



**POLITÉCNICO  
ESTELLA**

**ASIGNATURA: ETORNOS DE DESARROLLO**

**GRADO SUPERIOR EN DESARROLLO DE  
APLICACIONES MULTIPLATAFORMA**

**TAREA 09 TEMA 05**

Presentado por: Eugen Moga

## ÍNDICE

Ejercicios 1 .....	1
Ejercicios 2 .....	2
Ejercicios 3 .....	3
Ejercicios 4 .....	6
Ejercicios 5 .....	8

# Ejercicios 1

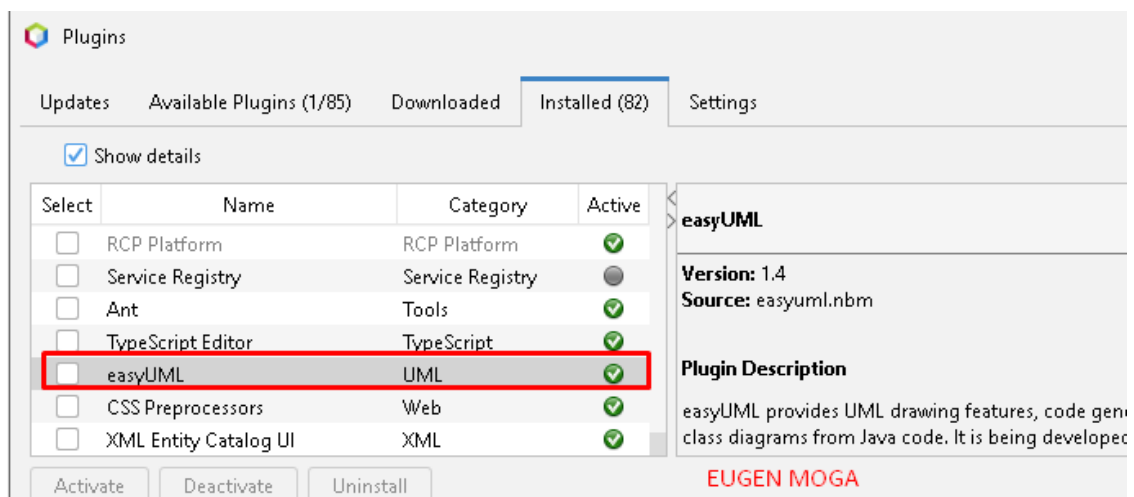
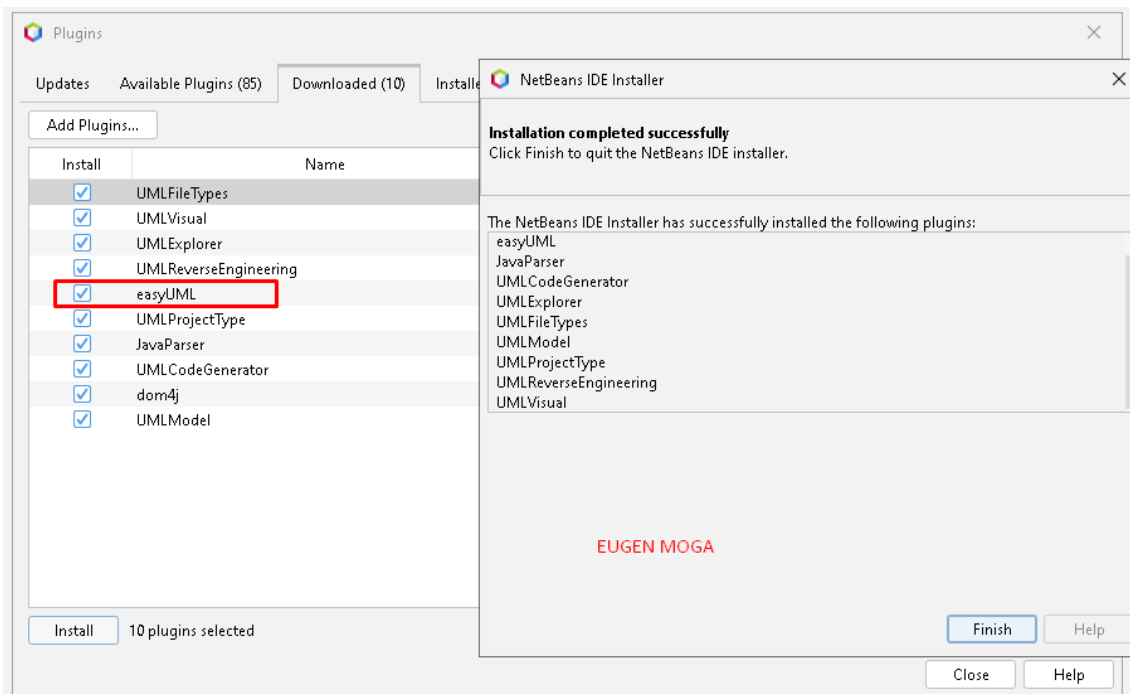
Instalar el plugin de easyUML en tu IDE Netbeans

