Programación Avanzada

P. O. O.

Java™

Programación Orientada a Objetos

- **Es un paradigma de programación**

- **Se basa en la implementación de un algoritmo utilizando objetos de distintas clases que interactúan entre sí.**

Programación Orientada a Objetos

- **Utilizado en el desarrollo de software, para aplicaciones grandes y complejas**

- **Permite crear sistemas más robustos, mantenibles y escalables**

Clase

- **Plantilla o molde que define:**
  - **las características(atributos)**
  - **los comportamientos (métodos)**

de un grupo de objetos

Clase

- supongamos que vamos a construir un edificio…
- La clase sería el plano de cada apartamento de un piso:
  - área, largo, ancho
  - habitaciones
  - ventanas, etc.
- luego, construimos cada apartamento de cada piso (objetos) basándonos en ese plano

Clase

- **Atributos:**
  - **son variables que almacenan información sobre los objetos creados a partir de la clase**
  - **ejemplos:**
    - **nombre**
    - **color**
    - **tamaño**
    - **piso**
    - **habitaciones**

Clase

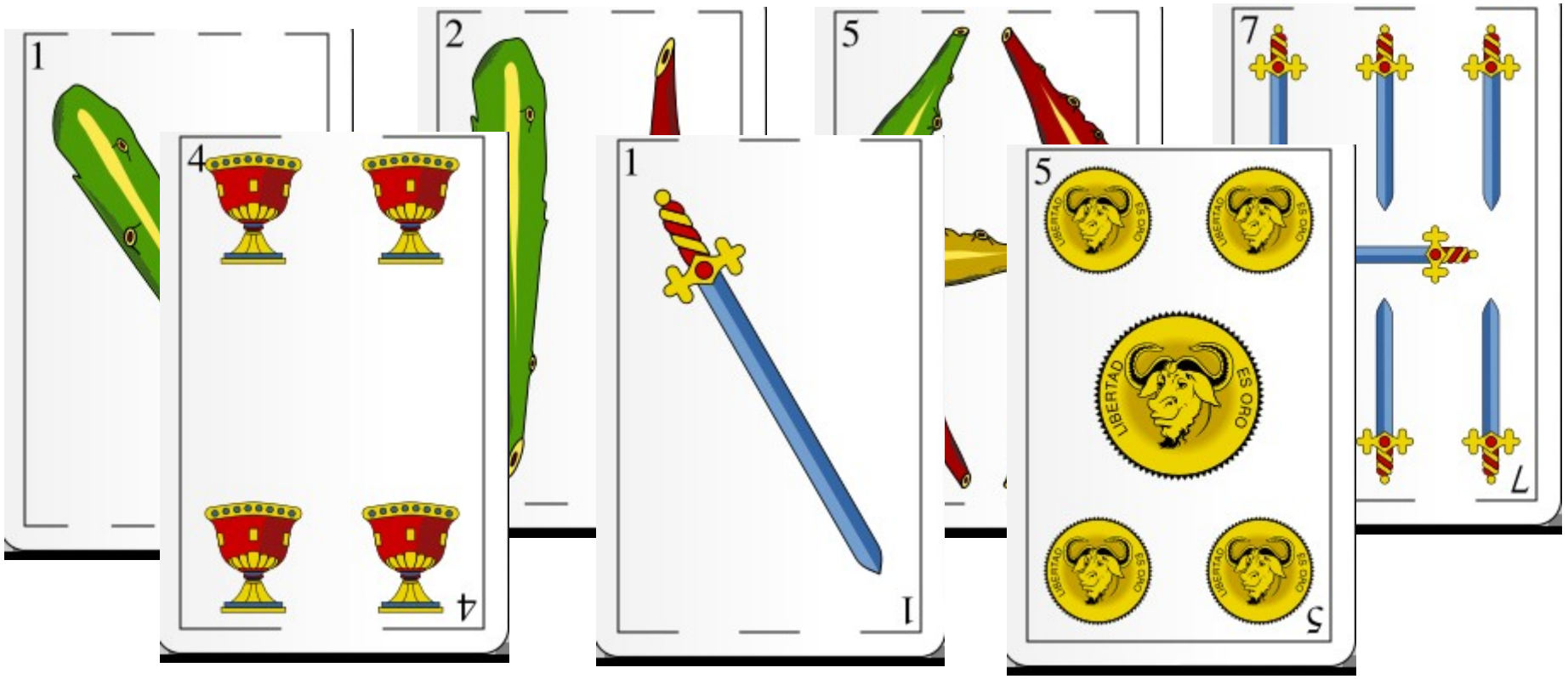
- **Métodos:**
  - **son funciones que definen el comportamiento de los objetos**
  - **que saben o pueden realizar**
  - **ejemplos:**
    - **arrancar**
    - **parar**
    - **calcular**
    - **mostrar**

