

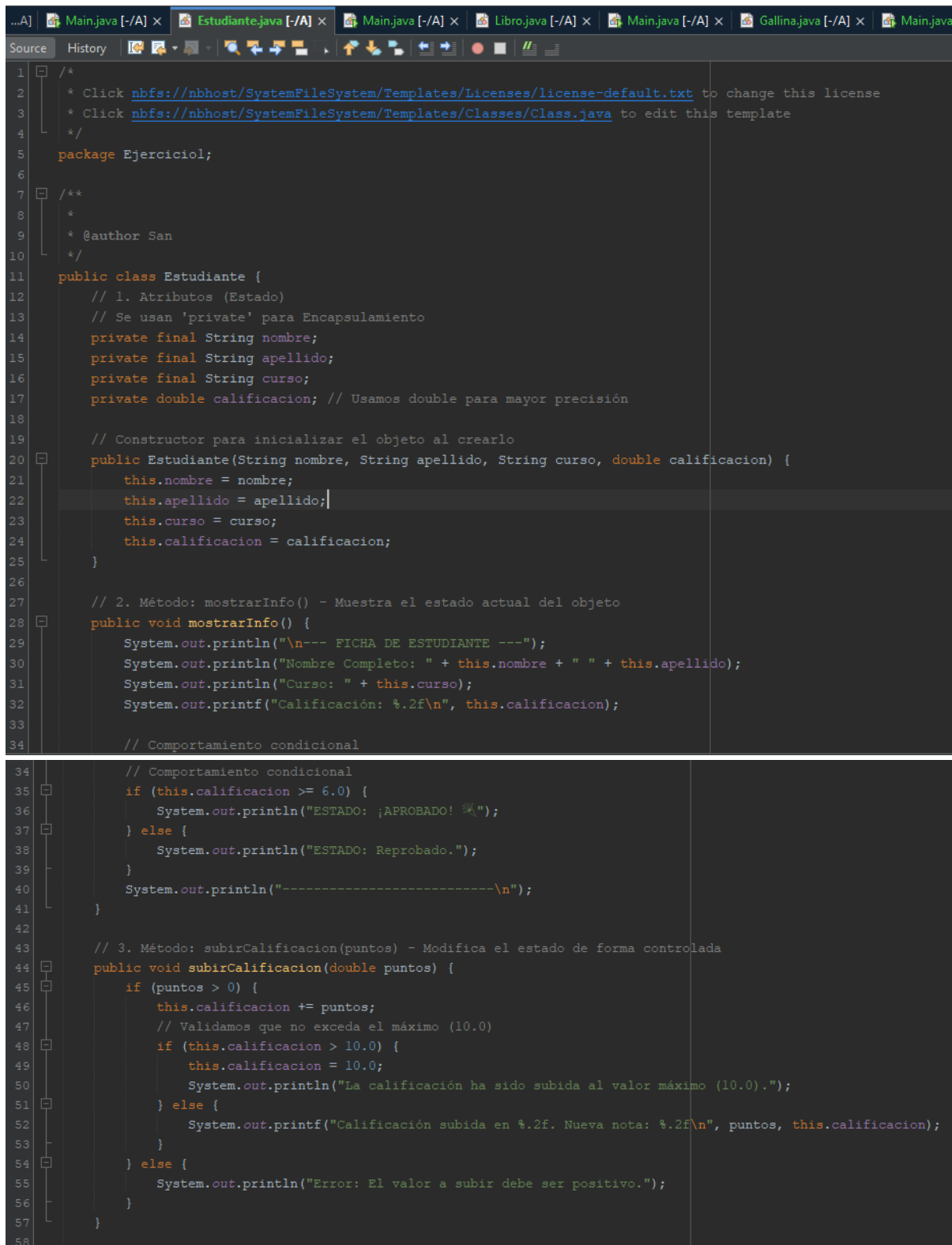
# PROGRAMACIÓN II

## Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

Alumno: Santiago Raúl Salinas

### Ejercicio 1:

#### Clase: Estudiante



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package Ejercicio1;
6
7  /**
8   *
9   * @author San
10  */
11  public class Estudiante {
12      // 1. Atributos (Estado)
13      // Se usan 'private' para Encapsulamiento
14      private final String nombre;
15      private final String apellido;
16      private final String curso;
17      private double calificacion; // Usamos double para mayor precisión
18
19      // Constructor para inicializar el objeto al crearlo
20      public Estudiante(String nombre, String apellido, String curso, double calificacion) {
21          this.nombre = nombre;
22          this.apellido = apellido;
23          this.curso = curso;
24          this.calificacion = calificacion;
25      }
26
27      // 2. Método: mostrarInfo() - Muestra el estado actual del objeto
28      public void mostrarInfo() {
29          System.out.println("\n--- FICHA DE ESTUDIANTE ---");
30          System.out.println("Nombre Completo: " + this.nombre + " " + this.apellido);
31          System.out.println("Curso: " + this.curso);
32          System.out.printf("Calificación: %.2f\n", this.calificacion);
33
34          // Comportamiento condicional
35          if (this.calificacion >= 6.0) {
36              System.out.println("ESTADO: ¡APROBADO! 🎉");
37          } else {
38              System.out.println("ESTADO: Reprobado.");
39          }
40          System.out.println("-----\n");
41      }
42
43      // 3. Método: subirCalificacion(puntos) - Modifica el estado de forma controlada
44      public void subirCalificacion(double puntos) {
45          if (puntos > 0) {
46              this.calificacion += puntos;
47              // Validamos que no exceda el máximo (10.0)
48              if (this.calificacion > 10.0) {
49                  this.calificacion = 10.0;
50                  System.out.println("La calificación ha sido subida al valor máximo (10.0).");
51              } else {
52                  System.out.printf("Calificación subida en %.2f. Nueva nota: %.2f\n", puntos, this.calificacion);
53              }
54          } else {
55              System.out.println("Error: El valor a subir debe ser positivo.");