<https://github.com/mgeeee35/easyUML>

[https://www.youtube.com/watch?v=jigmoX\\_Ku8o](https://www.youtube.com/watch?v=jigmoX_Ku8o)

Para instalar easyUML en NetBeans acedo al link de github descargo easyUML con todos los paquetes y después voy a NetBeans >> Tools >> Plugins >> Downloaded >> Add Plugins y selecciono los paquetes descargados. Le doy a Install.

Pd: me saltaba un error de validación porque no encontraba el archivo unpack200.exe y lo busqué por internet y así pude finalizar la instalación



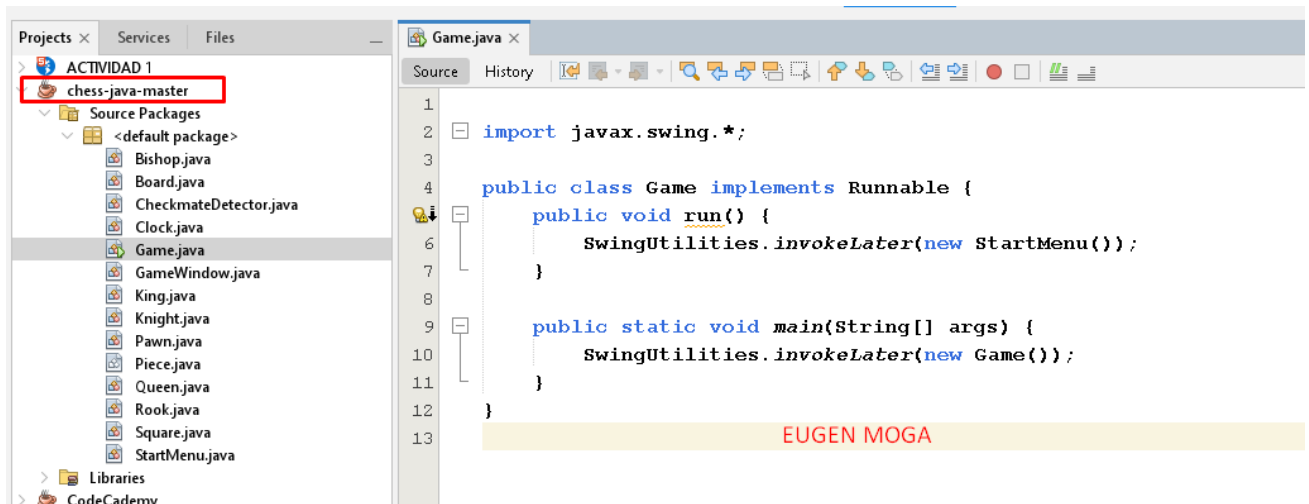
## Ejercicios 2

Descarga el proyecto siguiente y ábrelo en NetBeans (no hace falta ejecutarlo):  
<https://github.com/jlundstedt/chess-java>

Para descargar el proyecto accedo al link mencionado y descargo el proyecto en un archivo .zip

Una vez descargado en NetBeans voy a File >> Import Project >> From ZIP y selecciono el archivo.zip descargado desde github. Y le doy a Import.

PD: Se cargo el archivo a la carpeta donde tengo todos los proyectos, pero netbeans no me muestra el Proyecto descargado e importado. Asi que creo manualmente el proyecto y copio todos los archivos fuente del proyecto descargado de github.