Objeto

- **Instancia de una clase**
- **Cada objeto es único**
- **Difiere de los demás por su nombre y por los valores específicos de cada uno de los atributos que tiene la clase que se utilizó para crearlo.**
- **Posee el mismo comportamiento (métodos) que todos demás objetos de su clase**

Ejemplo Práctico

- **Aplicación que simule un juego de cartas…**



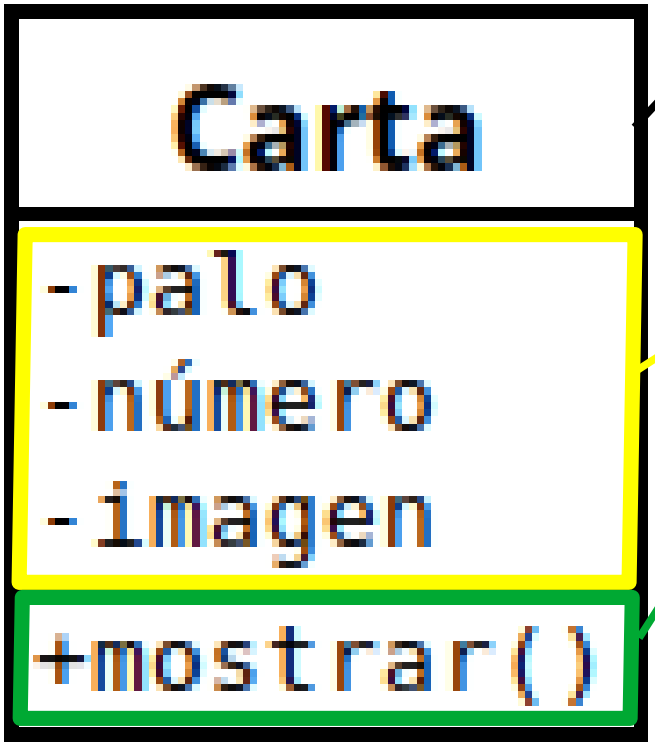
Ejemplo Práctico

- Deberíamos crear una clase que permita crear todas las cartas de este juego.
  - La clase: Carta
    - Comienza con mayúsculas y en singular
- Cada carta tiene los siguientes atributos:
  - palo (oro, copa, espada o basto)
  - número (1 al 12)
  - imagen

Ejemplo Práctico

- **Para diseñar aplicaciones orientadas a objetos se utiliza el Diagrama de Clases, entre otras herramientas.**
- **En este diagrama figuran todas las clases necesarias, sus atributos, sus métodos y los vínculos o relaciones que tienen entre sí.**

