciones;
}
public int miMetodo2() {
//instrucciones;
//instrucciones;
return atributo1;
}
private int miMetodo3(int parametro1, boolean parametro2, MiClase parametro3) {
//instrucciones;
//instrucciones;
return atributo1;
}

```
private int miMetodo(int parametro1, boolean parametro2) {
        //instrucciones ;
           if(parametro1 == 5){
               parametro2= true;
           }
        return atributo1;
    }
```


Start Page
ArchivoCliente.java
Ventana.java
HolaMundo.java
Source
History
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package aragon;
7
8 /**
9 *
10 * @author Eduardo Sánchez
11 */
12 public class HolaMundo {
13
14 public static void main(String[] args) {
15
16 System.out.println("Hola Mundo! :)");
17
18 }
19 }

NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects

- Abrir Archivos
- Burbuja2
- Cifrado
- Herramientas
- HolaMundo
  - Source Packages
    - aragon
  - Libraries
- IngenieriaEconomica
- OrdenamientoBurbuja
- Registro
- RSA
- RSA_version0.0
- RSA_version0.0_1
- Sistema RSA
- UNAM

Navigator

Members

- HolaMundo
  - main(String[] args)

Start Page | ArchivoCliente.java | Ventana.java | HolaMundo.java

Source History

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package aragon;
7
8  /**
9   *
10  * @author Eduardo Sánchez
11  */
12 public class HolaMundo {
13
14     public static void main(String[] args) {
15
16         System.out.println("Hola Mundo! :)");
17
18     }
19 }
20
```

Output - HolaMundo (run)

```
run:
Hola Mundo! :)
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output

20:1 INS

10:38 p. m.
10/06/2017

Objetos

- Referencia de clase
- Instancia ‘
- Espacio en memoria dinámica
- ‘ Objeto utilizable

Objetos

```
MiClase m;
```

Referencia

```
m = new MiClase();
```

Instancia

```
MiClase m = new MiClase();
```

Referencia

Instancia

Métodos

- Constructor
- Consultor
- Modificador
- Analizador

Métodos

Constructor

- Primero en ejecutarse'
- Inicializar atributos
- Publico
- Sin retorno

Consultor

- Retorna valor de un -atributo

Métodos

Constructor

```
public MiClase(int atributo1, MiClase atributo2) {
        this.atributo1 = atributo1;
        this.atributo2 = atributo2;
        this.atributo3 = false;
    }
```

Consultor

```
public int getAtributo1() {
        return atributo1;
    }
```

Métodos

Modificador

- Asignar valor a -atributos

Analizador

- Implementar lógica de servicio
- Algoritmos requeridos

Métodos

Modificador

```
public void setAtributo1(int atributo1) {
        this.atributo1 = atributo1;
    }
```

Analizador

```
public int calcularMayor() {
        if(this.atributo1 > this.atributo2){
            return this.atributo1;
        }else{
            return this.atributo2;
        }
    }
```

Encapsulamiento

- Ocultación del estado del objeto
- Modifica por métodos para el objeto
- Objetos aislado del exterior

Encapsulamiento

- Aplicación:
  - Conjunto objetos
  - Trabajan mutuamente
  - Métodos

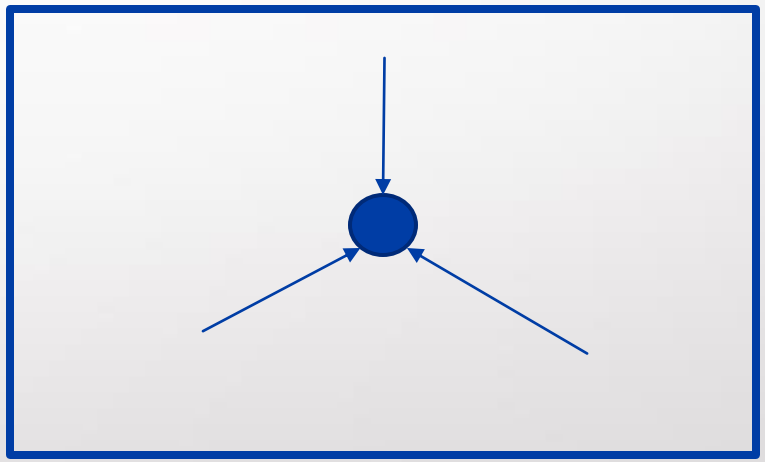
- Detalles ocultos
- Evitar modificaciones
- Accesos indebidos
- Una principal ventaja