56          }
57      }
58  }
```

```

59 // 4. Método: bajarCalificacion(puntos) - Modifica el estado de forma controlada
60 public void bajarCalificacion(double puntos) {
61     if (puntos > 0) {
62         this.calificacion -= puntos;
63         // Validamos que no sea menor que el minimo (0.0)
64         if (this.calificacion < 0.0) {
65             this.calificacion = 0.0;
66             System.out.println("La calificación ha sido bajada al valor mínimo (0.0).");
67         } else {
68             System.out.printf("Calificación bajada en %.2f. Nueva nota: %.2f\n", puntos, this.calificacion);
69         }
70     } else {
71         System.out.println("Error: El valor a bajar debe ser positivo.");
72     }
73 }
74 }

```

## Main:

```

...A] Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x Gallina.java [-/A] x Main.java [-/A] x
Source History
9 * @author San
10 */
11 public class Main {
12     public static void main(String[] args) {
13
14         // Tarea 1: Instanciar a un estudiante (Creación de Objeto)
15         // Se inicializa con una calificación de 5.5
16         Estudiante ana = new Estudiante("Ana", "Gómez", "Programación Orientada a Objetos", 5.5);
17
18         System.out.println("### ESTADO INICIAL DEL OBJETO 'ana' ###");
19         // Tarea 2: Mostrar su información
20         ana.mostrarInfo();
21
22         // Tarea 3: Aumentar calificación (Comportamiento que cambia el Estado)
23         System.out.println("\n>>> Se sube la calificación por trabajo extra (1.5 puntos) <<<");
24         ana.subirCalificacion(1.5); // 5.5 + 1.5 = 7.0
25         ana.mostrarInfo();
26
27         // Prueba de limite superior
28         System.out.println("\n>>> Se intenta subir la calificación con un valor muy alto (5.0 puntos) <<<");
29         ana.subirCalificacion(5.0); // 7.0 + 5.0 = 12.0 (se limitará a 10.0)
30         ana.mostrarInfo();
31
32         // Tarea 4: Disminuir calificaciones (Comportamiento que cambia el Estado)
33         System.out.println("\n>>> Se baja la calificación por un error en un examen (2.0 puntos) <<<");
34         ana.bajarCalificacion(2.0); // 10.0 - 2.0 = 8.0
35
36         System.out.println("\n### ESTADO FINAL DEL OBJETO 'ana' ###");
37         // Tarea 5: Mostrar la información final
38         ana.mostrarInfo();
39     }
40 }

```

## Output:

```
Output x
x JavaApplication3 (run) #9 x JavaApplication3 (run) #10 x JavaApplication3 (run) #11 x JavaApplication3 (run) #12 x TP 3 - POO (run) x
run:
### ESTADO INICIAL DEL OBJETO 'ana' ###

--- FICHA DE ESTUDIANTE ---
Nombre Completo: Ana Gómez
Curso: Programación Orientada a Objetos
Calificación: 5,50
ESTADO: Reprobado.
-----

>>> Se sube la calificación por trabajo extra (1.5 puntos) <<<
Calificación subida en 1,50. Nueva nota: 7,00

--- FICHA DE ESTUDIANTE ---
Nombre Completo: Ana Gómez
Curso: Programación Orientada a Objetos
Calificación: 7,00
ESTADO: APROBADO! ?
-----

>>> Se intenta subir la calificación con un valor muy alto (5.0 puntos) <<<
La calificación ha sido subida al valor máximo (10.0).

--- FICHA DE ESTUDIANTE ---
Nombre Completo: Ana Gómez
Curso: Programación Orientada a Objetos
Calificación: 10,00
ESTADO: APROBADO! ?
-----

>>> Se baja la calificación por un error en un examen (2.0 puntos) <<<
Calificación bajada en 2,00. Nueva nota: 8,00

### ESTADO FINAL DEL OBJETO 'ana' ###

--- FICHA DE ESTUDIANTE ---
Nombre Completo: Ana Gómez
Curso: Programación Orientada a Objetos
Calificación: 8,00
ESTADO: APROBADO! ?
-----

BUILD SUCCESSFUL (total time: 0 seconds)
```

## Ejercicio 2:

### Clase: Registro de Mascotas

```
...A] RegistroMascotas.java [-/A] x Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x G
Source History
3 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4 */
5 package Ejercicio2;
6
7 /**
8  *
9  * @author San
10 */
11 public class RegistroMascotas {
12     // 1. Atributos (Estado)
13     private final String nombre;
14     private final String especie;
15     private int edad;
16
17     // Constructor para inicializar el objeto al crearlo
18     public RegistroMascotas(String nombre, String especie, int edadInicial) {
19         this.nombre = nombre;
20         this.especie = especie;
21         this.edad = edadInicial;
22     }
23
24     // 2. Método: mostrarInfo() - Muestra el estado actual del objeto
25     public void mostrarInfo() {
26         System.out.println("\n--- FICHA DE MASCOTA ---");
27         System.out.println("Nombre: " + this.nombre);
28         System.out.println("Especie: " + this.especie);
29         System.out.println("Edad: " + this.edad + " años");
30         System.out.println("-----\n");
31     }
32
33     // 3. Método: cumplirAños() - Modifica el estado del objeto
34     public void cumplirAños() {
35         // Incrementa el atributo 'edad' (cambio de estado)
36         this.edad++;
37         System.out.println("¡Feliz cumpleaños, " + this.nombre + "! Ahora tiene " + this.edad + " años");
38     }
39 }
```

## Main:

```
...A] RegistroMascotas.java [-/A] x Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x M
Source History
2 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4 */
5 package Ejercicio2;
6
7 /**
8  *
9  * @author San
10 */
11 public class Main {
12     public static void main(String[] args) {
13
14         // Tarea 1: Crear una mascota (Instanciación de Objeto)
15         RegistroMascotas firulais = new RegistroMascotas("Firulais", "Perro", 3);
16
17         System.out.println("### ESTADO INICIAL DEL OBJETO 'firulais' ###");
18         // Tarea 2: Mostrar su información
19         firulais.mostrarInfo();
20
21         // Tarea 3: Simular el paso del tiempo y verificar los cambios (Comportamiento)
22         System.out.println(">>> Simulando que Firulais cumple un año más... <<<");
23         firulais.cumplirAños(); // 3 -> 4
24
25         System.out.println("\n>>> Simulando que Firulais cumple otro año más... <<<");
26         firulais.cumplirAños(); // 4 -> 5
27
28         System.out.println("\n### ESTADO FINAL DEL OBJETO 'firulais' ###");
29         // Tarea 4: Verificar el cambio de estado
30         firulais.mostrarInfo();
31     }
32 }
```

## OutPut:

```
Output x
JavaApplication3 (run) #9 x JavaApplication3 (run) #10 x Java
run:
### ESTADO INICIAL DEL OBJETO 'firulais' ###

--- FICHA DE MASCOTA ---
Nombre: Firulais
Especie: Perro
Edad: 3 años
-----

>>> Simulando que Firulais cumple un año más... <<<
Feliz cumple años, Firulais! Ahora tiene 4 años

>>> Simulando que Firulais cumple otro año más... <<<
Feliz cumple años, Firulais! Ahora tiene 5 años

### ESTADO FINAL DEL OBJETO 'firulais' ###

--- FICHA DE MASCOTA ---
Nombre: Firulais
Especie: Perro
Edad: 5 años
-----

BUILD SUCCESSFUL (total time: 0 seconds)
```

### Ejercicio 3:

#### Clase: Libro

```
...A] RegistroMascotas.java [-/A] x Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x Gallina.java
Source History
1 /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5 package Ejercicio3;
6
7 /**
8  *
9  * @author San
10  */
11 public class Libro {
12     // 1. Atributos Privados para Encapsulamiento
13     private String titulo;
14     private String autor;
15     private int añoPublicacion; // Atributo que será validado
16
17     // Constructor para inicializar el objeto
18     public Libro(String titulo, String autor, int añoPublicacion) {
19         this.titulo = titulo;
20         this.autor = autor;
21         // El constructor también puede usar el setter para validar el valor inicial
22         setAñoPublicacion(añoPublicacion);
23     }
24
25     // 2. Getters (Métodos de Acceso para lectura)
26     public String getTitulo() {
27         return titulo;
28     }
29
30     public String getAutor() {
31         return autor;
32     }
33
34     public int getAñoPublicacion() {
35         return añoPublicacion;
36     }
37 }
```

```

37
38 // 3. Setter con Validación (Método Mutador para escritura controlada)
39 public void setAñoPublicacion(int nuevoAño) {
40     // Obtenemos el año actual para la validación
41     int añoActual = 2025; // Usamos 2025 como referencia para el contexto del ejercicio
42     int añoMínimo = 1000;
43
44     // Lógica de Validación
45     if (nuevoAño > añoActual) {
46         System.out.println("X ERROR: El año de publicación (" + nuevoAño + ") no puede ser en el futuro. No se ha modificado el año.");
47     } else if (nuevoAño < añoMínimo) {
48         System.out.println("X ERROR: El año de publicación (" + nuevoAño + ") es demasiado antiguo. No se ha modificado el año.");
49     } else {
50         // Si la validación pasa, modificamos el estado interno (Encapsulamiento exitoso)
51         this.añoPublicacion = nuevoAño;
52         System.out.println("✓ Éxito: Año de publicación actualizado a " + nuevoAño + ".");
53     }
54 }
55
56 // Método auxiliar para mostrar toda la información
57 public void mostrarInfo() {
58     System.out.println("\n--- DETALLES DEL LIBRO ---");
59     System.out.println("Título: " + this.titulo);
60     System.out.println("Autor: " + this.autor);
61     System.out.println("Año de Publicación: " + this.añoPublicacion);
62     System.out.println("-----\n");
63 }
64 }

```

**Main:**