Diagrama de Clases



- **Nombre**
  - **En sigular**
  - **Comienza con mayúsculas**

- **Atributos (características)**

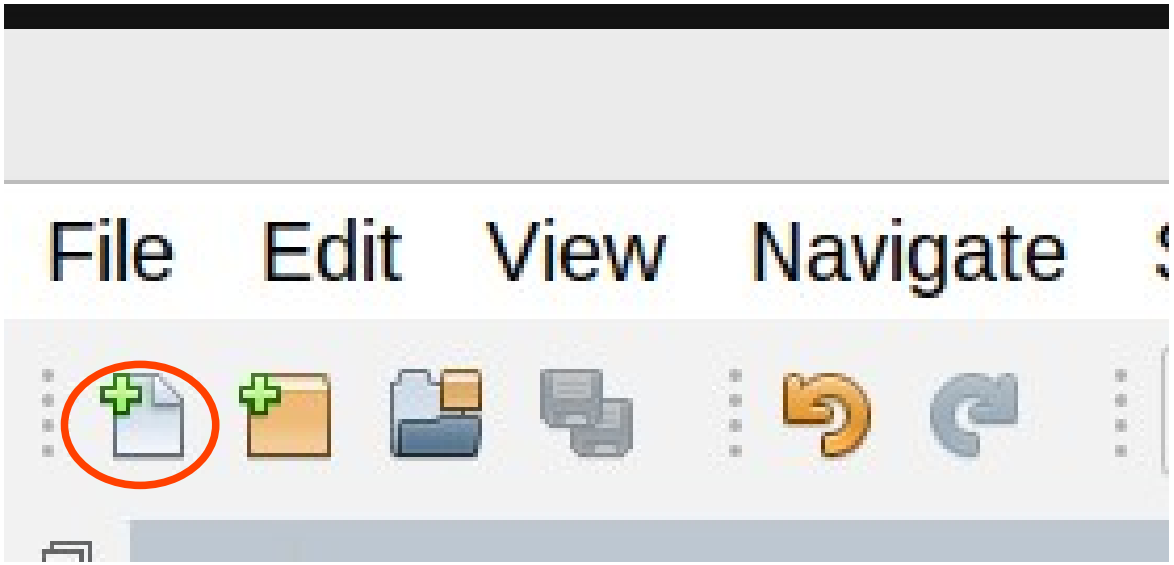
- **Métodos (comportamiento)**

Visibilidad

- Privado
+ Público
# Protegido

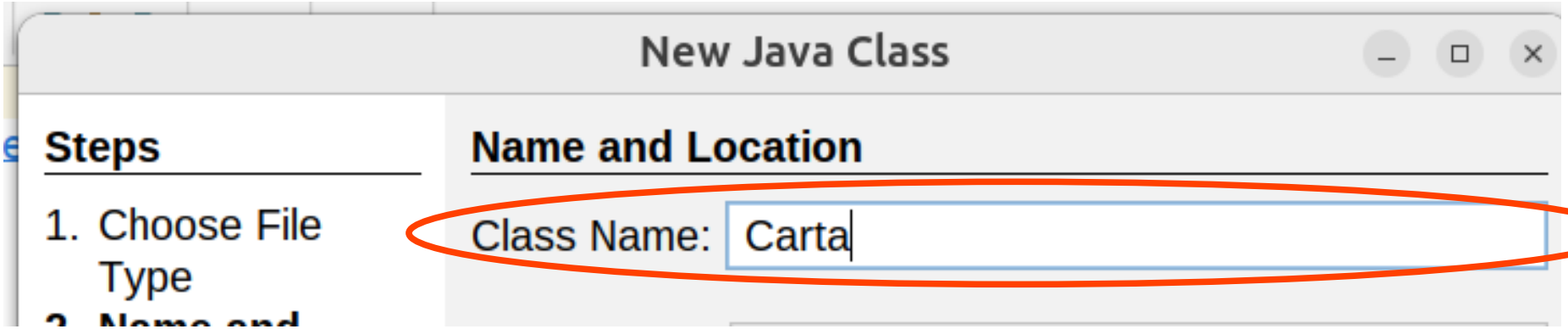
Ejemplo Práctico

- Luego de crear un nuevo proyecto en NetBeans
- Agregamos una nueva clase en el proyecto



Ejemplo Práctico

- **Indicamos su nombre**



Ejemplo Práctico

- **Generamos un archivo que contendrá la definición de la clase Carta**

```
public class Carta {

}
```

Ejemplo Práctico

- **Definimos cada atributo de la clase**
- **Los atributos son todos privados**

```
public class Carta {
    //Definicion de los atributos
    private int numero;
    private int palo;
    private String imagen;

}
```