Niveles de visibilidad

- Private
- Public
- Protected

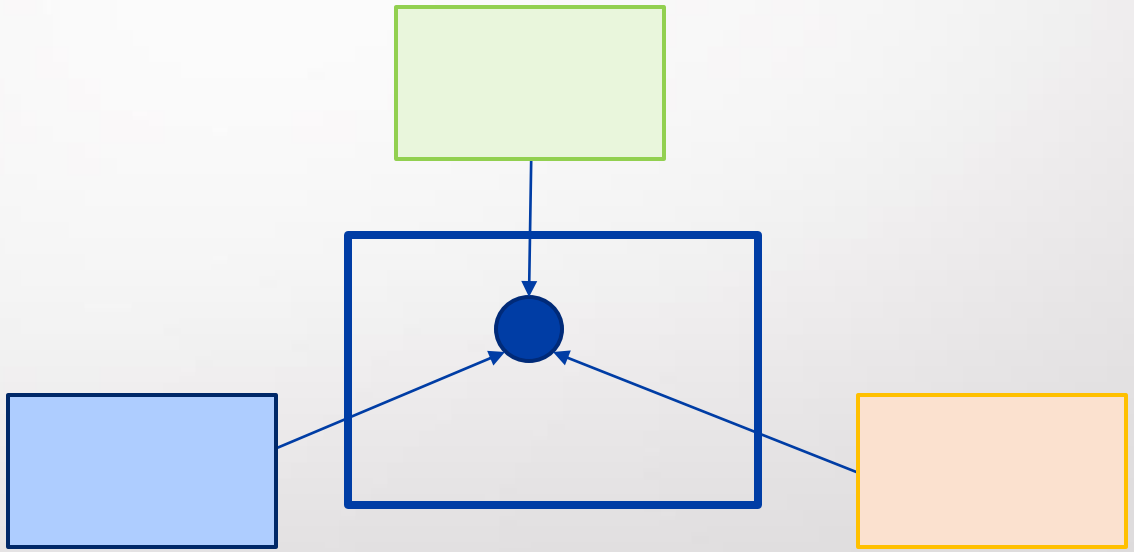
Private

- Solo por métodos en la clase



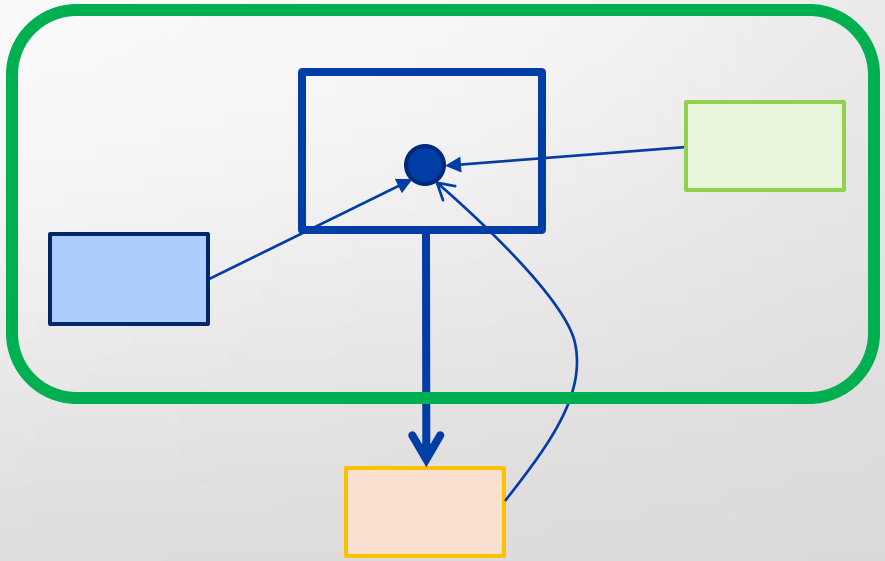
Public

- Métodos de cualquier clase



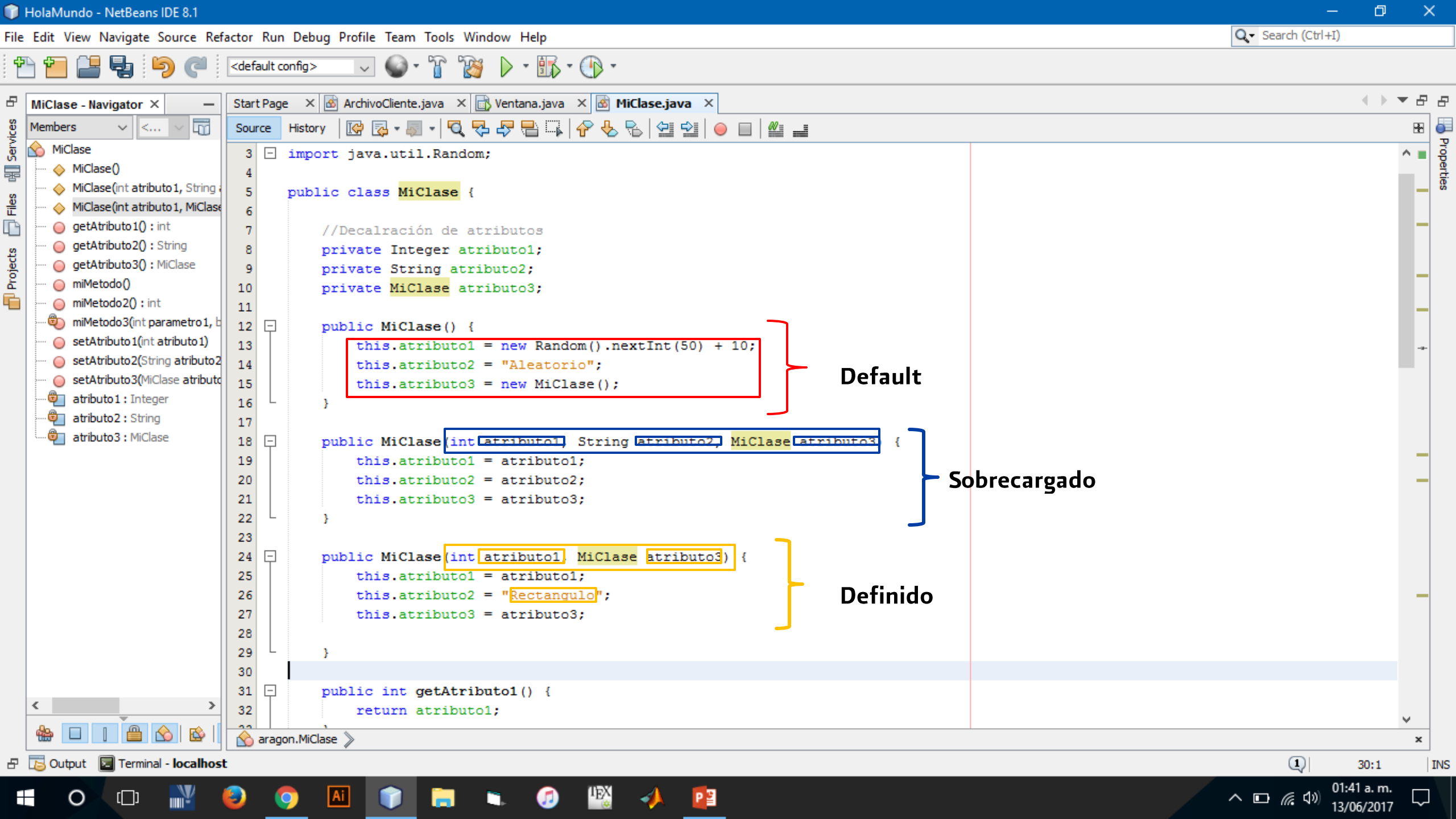