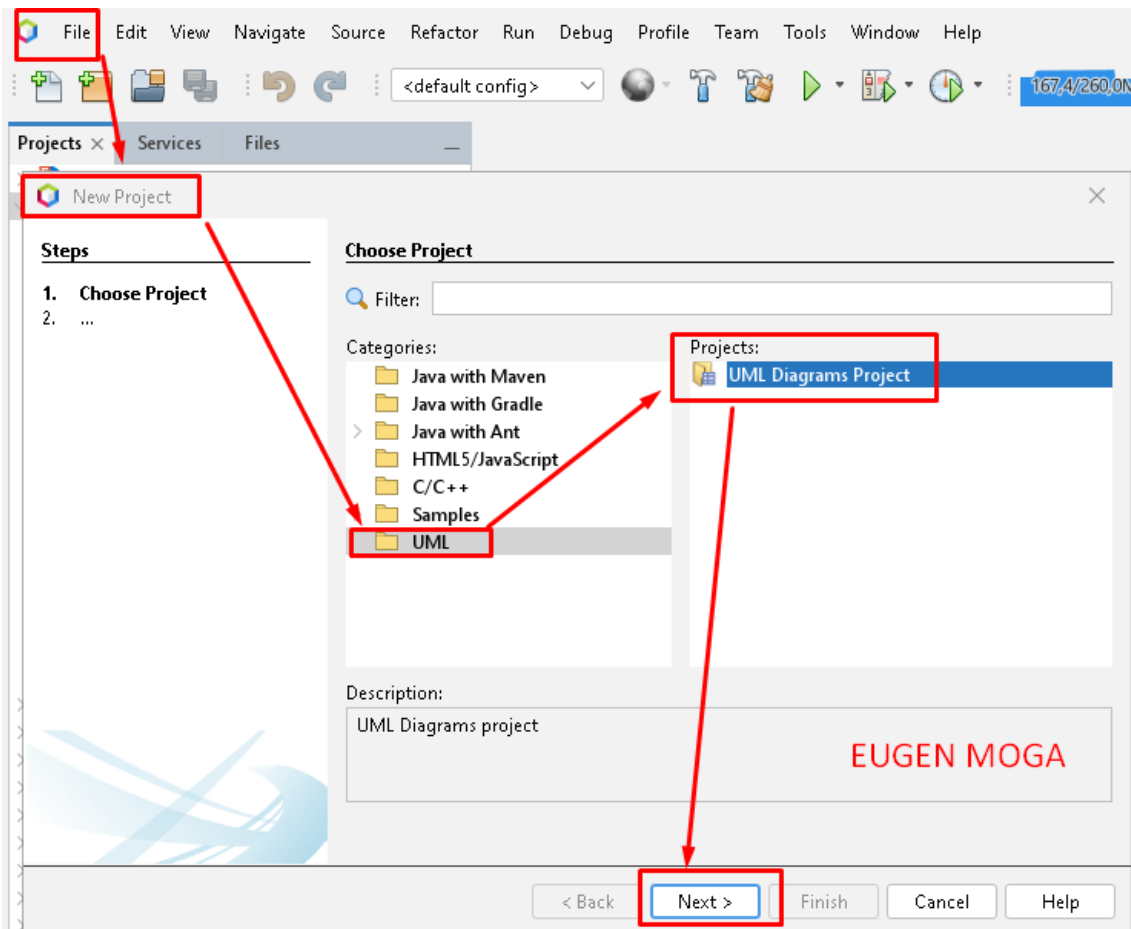


### Ejercicios 3

Genera el diagrama de clases UML con easyUML desde el icono de proyecto o seleccionando todas las clases.