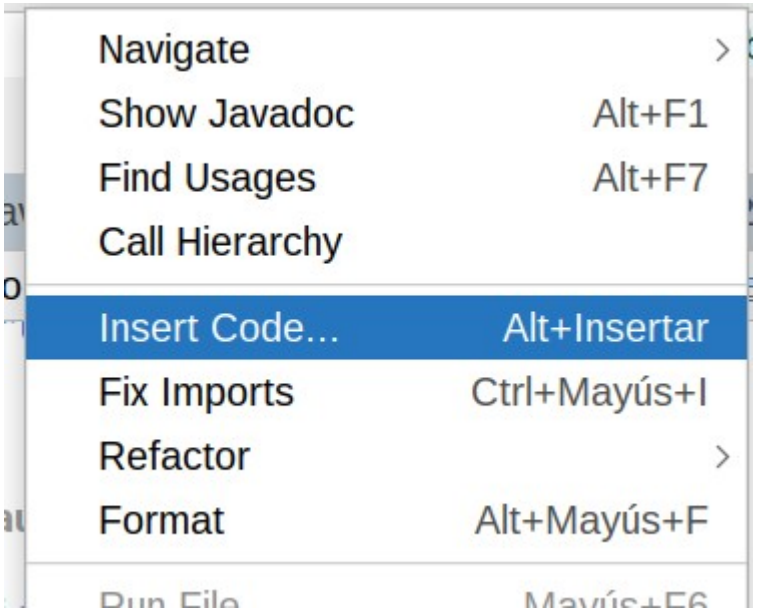
Ejemplo Práctico

- **Algunas constantes necesarias**

```
public class Carta {
    //Definicion de los atributos
    private int numero;
    private int palo;
    private String imagen;

    //Definir constantes
    private final int ORO = 0, COPA = 1, ESPADA = 2, BASTO = 3;
    private final String[] NOMBRES = {"c_o_", "c_c_", "c_e_", "c_b_"};
```

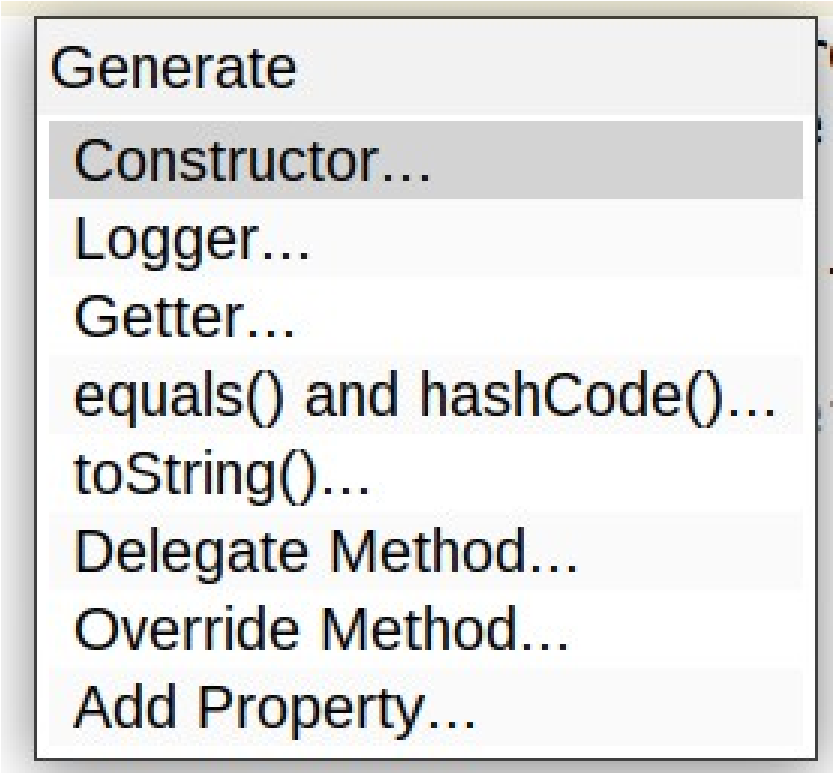
Ejemplo Práctico

- **Método Constructor**
  - **Encargado de inicializar los atributos de cada nuevo objeto**
  - **Se llama cada vez que se crea un objeto**
  - **Para generar su código debemos realizar un clic derecho y seleccionar la opción “Insert Code...”**