Protected

- Por métodos en clases *hijas*
- Por métodos en clases del mismo paquete



Constructores

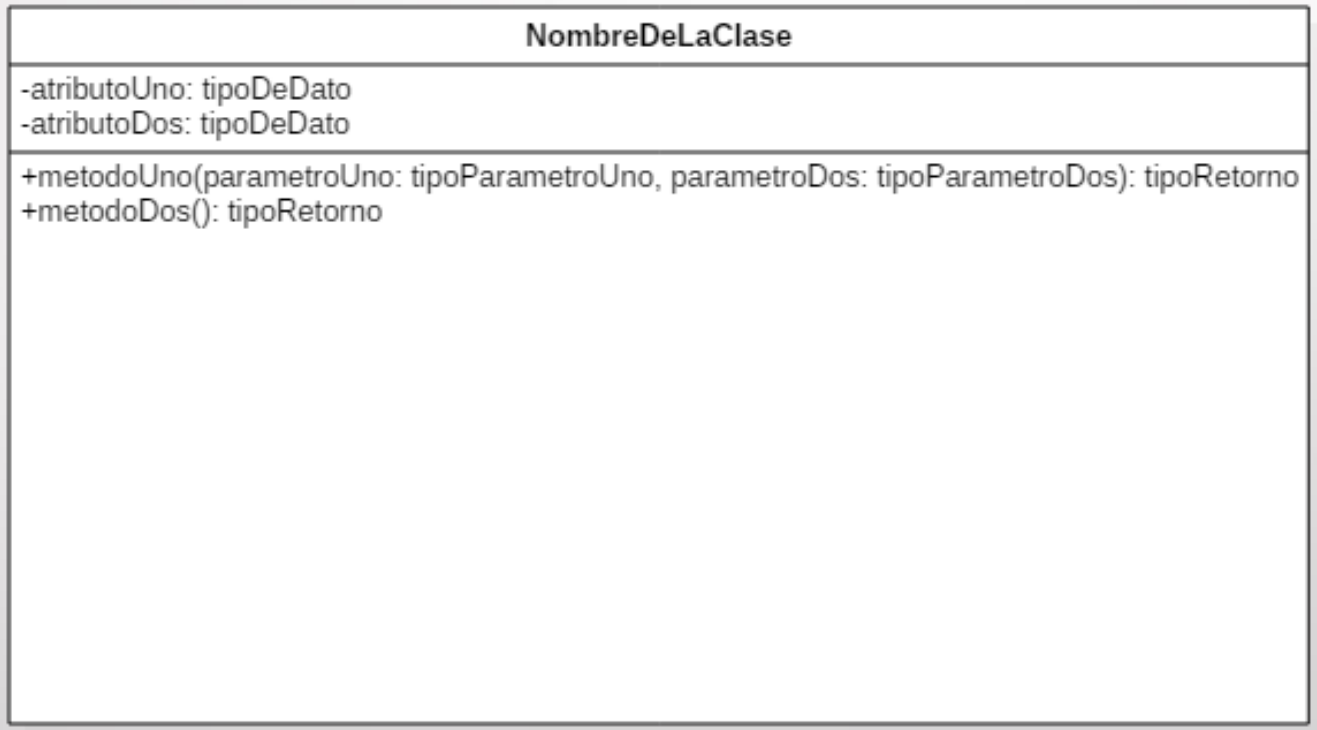
- Inicializador de atributos
- No tiene retorno
- Pueden existir varios
- Sólo se ejecuta uno
- Nombre igual a la clase