```

...A] Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x Gallina.java [-/A] x
Source History
9  * @author San
10  */
11  public class Main {
12      public static void main(String[] args) {
13
14          // Tarea 1: Crear un libro (Instanciación de Objeto)
15          // Se inicializa con un año válido.
16          Libro miLibro = new Libro("Cien Años de Soledad", "Gabriel García Márquez", 1967);
17
18          System.out.println("### ESTADO INICIAL DEL OBJETO ###");
19          miLibro.mostrarInfo();
20
21          // Tarea 2: Intentar modificar el año con un valor INVÁLIDO (Futuro)
22          System.out.println("\n>>> Intento de modificación: Año 2030 (Inválido) <<<");
23          miLibro.setAñoPublicacion(2030);
24
25          // Tarea 3: Intentar modificar el año con un valor INVÁLIDO (Muy Antiguo)
26          System.out.println("\n>>> Intento de modificación: Año 900 (Inválido) <<<");
27          miLibro.setAñoPublicacion(900);
28
29          // Tarea 4: Intentar modificar el año con un valor VÁLIDO
30          System.out.println("\n>>> Intento de modificación: Año 1982 (Válido) <<<");
31          miLibro.setAñoPublicacion(1982);
32
33          System.out.println("\n### INFORMACIÓN FINAL DEL LIBRO ###");
34          // Tarea 5: Mostrar la información final para verificar los cambios
35          miLibro.mostrarInfo();
36
37          // Acceso controlado al atributo usando el getter
38          System.out.println("Verificación con Getter: " + miLibro.getAñoPublicacion());
39      }
40  }

```

**Output:**

```
JavaApplication3 (run) #9 x JavaApplication3 (run) #10 x JavaApplication3 (run) #11 x JavaApplication3 (run) #12

run:
? Éxito: Año de publicación actualizado a 1967.
### ESTADO INICIAL DEL OBJETO ###

--- DETALLES DEL LIBRO ---
Título: Cien Años de Soledad
Autor: Gabriel García Márquez
Año de Publicación: 1967
-----

>>> Intento de modificación: Año 2030 (Inválido) <<<
? ERROR: El año de publicación (2030) no puede ser en el futuro. No se ha modificado el año.

>>> Intento de modificación: Año 900 (Inválido) <<<
? ERROR: El año de publicación (900) es demasiado antiguo. No se ha modificado el año.

>>> Intento de modificación: Año 1982 (Válido) <<<
? Éxito: Año de publicación actualizado a 1982.

### INFORMACIÓN FINAL DEL LIBRO ###

--- DETALLES DEL LIBRO ---
Título: Cien Años de Soledad
Autor: Gabriel García Márquez
Año de Publicación: 1982
-----

Verificación con Getter: 1982
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Ejercicio 4:

### Clase: Gallina

```
...A] Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x Gallina.java [-/A] x Main.java [-/A] x

Source History
9  * @author San
10  */
11  public class Gallina {
12      // 1. Atributos (Estado) - Utilizamos 'private' para encapsulamiento
13      private String idGallina;
14      private int edad; // en meses o años, lo que se prefiera simular
15      private int huevosPuestos;
16
17      // Constructor para inicializar la gallina al crear el objeto
18      public Gallina(String id, int edadInicial) {
19          this.idGallina = id;
20          this.edad = edadInicial;
21          this.huevosPuestos = 0; // Siempre inicia en cero
22      }
23
24      // 2. Método: ponerHuevo() - Incrementa el contador de huevos (Comportamiento)
25      public void ponerHuevo() {
26          // Lógica de negocio: incrementa el estado 'huevosPuestos'
27          this.huevosPuestos++;
28          System.out.println("🐔 " + this.idGallina + " ha puesto un huevo. Total: " + this.huevosPuestos);
29      }
30
31      // 3. Método: envejecer() - Incrementa la edad (Cambio de Estado)
32      public void envejecer() {
33          this.edad++;
34          System.out.println("🐣 " + this.idGallina + " ha envejecido un año/mes. Edad actual: " + this.edad);
35      }
36
37      // 4. Método: mostrarEstado() - Muestra la información actual de la gallina
38      public void mostrarEstado() {
39          System.out.println("\n--- ESTADO DE GALLINA [" + this.idGallina + "] ---");
40          System.out.println("Edad: " + this.edad);
41          System.out.println("Huevos Totales Puestos: " + this.huevosPuestos);
42          System.out.println("-----");
43      }
44  }
```