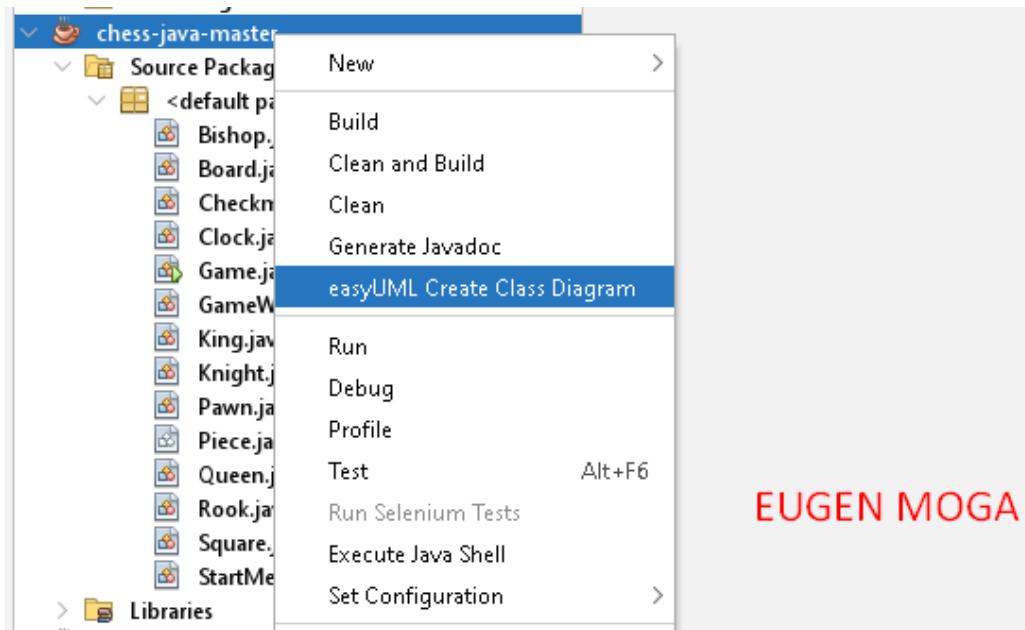
- Ordena el diagrama de manera a ver mejor las relaciones entre clases, en particular entre la clase Piece y sus subclases (para más claridad puedes reducir las ventanas de cada clase).
- Exporta el diagrama a un fichero imagen y adjúntalo en la tarea.

Para Generar el diagrama de clases primero genero un nuevo proyecto UML para ello voy a File >> New Project >> UML >> UML Diagrams Project

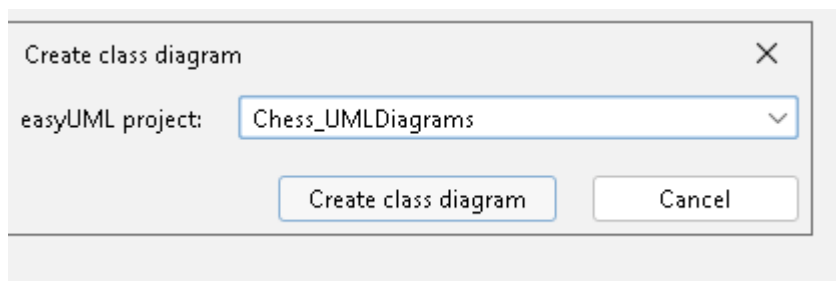


Le pongo de nombre Chess\_UMLDiagrams y le doy a Finish. Una vez creado el proyecto UML le doy clic derecho en el proyecto chess-java-master que es el proyecto donde están todos los archivos fuentes descargados de github.

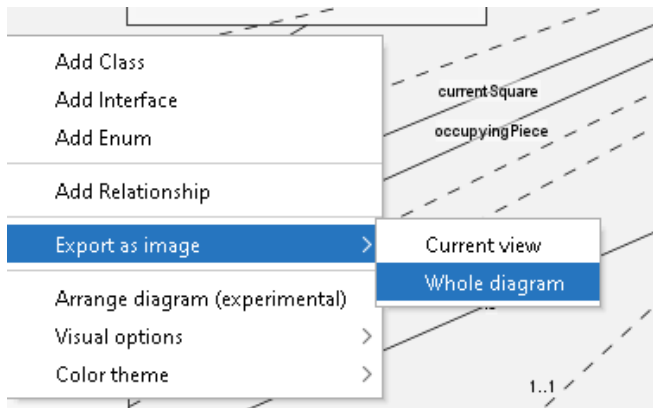
Clic derecho >> easyUML Create Class Diagram