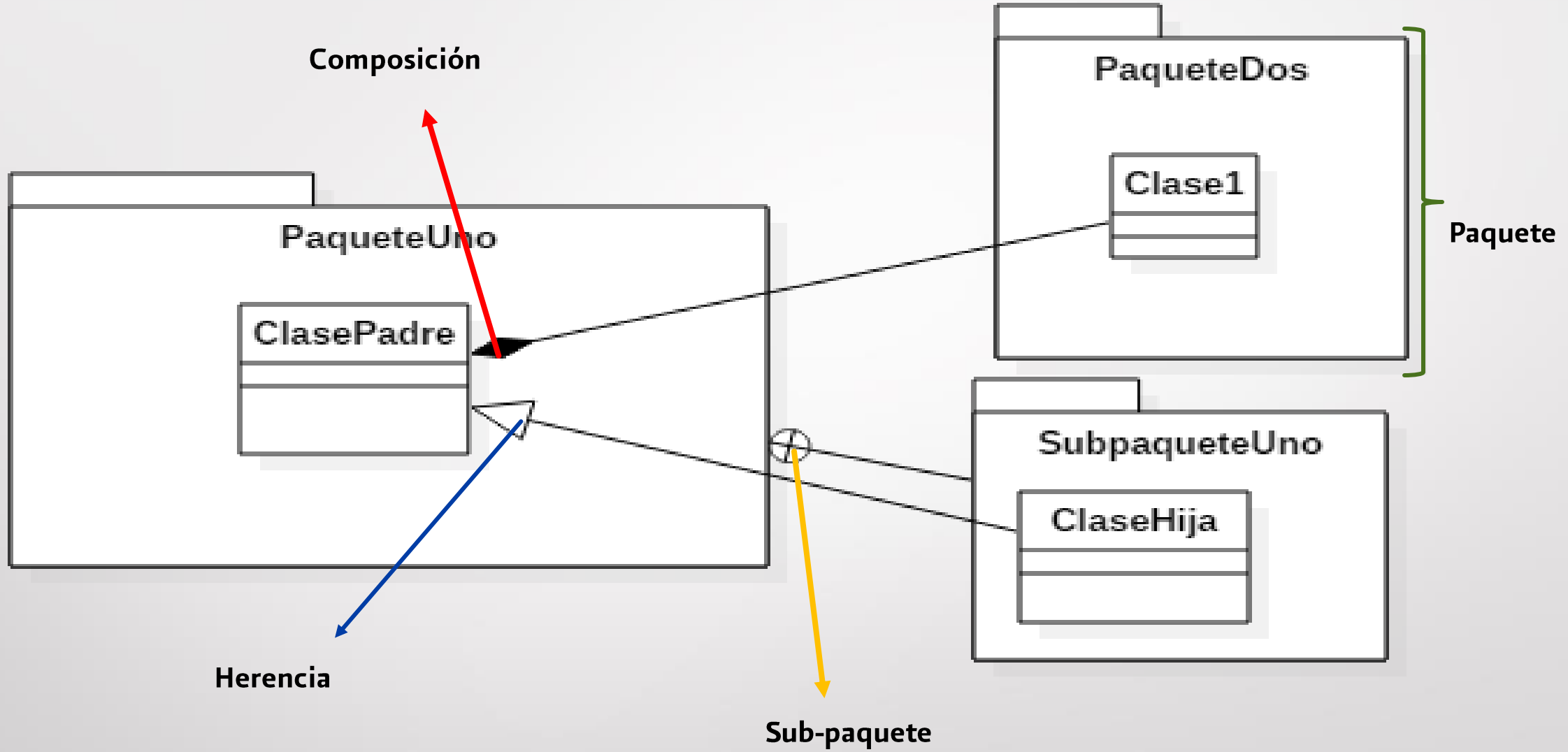

HolaMundo - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
MiClase - Navigator
Members
MiClase
MiClase()
MiClase(int atributo1, String
MiClase(int atributo1, MiClase
getAtributo1() : int
getAtributo2() : String
getAtributo3() : MiClase
miMetodo()
miMetodo2() : int
miMetodo3(int parametro1, b
setAtributo1(int atributo1)
setAtributo2(String atributo2
setAtributo3(MiClase atributo
atributo1 : Integer
atributo2 : String
atributo3 : MiClase
Start Page
ArchivoCliente.java
Ventana.java
MiClase.java
Source
History
3 import java.util.Random;
4
5 public class MiClase {
6
7 //Decalración de atributos
8 private Integer atributo1;
9 private String atributo2;
10 private MiClase atributo3;
11
12 public MiClase() {
13 this.atributo1 = new Random().nextInt(50) + 10;
14 this.atributo2 = "Aleatorio";
15 this.atributo3 = new MiClase();
16 }
17
18 public MiClase(int atributo1, String atributo2, MiClase atributo3) {
19 this.atributo1 = atributo1;
20 this.atributo2 = atributo2;
21 this.atributo3 = atributo3;
22 }
23
24 public MiClase(int atributo1, MiClase atributo3) {
25 this.atributo1 = atributo1;
26 this.atributo2 = "Rectangulo";
27 this.atributo3 = atributo3;
28
29 }
30
31 public int getAtributo1() {
32 return atributo1;
Default
Sobrecargado
Definido
aragon.MiClase
Output
Terminal - localhost
30:1
INS
01:41 a. m.
13/06/2017

UML

- Lenguaje Unificado de Modelado
- Visualizar
- Especificar
- Construir
- Documentar sistema
- “Plano”
- Definir sistema

