

## Trabajo práctico N° 3

# “Introducción a la Programación Orientada a Objetos”

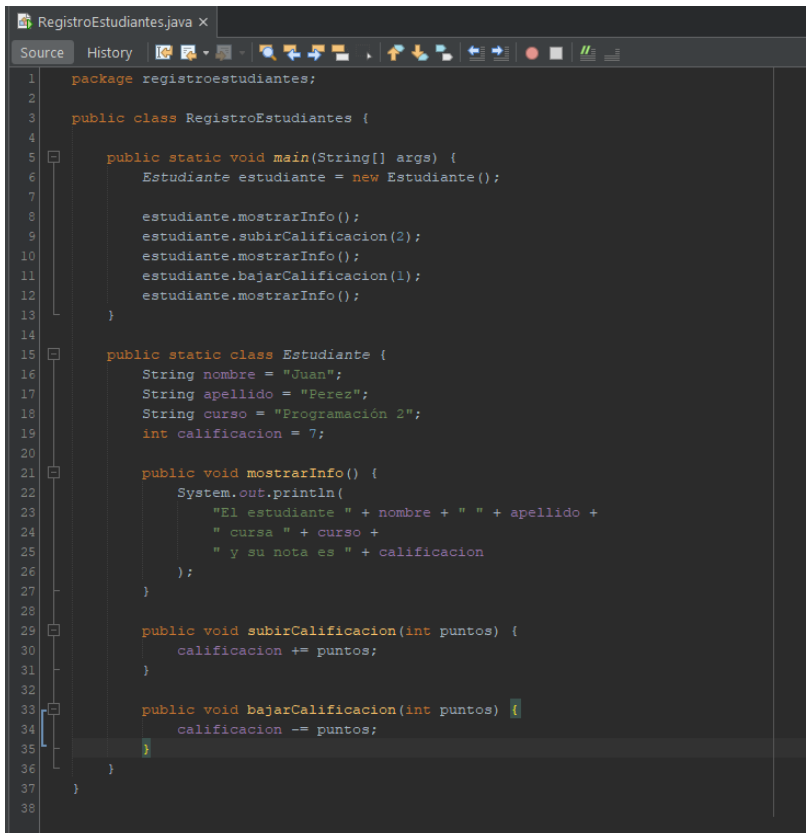
Alumnos: Alemis, Juan Cruz – [juancruzalemis@gmail.com](mailto:juancruzalemis@gmail.com)

Materia: Programacion II

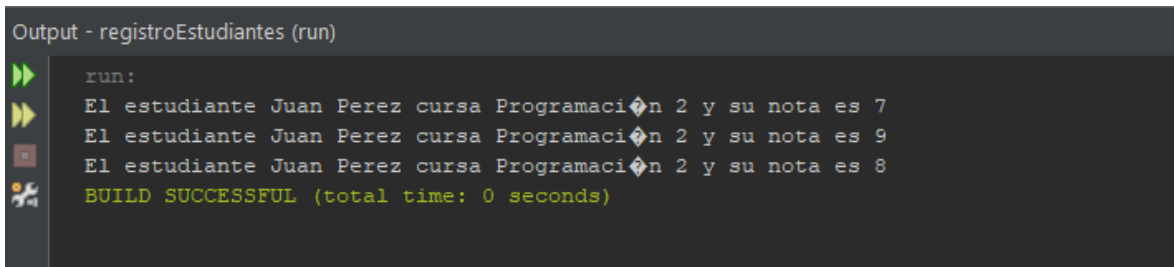
## 1. Registro de Estudiantes

a. Crear una clase Estudiante con los atributos: nombre, apellido, curso, calificación. Métodos requeridos: mostrarInfo(), subirCalificacion(puntos), bajarCalificacion(puntos).