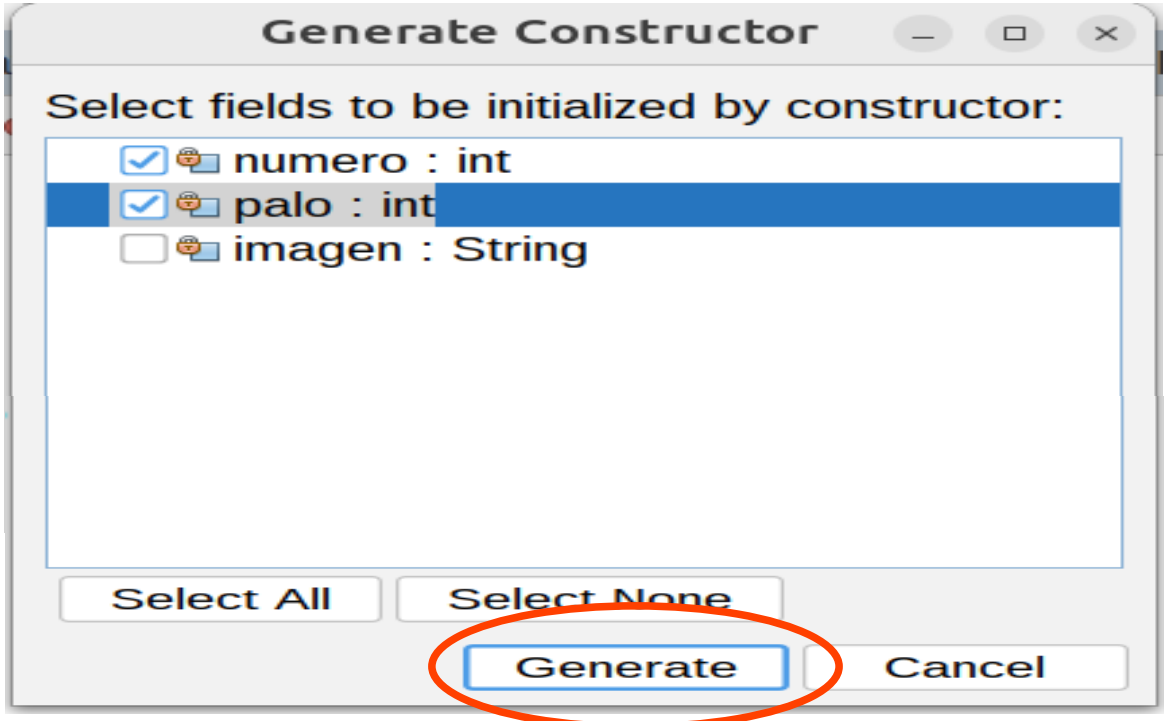
Ejemplo Práctico

- **Método Constructor**
  - **Luego seleccionar la opción “Constructor”**



Ejemplo Práctico

- **Método Constructor**
  - **Seleccionamos los atributos a inicializar**



Ejemplo Práctico

- **Método Constructor**
  - **Agregamos la siguiente inicialización para el atributo imagen**

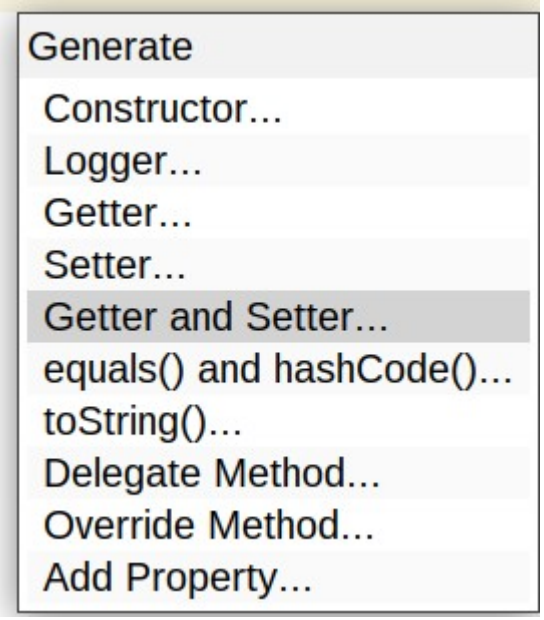
```
    //Metodo constructor

    public Carta(int numero, int palo) {
        this.numero = numero;
        this.palo = palo;
        this.imagen = "/" + NOMBRES[palo] + numero + ".png";
    }
```