UML Java

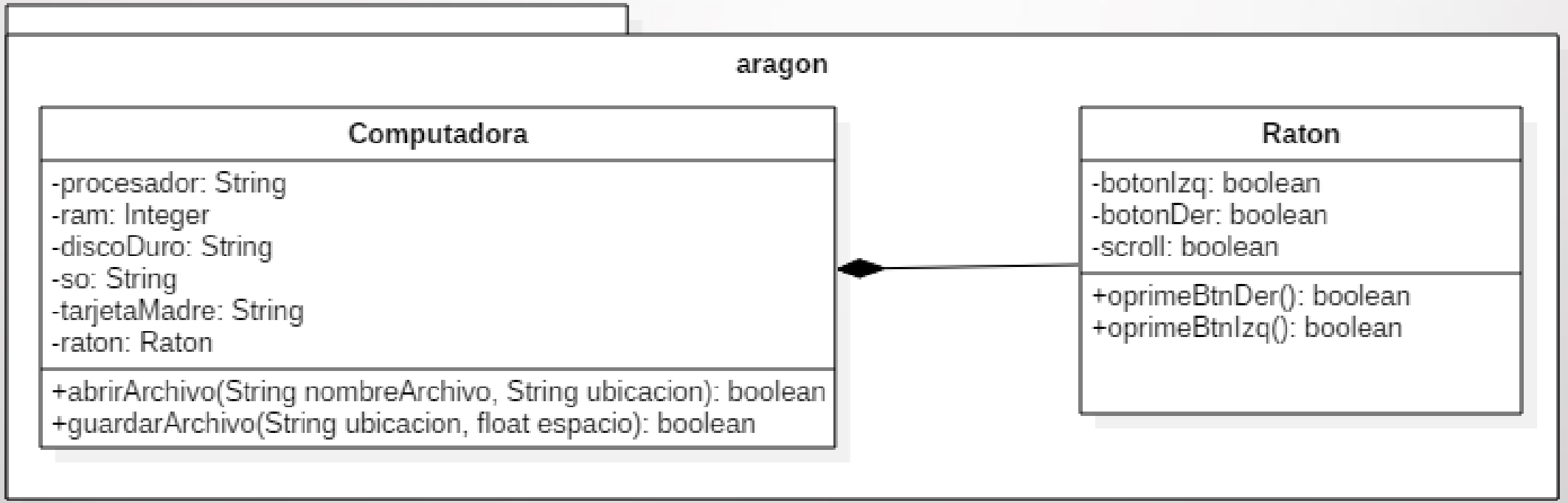

NombreDeLaClase
-atributoUno: tipoDeDato
-atributoDos: tipoDeDato
+metodoUno(parametroUno: tipoParametroUno, parametroDos: tipoParametroDos): tipoRetorno
+metodoDos(): tipoRetorno


Composición
PaqueteDos
Clase1
Paquete
PaqueteUno
ClasePadre
SubpaqueteUno
ClaseHija
Herencia
Sub-paquete

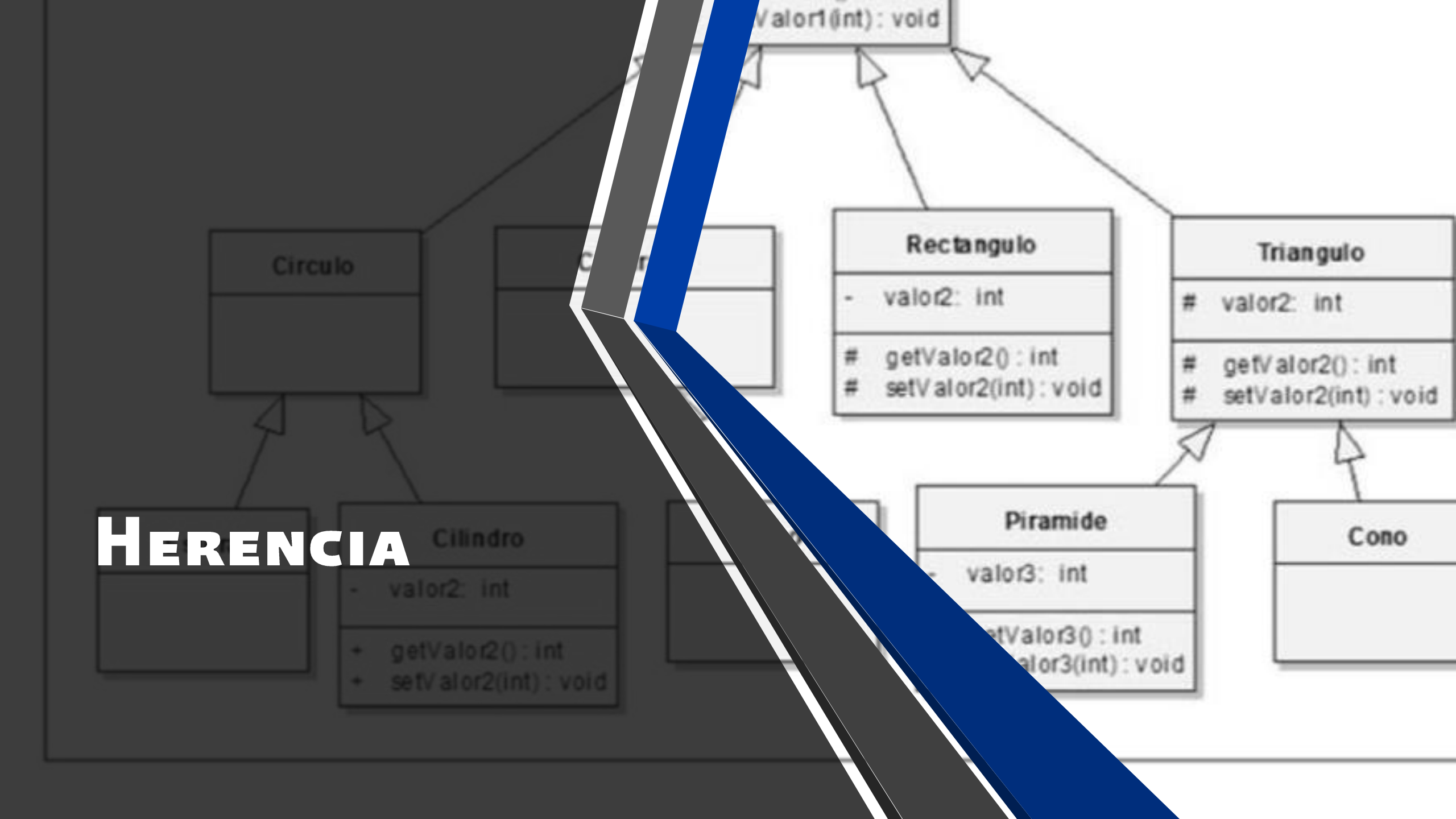
Composición

- Relación dependiente
- Objeto complejo:
  - Objetos pequeños
- “Tiene un”
- Objetos de CC deben existir primero
- La CC ni puede existir sin ellos