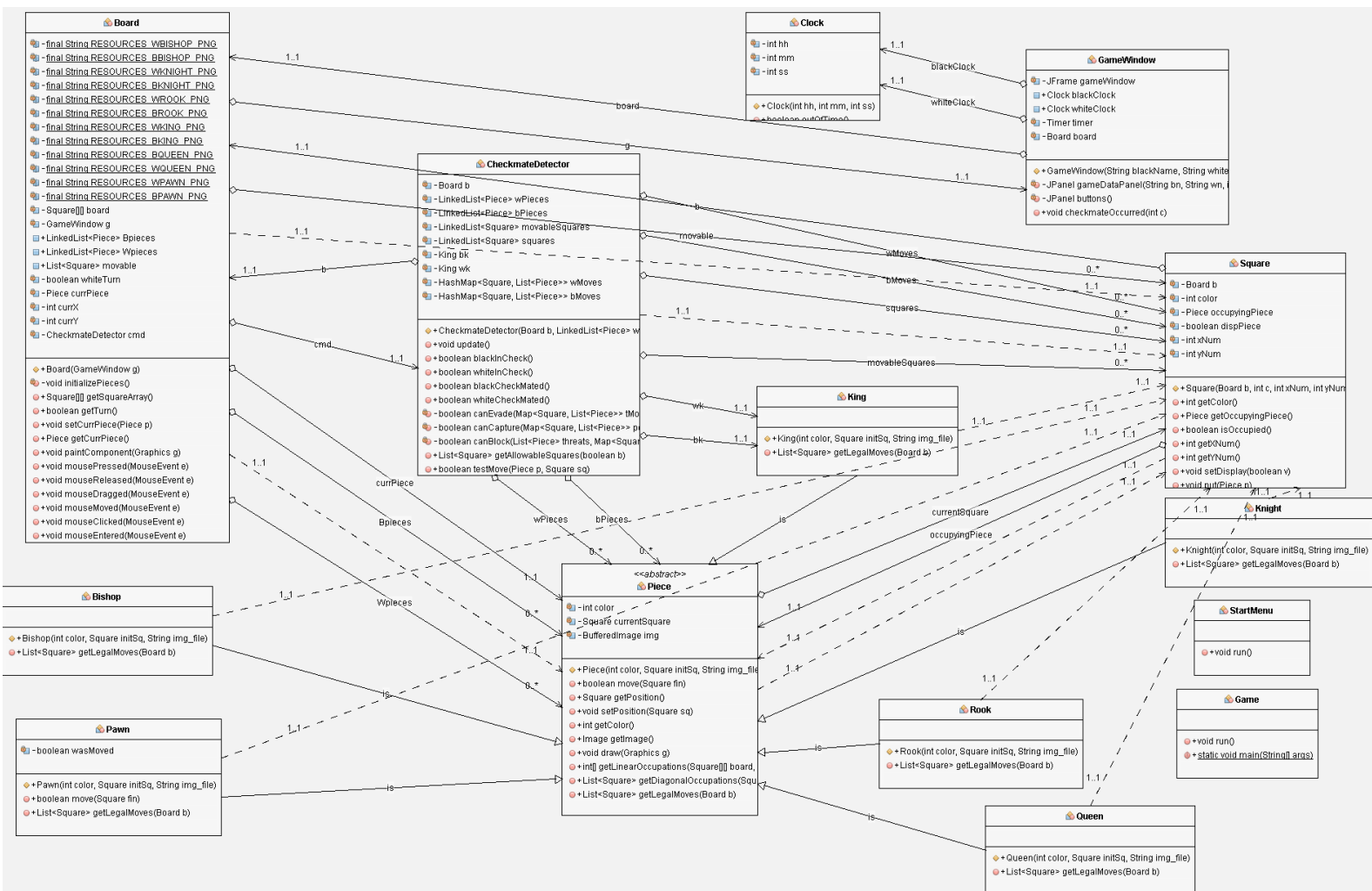
Selecciono el Proyecto UML donde quiero generar el diagrama, en mi caso Chess\_UMLDiagrams y le doy a Create class diagram



Ordeno las clases para que se pueda ver medianamente bien, aunque es imposible y una vez que lo tengo le pulso clic derecho Export as image >> Whole diagram



Este es el resultado



Pd: para que se vea algo decente recomiendo ampliar a 200%

## Ejercicios 4


Analiza la clase “Piece” desde su representación en el diagrama producido:

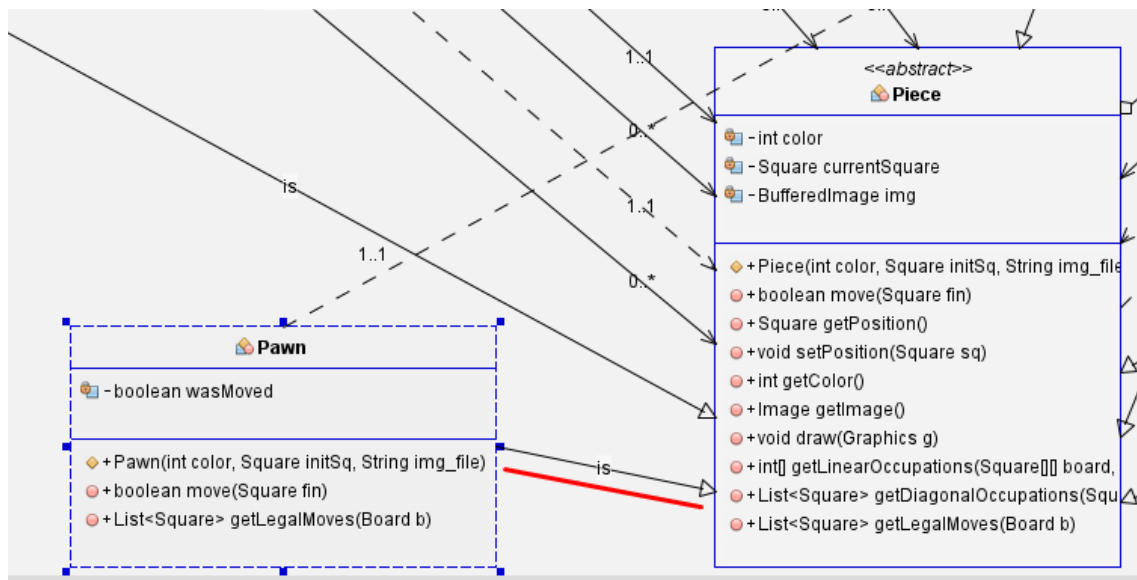
- ¿De qué tipo de clase se trata? ¿Cuál es su significado? ¿Qué clases heredan de ella?

“Piece” Se trata de una clase abstracta,



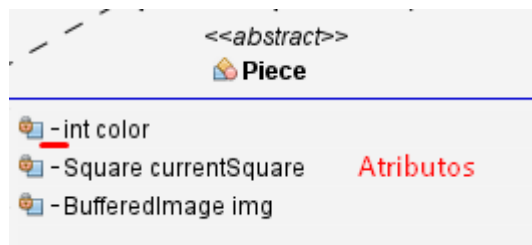
Significa que no se puede instanciar directamente o dicho de otra forma no se puede crear un objeto de tipo “Piece” solo sirve como plantilla para las clases que heredan de esta clase.

Las clases que heredan de “Piece” son: Pawn, Bishop, Queen, Rook, King y Knight. La herencia se representa con esta flecha  ejemplo:



- ¿Cuáles son sus atributos? ¿Qué visibilidad tienen dentro del paquete?

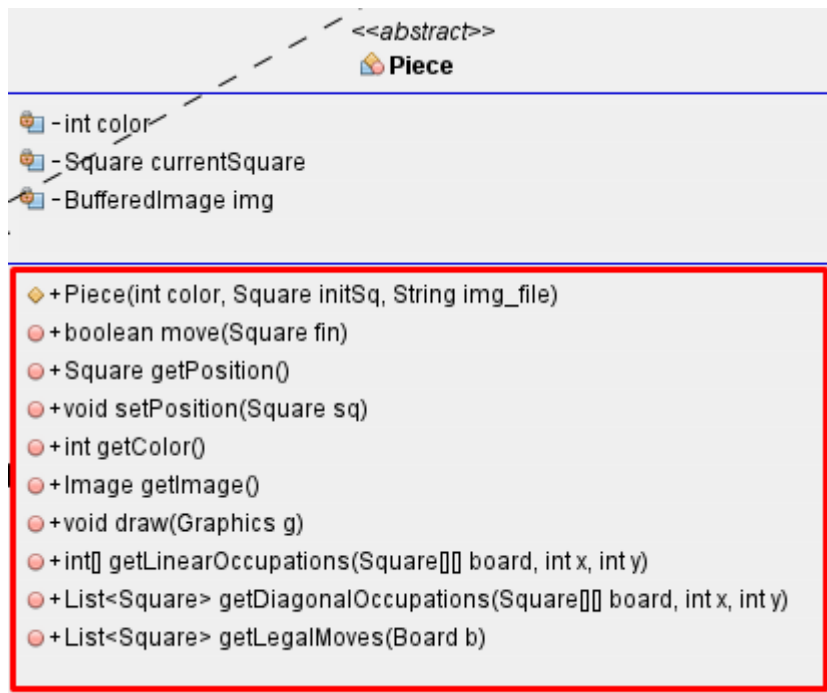
Sus atributos son: color, currentSquare e img y su visibilidad es privada por el signo negativo que tiene antes de su declaración.