Main:

```
5 package Ejercicio4;
6
7 /**
8  *
9  * @author San
10  */
11 public class Main {
12     public static void main(String[] args) {
13
14         // Tarea 1: Crear dos gallinas (Instanciación de Objetos)
15         Gallina gallinal = new Gallina("G-001", 1); // Gallina joven
16         Gallina gallina2 = new Gallina("G-002", 3); // Gallina adulta
17
18         System.out.println("### ESTADO INICIAL ###");
19         gallinal.mostrarEstado();
20         gallina2.mostrarEstado();
21
22         // Tarea 2: Simular acciones de Gallina 1
23         System.out.println("\n>>> Acciones de Gallina G-001 (La joven) <<<");
24         gallinal.ponerHuevo();
25         gallinal.ponerHuevo();
26         gallinal.envejecer();
27
28         // Tarea 3: Simular acciones de Gallina 2
29         System.out.println("\n>>> Acciones de Gallina G-002 (La adulta) <<<");
30         gallina2.envejecer();
31         gallina2.ponerHuevo();
32         gallina2.ponerHuevo();
33         gallina2.ponerHuevo();
34
35         // Tarea 4: Mostrar el estado final
36         System.out.println("\n### ESTADO FINAL ###");
37         gallinal.mostrarEstado();
38         gallina2.mostrarEstado();
39     }
40 }
```

Output:

```
run:
### ESTADO INICIAL ###

--- ESTADO DE GALLINA [G-001] ---
Edad: 1
Huevos Totales Puestos: 0
-----

--- ESTADO DE GALLINA [G-002] ---
Edad: 3
Huevos Totales Puestos: 0
-----

>>> Acciones de Gallina G-001 (La joven) <<<
? G-001 ha puesto un huevo. Total: 1
? G-001 ha puesto un huevo. Total: 2
? G-001 ha envejecido un a❖o/mes. Edad actual: 2

>>> Acciones de Gallina G-002 (La adulta) <<<
? G-002 ha envejecido un a❖o/mes. Edad actual: 4
? G-002 ha puesto un huevo. Total: 1
? G-002 ha puesto un huevo. Total: 2
? G-002 ha puesto un huevo. Total: 3

### ESTADO FINAL ###

--- ESTADO DE GALLINA [G-001] ---
Edad: 2
Huevos Totales Puestos: 2
-----

--- ESTADO DE GALLINA [G-002] ---
Edad: 4
Huevos Totales Puestos: 3
-----

BUILD SUCCESSFUL (total time: 0 seconds)
```

## Ejercicio 5:

### Clase: Nave Espacial



```
...A] Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x Gallina.java [-/A] x Main.java [-/A] x NaveEspacial.java [-/A] x
Source History
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package Ejercicio5;

/**
 *
 * @author San
 */
public class NaveEspacial {
    // Atributos (Estado) - Privados para Encapsulamiento
    private String nombre;
    private int combustible; // En unidades
    private final int MAX_COMBUSTIBLE = 100; // Limite máximo de combustible

    // Constructor
    public NaveEspacial(String nombre, int combustibleInicial) {
        this.nombre = nombre;
        // Se asegura que el combustible inicial no exceda el limite
        if (combustibleInicial > MAX_COMBUSTIBLE) {
            this.combustible = MAX_COMBUSTIBLE;
        } else {
            this.combustible = combustibleInicial;
        }
    }

    // Método Despegar - Comportamiento simple
    public void despegar() {
        if (combustible >= 20) {
            combustible -= 20; // Costo de despegue
            System.out.println("¡Despegue exitoso! Consumo inicial de 20u.");
        } else {
            System.out.println("X ERROR: Combustible insuficiente para despegar. Se requieren 20u.");
        }
    }
}
```

```
...A] Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x Gallina.java [-/A] x Main.java [-/A] x NaveEspacial.java [-/A] x Main.java [-/A] x
Source History
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
// Método Avanzar - Comportamiento con Validación (Regla 1)
public boolean avanzar(int distancia) {
    int costo = distancia / 2; // Ejemplo: 1u de combustible por cada 2 unidades de distancia

    if (this.combustible >= costo) {
        this.combustible -= costo;
        System.out.printf("== %s ha avanzado %d unidades (consumo: %d). Combustible restante: %d.\n", this.nombre, distancia, costo, this.combustible);
        return true;
    } else {
        System.out.printf("X ERROR: No hay suficiente combustible para avanzar %d unidades. Costo: %d. Disponible: %d.\n", distancia, costo, this.combustible);
        return false;
    }
}

// Método Recargar Combustible - Comportamiento con Limite (Regla 2)
public void recargarCombustible(int cantidad) {
    int nuevoTotal = this.combustible + cantidad;

    if (nuevoTotal <= MAX_COMBUSTIBLE) {
        this.combustible = nuevoTotal;
        System.out.printf("M Recarga exitosa: %d añadidas. Total actual: %d.\n", cantidad, this.combustible);
    } else {
        // Si supera el limite, solo se recarga hasta el máximo.
        int recargaEfectiva = MAX_COMBUSTIBLE - this.combustible;
        this.combustible = MAX_COMBUSTIBLE;
        System.out.printf("A Advertencia: Solo se recargaron %d. Limite (%d) alcanzado. Total actual: %d.\n", recargaEfectiva, MAX_COMBUSTIBLE, this.combustible);
    }
}

// Método Mostrar Estado
public void mostrarEstado() {
    System.out.println("\n== ESTADO ACTUAL DE LA NAVE ==");
    System.out.println("Nombre: " + this.nombre);
    System.out.printf("Combustible: %d / %d (Máx)\n", this.combustible, MAX_COMBUSTIBLE);
    System.out.println("=====\n");
}
}
```