Composición



Herencia


Valor1(int) : void
Circulo
Rectangulo
- valor2: int
getValor2() : int
setValor2(int) : void
Triangulo
valor2: int
getValor2() : int
setValor2(int) : void
Cilindro
- valor2: int
+ getValor2() : int
+ setValor2(int) : void
Piramide
- valor3: int
Cono

Herencia

- Definir nuevas clases basadas en otras
- Reutilizar código
- En Java solo se puede heredar de una clase
- En otros lenguajes existe *herencia múltiple*

Herencia

- Clase *hija* hereda atributos y métodos
- Nuevos atributos
- Nuevos métodos
- Redefinir atributos
- Redefinir métodos

Herencia

```
 public class Persona{
     //Declaración de atributos

     //Declaración de métodos
 }


public class Profesor extends Persona{
    //Declaración de atributos

    //Declaración de métodos
}
```

super

- **Acceder a atributos**
- **Acceder a métodos**
- **Clase superior**

```java
package aragon;

public class Persona {

    private String nombre;
    private String apellido;
    private String correo;
    private Integer edad;

    public Persona() {
    }

    public Persona(String nombre, String apellido, String correo, Integer edad) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.correo = correo;
        this.edad = edad;
    }

}
```

```java
package aragon;

public class Estudiante extends Persona {

    private Integer numeroDeCuenta;
    private String facultad;

    public Estudiante() {
    }

    public Estudiante(Integer numeroDeCuenta, String facultad) {
        this.numeroDeCuenta = numeroDeCuenta;
        this.facultad = facultad;
    }

    public Estudiante(Integer numeroDeCuenta, String facultad, String nombre, String apellido, String correo, Integer edad) {
        super(nombre, apellido, correo, edad);
        this.numeroDeCuenta = numeroDeCuenta;
        this.facultad = facultad;
    }

}
```