- Para cada método de esa clase, determina la visibilidad, los parámetros y sus tipos, el tipo de variable devuelto.

Método	Visibilidad	Parámetros y tipos	Tipo variable devuelto
Piece (constructor)	+ publico	int color Square initSq String img_file	No devuelve nada
move	+ publico	Square fin	Booleano
getPosition	+ publico	Nada	Square
setPosition	+ publico	Square sq	Void
getColor	+ publico	Nada	Entero
getImage	+ publico	Nada	Imagen
draw	+ publico	Graphics g	Void
getLinearOccupations	+ publico	Square board int x int y	Entero Array
getDiagonalOccupations	+ publico	Square board int x int y	List Square
getLegalMoves	+ publico	Board b	List Square



## Ejercicios 5

Localiza en el código de la clase “Piece” los elementos anteriormente descritos.

Los métodos descritos anteriormente son:

Contructor

```
public Piece(int color, Square initSq, String img file) {  
    this.color = color;  
    this.currentSquare = initSq;  
  
    try {  
        if (this.img == null) {  
            this.img = ImageIO.read(getClass().getResource(img_file));  
        }  
    } catch (IOException e) {  
        System.out.println("File not found: " + e.getMessage());  
    }  
}
```

Método move:

```
public boolean move(Square fin) {  
    Piece occup = fin.getOccupyingPiece();  
  
    if (occup != null) {  
        if (occup.getColor() == this.color) return false;  
        else fin.capture(this);  
    }  
  
    currentSquare.removePiece();  
    this.currentSquare = fin;  
    currentSquare.put(this);  
    return true;  
}
```

Método getPosition:

```
public Square getPosition() {  
    return currentSquare;  
}
```

Método setPosition:

```
public void setPosition(Square sq) {  
    this.currentSquare = sq;  
}
```

Método getColor:

```
public int getColor() {  
    return color;  
}
```

Método getImage:

```
public Image getImage() {  
    return img;  
}
```

Método draw:

```
public void draw(Graphics g) {  
    int x = currentSquare.getX();  
    int y = currentSquare.getY();  
  
    g.drawImage(this.img, x, y, null);  
}
```

Método getLinearOccupations:

```
public int[] getLinearOccupations(Square[][] board, int x, int y) {  
    int lastYabove = 0;  
    int lastXright = 7;  
    int lastYbelow = 7;  
    int lastXleft = 0;
```

Método getDiagonalOccupations:

```
public List<Square> getDiagonalOccupations(Square[][] board, int x, int y) {  
    LinkedList<Square> diagOccup = new LinkedList<Square>();  
  
    int xNW = x - 1;  
    int xSW = x - 1;  
    int xNE = x + 1;  
    int xSE = x + 1;  
    int yNW = y - 1;  
    int ySW = y + 1;  
    int yNE = y - 1;  
    int ySE = y + 1;
```

Método getLegalMoves:

```
// No implementation, to be implemented by each subclass  
public abstract List<Square> getLegalMoves(Board b);
```

¿Cómo se notifica en las subclases la herencia a la clase “piece”?

En las subclases se notifica que heredan de “Piece” cuando se define la declaración de la clase después del nombre de la propia clase se pone extends Piece

```
public class Pawn extends Piece {  
    private boolean wasMoved;
```