Main:

```
Source History
Main.java [-/A] x Estudiante.java [-/A] x Main.java [-/A] x Libro.java [-/A] x Main.java [-/A] x Gallina.java [-/A] x NaveEspacial.java [-/A] x Main.java [-/A] x

8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

* @author San
*/
public class Main {
    public static void main(String[] args) {

        // Tarea 1: Crear una nave con 50 unidades de combustible
        NaveEspacial enterprise = new NaveEspacial("Enterprise NCC-1701", 50);

        System.out.println("### INICIO DE LA SIMULACIÓN ###");
        enterprise.mostrarEstado();

        // Despegue (consume 20u). Saldo: 30u
        enterprise.despegar();

        // Tarea 2: Intentar avanzar sin recargar (Costo: 50u. Saldo: 30u)
        System.out.println("\n--- Intento 1: Avanzar 100u (Costo 50u) ---");
        enterprise.avanzar(100);

        // Tarea 3: Recargar (50u). Saldo: 30u + 50u = 80u
        System.out.println("\n--- Recargando combustible (50u) ---");
        enterprise.recargarCombustible(50);

        // Prueba de limite de recarga (Intenta añadir 50u, pero solo le faltan 20u para 100).
        System.out.println("\n--- Intento de Recarga Excesiva (50u) ---");
        enterprise.recargarCombustible(50);

        // Tarea 4: Avanzar correctamente (Costo: 30u. Saldo: 100u - 30u = 70u)
        System.out.println("\n--- Intento 2: Avanzar 60u (Costo 30u) ---");
        enterprise.avanzar(60);

        // Tarea 5: Mostrar el estado final
        System.out.println("\n### ESTADO FINAL ###");
        enterprise.mostrarEstado();
    }
}
```

## Output:

```
Output x
JavaApplication3 (run) #9 x JavaApplication3 (run) #10 x JavaApplication3 (run) #11 x JavaApplication3 (run) #12 x

run:
### INICIO DE LA SIMULACIÓN ###

=== ESTADO ACTUAL DE LA NAVE ===
Nombre: Enterprise NCC-1701
Combustible: 50u / 100u (Mx)
=====

? Despegue exitoso! Consumo inicial de 20u.

--- Intento 1: Avanzar 100u (Costo 50u) ---
? ERROR: No hay suficiente combustible para avanzar 100 unidades. Costo: 50u. Disponible: 30u.

--- Recargando combustible (50u) ---
? Recarga exitosa: 50u añadidas. Total actual: 80u.

--- Intento de Recarga Excesiva (50u) ---
?? Advertencia: Solo se recargaron 20u. Límite (100u) alcanzado. Total actual: 100u.

--- Intento 2: Avanzar 60u (Costo 30u) ---
?? Enterprise NCC-1701 ha avanzado 60 unidades (consumo: 30u). Combustible restante: 70u.

### ESTADO FINAL ###

=== ESTADO ACTUAL DE LA NAVE ===
Nombre: Enterprise NCC-1701
Combustible: 70u / 100u (Mx)
=====

BUILD SUCCESSFUL (total time: 0 seconds)
```