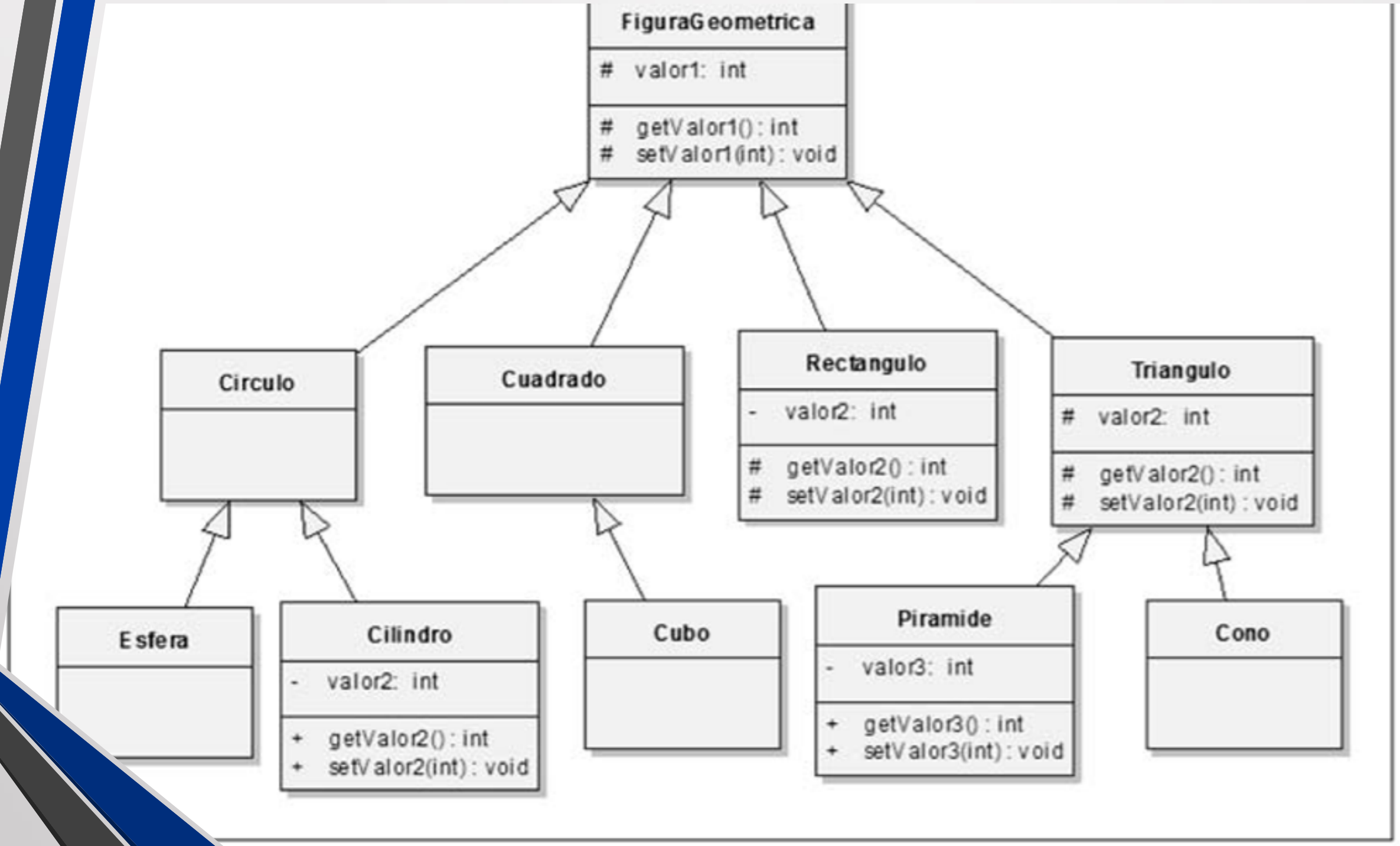
Sobre-escritura de métodos

- Característica de la herencia
- Implementar un método en clase padre y clase hija

Clases Abstractas

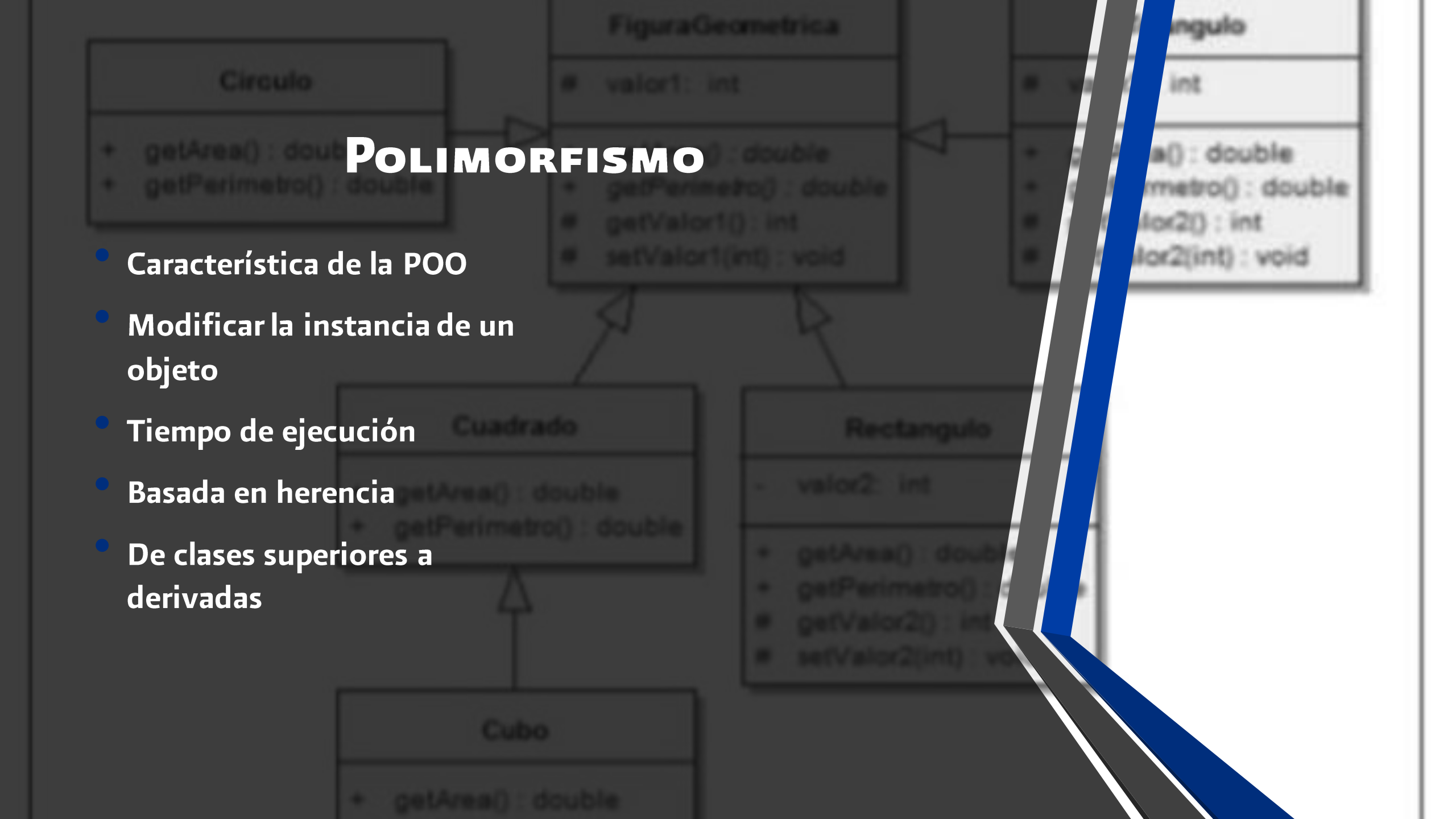
- No puede ser instanciada
- No se pueden crear objetos
- Puede heredar