Tarea: Instanciar a un estudiante, mostrar su información, aumentar y disminuir calificaciones.



```
1 package registroestudiantes;
2
3 public class RegistroEstudiantes {
4
5     public static void main(String[] args) {
6         Estudiante estudiante = new Estudiante();
7
8         estudiante.mostrarInfo();
9         estudiante.subirCalificacion(2);
10        estudiante.mostrarInfo();
11        estudiante.bajarCalificacion(1);
12        estudiante.mostrarInfo();
13    }
14
15    public static class Estudiante {
16        String nombre = "Juan";
17        String apellido = "Perez";
18        String curso = "Programación 2";
19        int calificacion = 7;
20
21        public void mostrarInfo() {
22            System.out.println(
23                "El estudiante " + nombre + " " + apellido +
24                " cursa " + curso +
25                " y su nota es " + calificacion
26            );
27        }
28
29        public void subirCalificacion(int puntos) {
30            calificacion += puntos;
31        }
32
33        public void bajarCalificacion(int puntos) {
34            calificacion -= puntos;
35        }
36    }
37 }
38
```



```
Output - registroEstudiantes (run)
run:
El estudiante Juan Perez cursa Programación 2 y su nota es 7
El estudiante Juan Perez cursa Programación 2 y su nota es 9
El estudiante Juan Perez cursa Programación 2 y su nota es 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

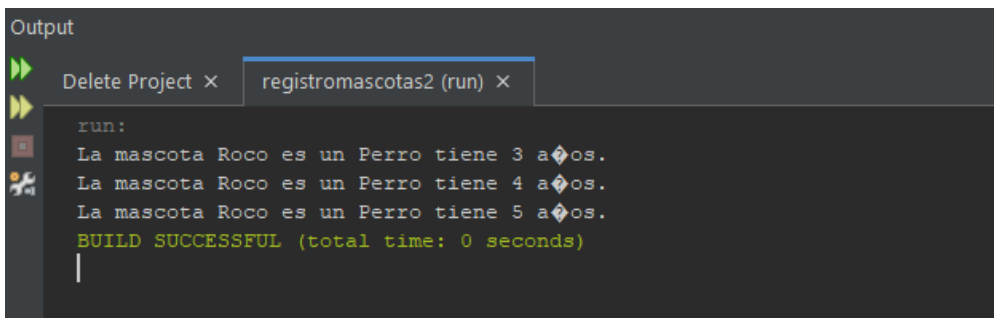
## 2. Registro de Mascotas

a. Crear una clase Mascota con los atributos: nombre, especie, edad. Métodos requeridos: mostrarInfo(), cumplirAños().

Tarea: Crear una mascota, mostrar su información, simular el paso del tiempo y verificar los cambios.



```
1 package registromascotas2;
2
3
4
5 public class Registromascotas2 {
6
7
8     public static void main(String[] args) {
9         Mascota mascota = new Mascota();
10        mascota.mostrarInfo();
11        mascota.cumplirAños();
12        mascota.mostrarInfo();
13        mascota.cumplirAños();
14        mascota.mostrarInfo();
15    }
16
17    public static class Mascota {
18        String nombre = "Roco";
19        String especie = "Perro";
20        int edad = 3;
21
22        public void mostrarInfo() {
23            System.out.println(
24                "La mascota " + nombre +
25                " es un " + especie +
26                " tiene " + edad + " años."
27            );
28        }
29
30        public void cumplirAños() {
31            edad++;
32        }
33    }
34 }
```



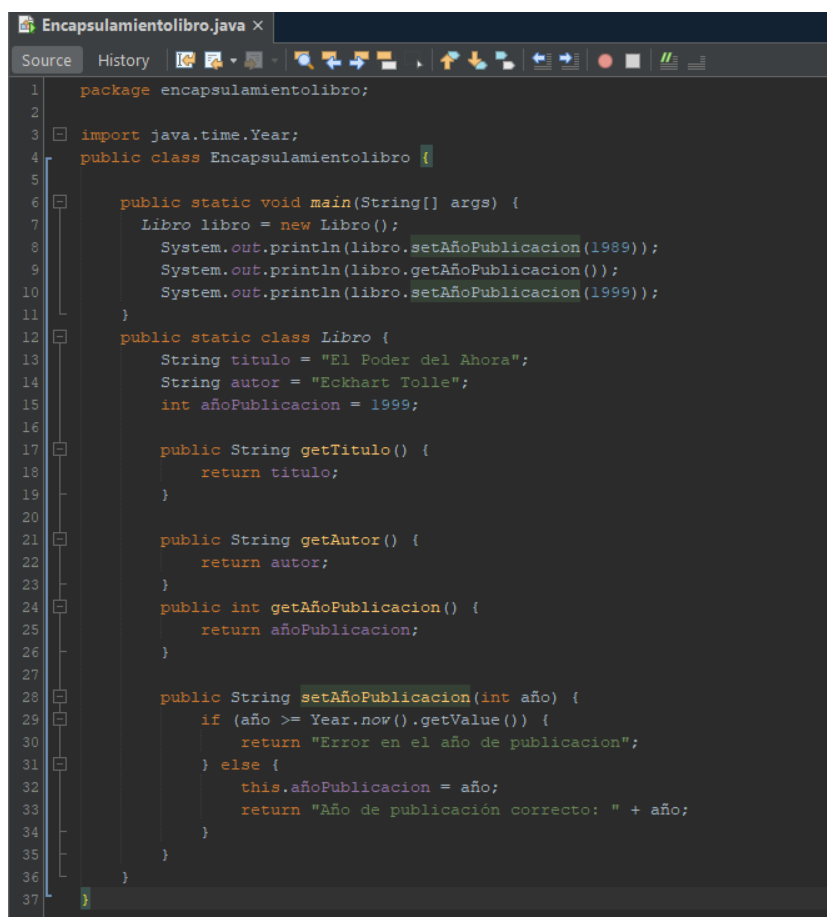
```
Output
Delete Project x registromascotas2 (run) x
run:
La mascota Roco es un Perro tiene 3 años.
La mascota Roco es un Perro tiene 4 años.
La mascota Roco es un Perro tiene 5 años.
BUILD SUCCESSFUL (total time: 0 seconds)
```

### 3. Encapsulamiento con la Clase Libro

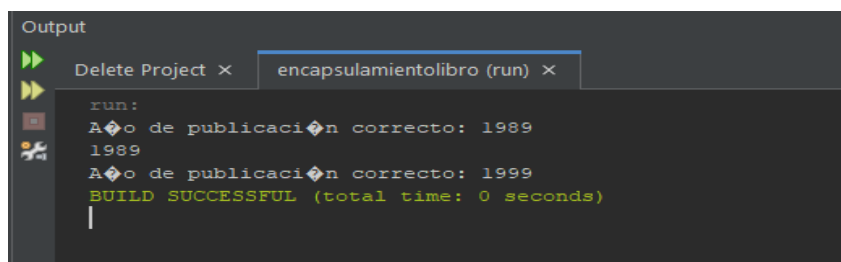
- a. Crear una clase Libro con atributos privados: titulo, autor, añoPublicacion.

Métodos requeridos: Getters para todos los atributos. Setter con validación para añoPublicacion.

Tarea: Crear un libro, intentar modificar el año con un valor inválido y luego con uno válido, mostrar la información final.