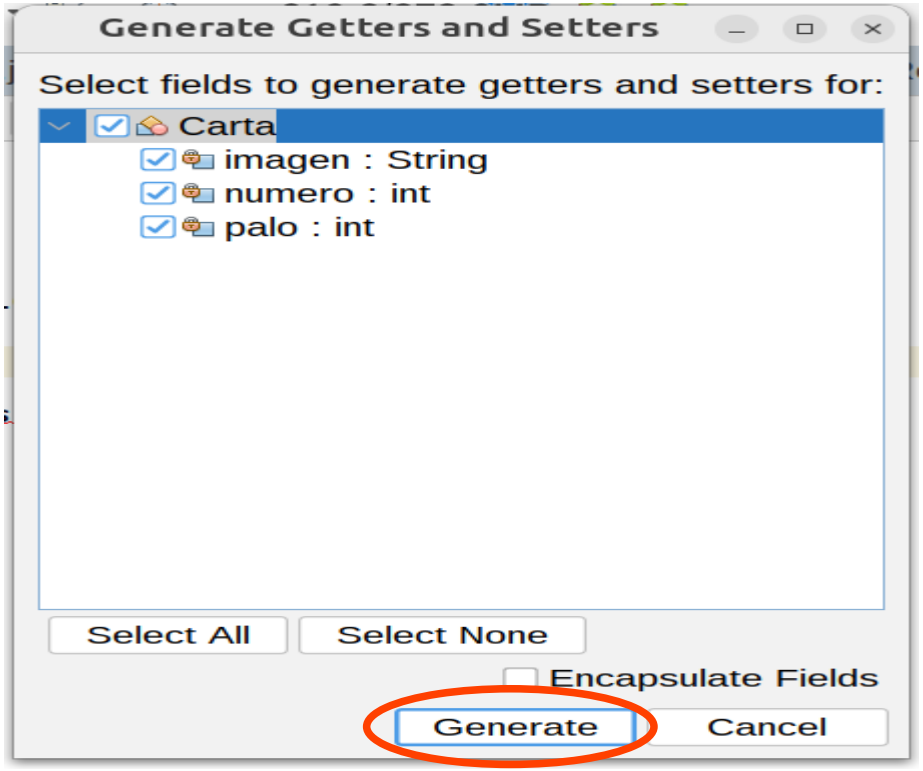
Ejemplo Práctico

- **Métodos Getters & Setters**
  - **Los atributos son todos privados**
  - **Para acceder o modificar su valor utilizamos métodos públicos**
  - **Getter: obtiene el valor de un atributo**
  - **Setter: modifica el valor de un atributo**
  - **Para generar su código debemos realizar un clic derecho y seleccionar la opción “Insert Code...”**

Ejemplo Práctico

- **Métodos Getters & Setters**





Ejemplo Práctico

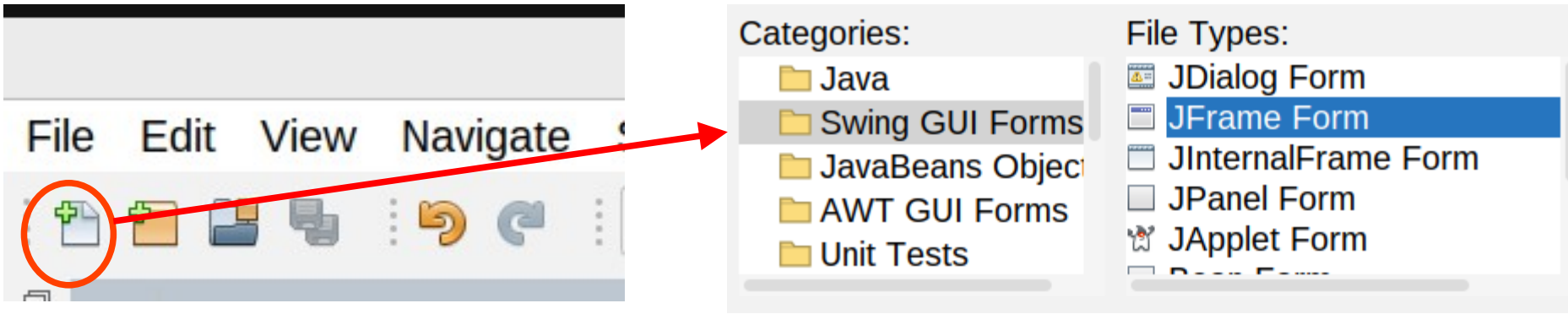
- **Falta agregar el código del método mostrar**

```
//Metodo mostrar una carta
public ImageIcon mostrar(){
    ImageIcon laImagen = new ImageIcon( getClass().getResource( imagen ) );
    return laImagen;
}
```