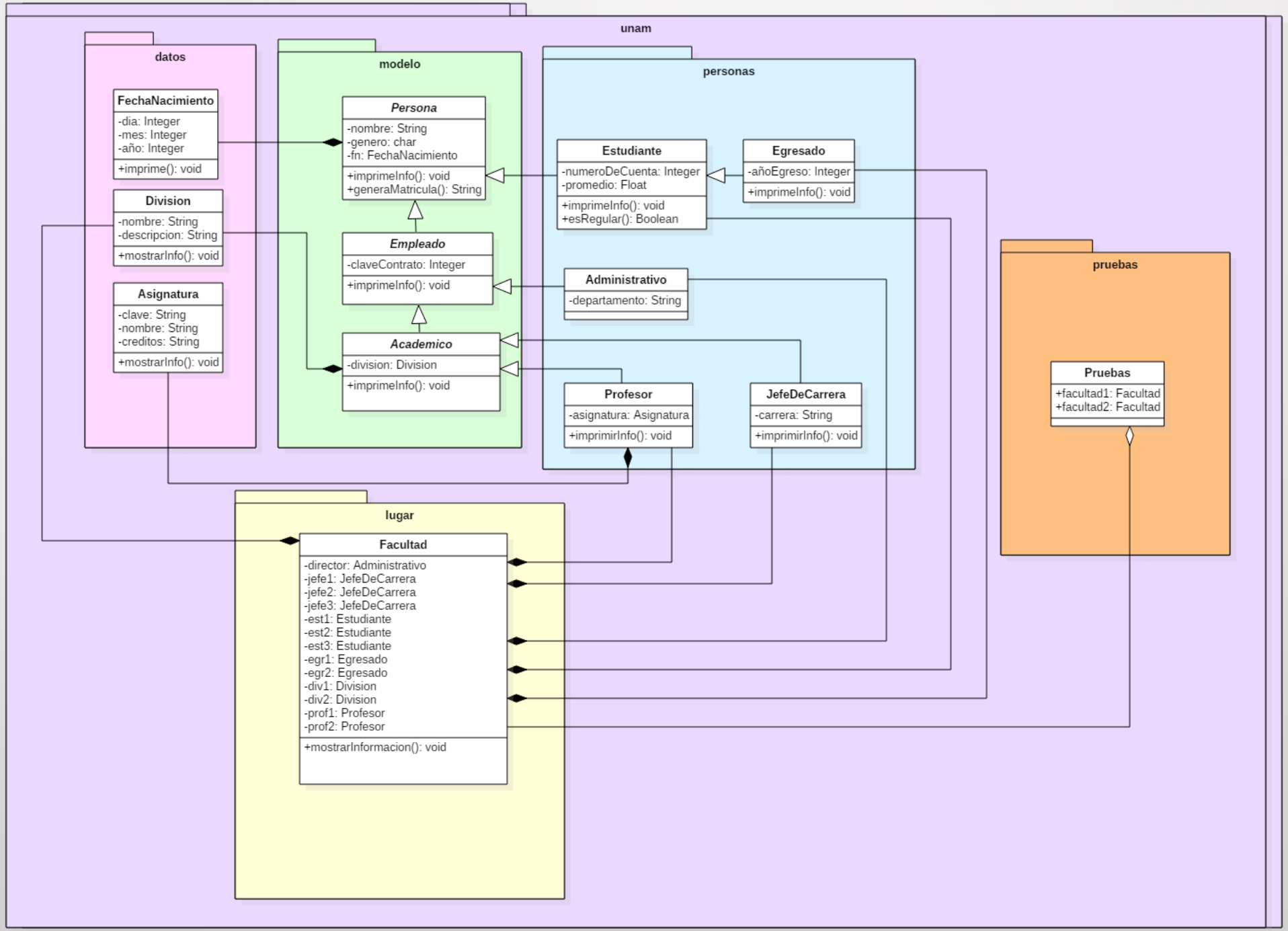
```
public abstract class Persona{
    …
}
```


FiguraGeometrica
valor1: int
getValor1() : int
setValor1(int) : void
Circulo
Cuadrado
Rectangulo
- valor2: int
getValor2() : int
setValor2(int) : void
Triangulo
valor2: int
getValor2() : int
setValor2(int) : void
Esfera
Cilindro
- valor2: int
+ getValor2() : int
+ setValor2(int) : void
Cubo
Piramide
- valor3: int
+ getValor3() : int
+ setValor3(int) : void
Cono

Polimorfismo

- Característica de la POO
- Modificar la instancia de un objeto
- Tiempo de ejecución
- Basada en herencia
- De clases superiores a derivadas




unam
datos
FechaNacimiento
-dia: Integer
-mes: Integer
-año: Integer
+imprime(): void
Division
-nombre: String
-descripcion: String
+mostrarInfo(): void
Asignatura
-clave: String
-nombre: String
-creditos: String
+mostrarInfo(): void
modelo
Persona
-nombre: String
-genero: char
-fn: FechaNacimiento
+imprimeInfo(): void
+generaMatricula(): String
Empleado
-claveContrato: Integer
+imprimeInfo(): void
Academico
-division: Division
+imprimeInfo(): void
personas
Estudiante
-numeroDeCuenta: Integer
-promedio: Float
+imprimeInfo(): void
+esRegular(): Boolean
Egresado
-añoEgreso: Integer
+imprimeInfo(): void
Administrativo
-departamento: String
Profesor
-asignatura: Asignatura
+imprimirInfo(): void
JefeDeCarrera
-carrera: String
+imprimirInfo(): void
pruebas
Pruebas
+facultad1: Facultad
+facultad2: Facultad
lugar
Facultad
-director: Administrativo
-jefe1: JefeDeCarrera
-jefe2: JefeDeCarrera
-jefe3: JefeDeCarrera
-est1: Estudiante
-est2: Estudiante
-est3: Estudiante
-egr1: Egresado
-egr2: Egresado
-div1: Division
-div2: Division
-prof1: Profesor
-prof2: Profesor
+mostrarInformacion(): void