```
1 package encapsulamientolibro;
2
3 import java.time.Year;
4 public class EncapsulamientoLibro {
5
6     public static void main(String[] args) {
7         Libro libro = new Libro();
8         System.out.println(libro.setAñoPublicacion(1989));
9         System.out.println(libro.getAñoPublicacion());
10        System.out.println(libro.setAñoPublicacion(1999));
11    }
12
13    public static class Libro {
14        String titulo = "El Poder del Ahora";
15        String autor = "Eckhart Tolle";
16        int añoPublicacion = 1999;
17
18        public String getTitulo() {
19            return titulo;
20        }
21
22        public String getAutor() {
23            return autor;
24        }
25
26        public int getAñoPublicacion() {
27            return añoPublicacion;
28        }
29
30        public String setAñoPublicacion(int año) {
31            if (año >= Year.now().getValue()) {
32                return "Error en el año de publicacion";
33            } else {
34                this.añoPublicacion = año;
35                return "Año de publicación correcto: " + año;
36            }
37        }
38    }
39 }
```



```
Output
Delete Project x encapsulamientolibro (run) x
run:
Año de publicación correcto: 1989
1989
Año de publicación correcto: 1999
BUILD SUCCESSFUL (total time: 0 seconds)
```

#### 4. Gestión de Gallinas en Granja Digital

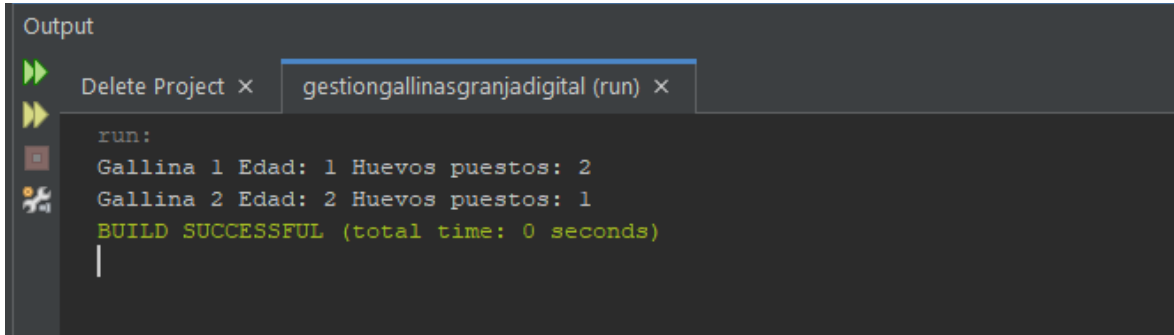
a. Crear una clase Gallina con los atributos: idGallina, edad, huevosPuestos.

Métodos requeridos: ponerHuevo(), envejecer(), mostrarEstado().

Tarea: Crear dos gallinas, simular sus acciones (envejecer y poner huevos), y mostrar su estado.