- **Hay que agregar el import para la clase ImageIcon**

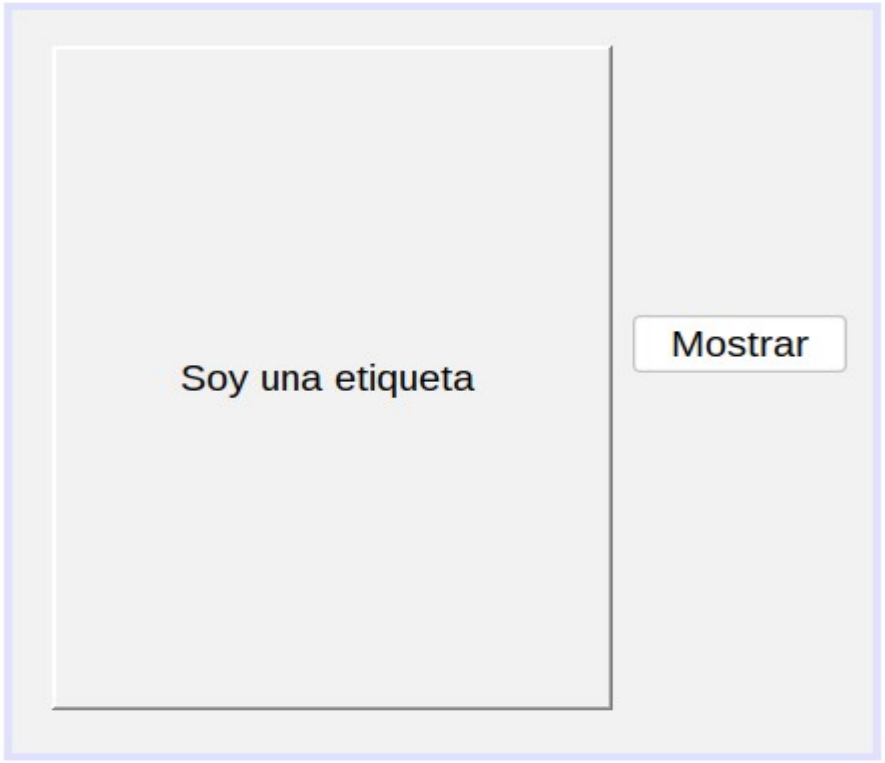
Ejemplo Práctico

- **Una vez definida la clase Carta debemos utilizarla en alguna ventana**
- **Para ello creamos un nuevo JFrame**