```
GestionGallinasGranjaDigital.java x
Source History
1 public class GestionGallinasGranjaDigital {
2
3     public static void main(String[] args) {
4         Gallina g1 = new Gallina();
5         Gallina g2 = new Gallina();
6         g1.idGallina = 1;
7         g1.edad = 0;
8         g1.huevosPuestos = 0;
9
10        g2.idGallina = 2;
11        g2.edad = 0;
12        g2.huevosPuestos = 0;
13
14        g1.ponerHuevo();
15        g1.ponerHuevo();
16        g1.envejecer();
17
18        g2.ponerHuevo();
19        g2.envejecer();
20        g2.envejecer();
21
22        g1.mostrarEstado();
23        g2.mostrarEstado();
24    }
25 }
```

```
25
26 public static class Gallina {
27     int idGallina;
28     int edad;
29     int huevosPuestos;
30
31     public void ponerHuevo() {
32         huevosPuestos++;
33     }
34
35     public void envejecer() {
36         edad++;
37     }
38
39     public void mostrarEstado() {
40         System.out.println("Gallina " + idGallina +
41                             " Edad: " + edad +
42                             " Huevos puestos: " + huevosPuestos);
43     }
44 }
45 }
```



```
Output
Delete Project x gestiongallinasgranjadigital (run) x
run:
Gallina 1 Edad: 1 Huevos puestos: 2
Gallina 2 Edad: 2 Huevos puestos: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 5. Simulación de Nave Espacial

Crear una clase NaveEspacial con los atributos: nombre, combustible.

Métodos requeridos: despegar(), avanzar(distancia), recargarCombustible(cantidad), mostrarEstado().

Reglas: Validar que haya suficiente combustible antes de avanzar y evitar que se supere el límite al recargar.

Tarea: Crear una nave con 50 unidades de combustible, intentar avanzar sin recargar, luego recargar y avanzar correctamente. Mostrar el estado al final.

```

1 package simulacionnaveespacial;
2
3 public class Simulacionnaveespacial {
4
5
6     public static void main(String[] args) {
7         NaveEspacial nave = new NaveEspacial();
8         nave.nombre = "LA_SABEIRO";
9         nave.combustible = 50;
10
11         nave.avanzar(75);
12
13         nave.recargarCombustible(25);
14
15         nave.avanzar(75);
16
17         nave.despegar();
18         nave.mostrarEstado();
19     }
20
21     public static class NaveEspacial {
22         String nombre;
23         int combustible;
24         final int MAX_COMBUSTIBLE = 100;
25
26         public void despegar() {
27             System.out.println(nombre + " está despegando");
28         }
29
30         public void avanzar(int distancia) {
31             if (combustible >= distancia) {
32                 combustible -= distancia;
33                 System.out.println(nombre + " avanzó" + distancia + " unidades.");
34             } else {
35                 System.out.println("No hay suficiente combustible para avanzar " + distancia + " unidades.");
36             }
37         }
38     }

```

```

39     public void recargarCombustible(int cantidad) {
40         if (combustible + cantidad > MAX_COMBUSTIBLE) {
41             combustible = MAX_COMBUSTIBLE;
42             System.out.println("Combustible recargado al máximo: " + MAX_COMBUSTIBLE);
43         } else {
44             combustible += cantidad;
45             System.out.println("Combustible recargado: " + cantidad + " unidades.");
46         }
47     }
48
49     public void mostrarEstado() {
50         System.out.println("Estado de " + nombre + " Combustible: " + combustible);
51     }
52 }
53

```

```

Output
Delete Project × simulacionnaveespacial (run) ×
run:
No hay suficiente combustible para avanzar 75 unidades.
Combustible recargado: 25 unidades.
LA_SABEIRO avanzó 75 unidades.
LA_SABEIRO está despegando
Estado de LA_SABEIRO Combustible: 0
BUILD SUCCESSFUL (total time: 0 seconds)

```