Ejemplo Práctico

- **Colocamos una etiqueta y un botón**



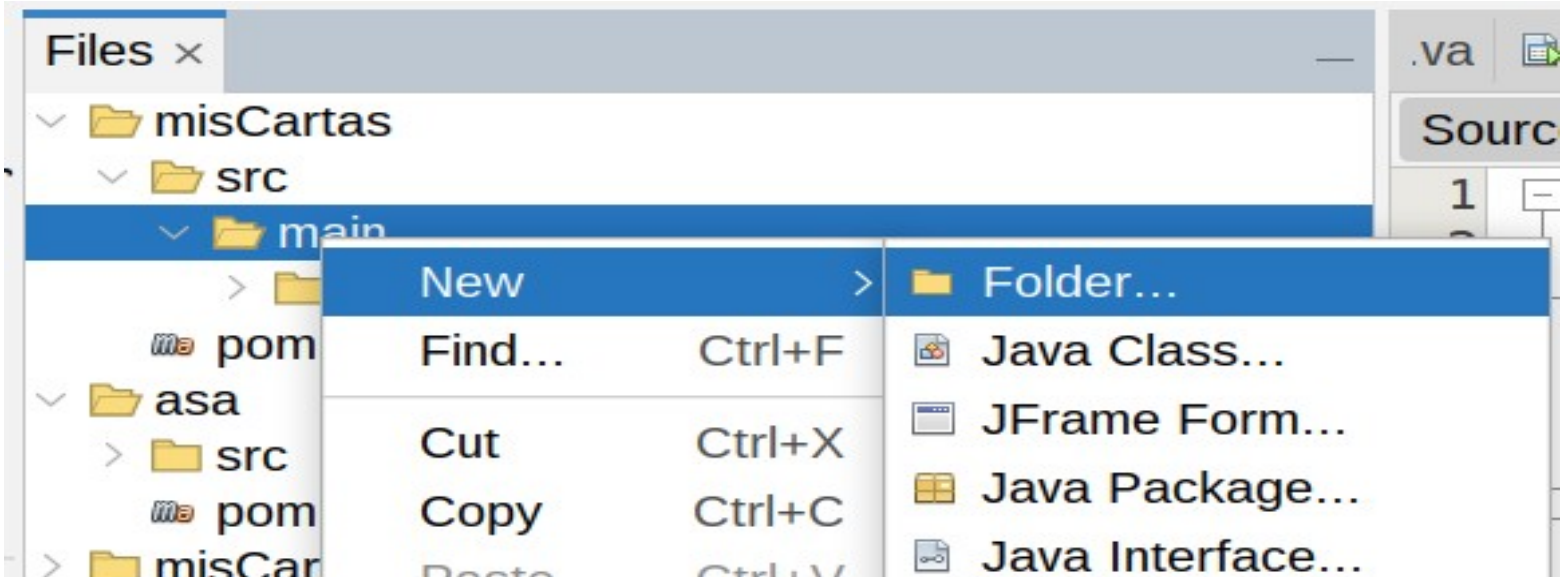
Ejemplo Práctico

- **Agregamos el siguiente código al botón**

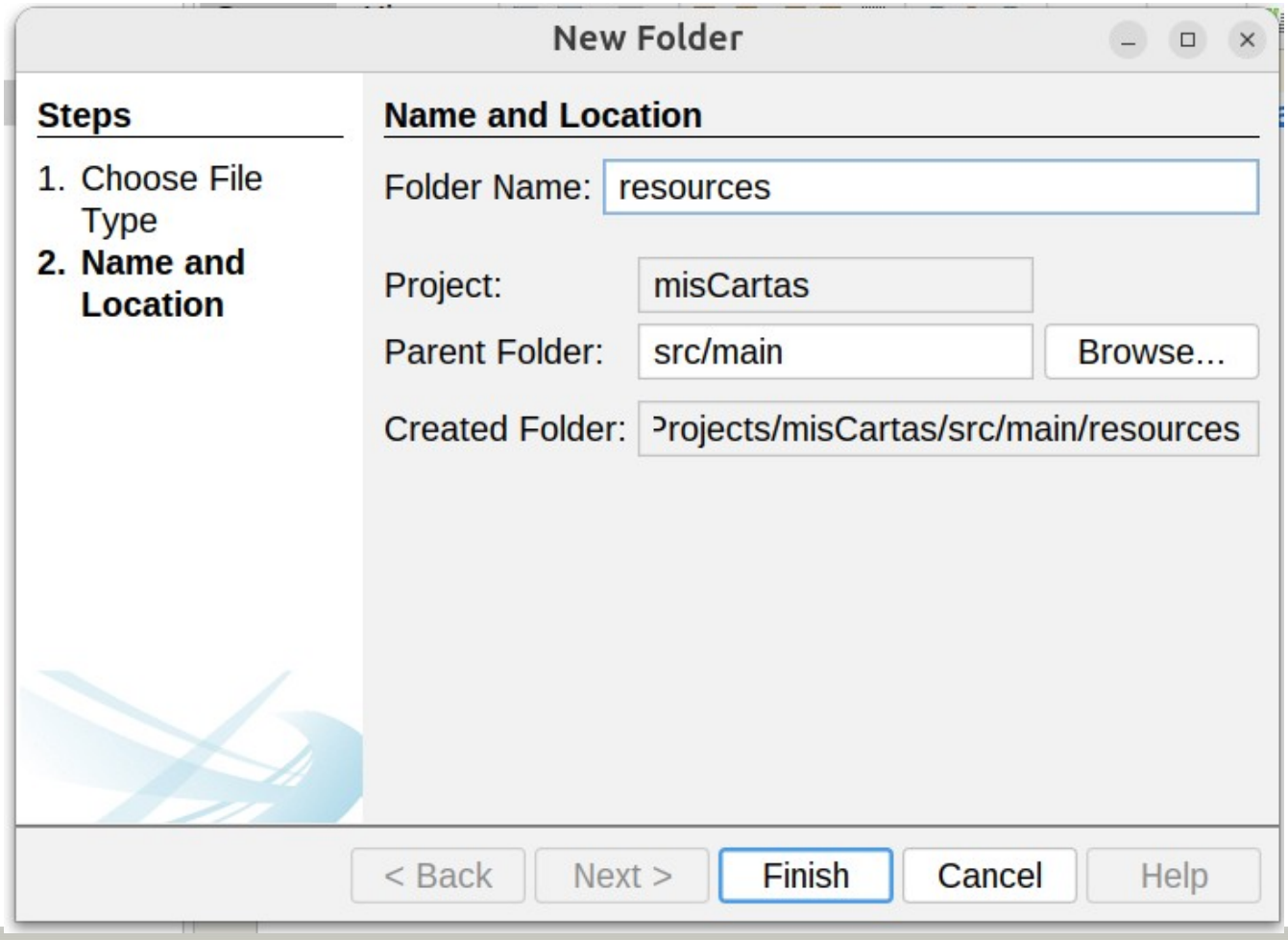
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //Crear una nueva - el 5 de oro (0)
     Carta unaCarta = new Carta(0, 5);

    //Mostrar la carta en la etiqueta
    jLabel1.setIcon(unaCarta.mostrar());

}
```

Ejemplo Práctico

- Agregamos las imágenes de las cartas
- Debemos crear la carpeta resources



Ejemplo Práctico



Ejemplo Práctico

- **Debemos arrastrar todas las imágenes a la carpeta anterior**