Kevin Acuña

Ka246410@fi365.ort.edu.uy

Marzo 2024

Carrera: Analista en Tecnologías de la Información  
Docente: Santiago Baillo  
Grupo: N2B  
Número Estudiante: 246410

Obligatorio 1

Programación 2

[Diagrama de Clases 2](#_Toc165508261)

[URL Imagen JPG ClassDiagram 2](#_Toc165508262)

[URL Imagen SVG ClassDiagram 2](#_Toc165508263)

[Tabla con la Información 3](#_Toc165508264)

[Datos precargados 3](#_Toc165508265)

[Peón 3](#_Toc165508266)

[Tarea 3](#_Toc165508267)

[Capataz 4](#_Toc165508268)

[Ganado Ovino 4](#_Toc165508269)

[Ganado Bovino 5](#_Toc165508270)

[Vacuna 6](#_Toc165508271)

[Potrero 7](#_Toc165508272)

[Registro Vacunación 7](#_Toc165508273)

[Asignación de Animales al Potrero 8](#_Toc165508274)

[Asignación de Tareas al Peón 9](#_Toc165508275)

[Código Fuente 10](#_Toc165508276)

[ConsoleApp\Program.cs 10](#_Toc165508277)

[ClassLibrary\Sistema.cs 29](#_Toc165508278)

[ClassLibrary\Enum\Sexo 48](#_Toc165508279)

[ClassLibrary\Enum\TipoAlimentacion 48](#_Toc165508280)

[ClassLibrary\Enum\TipoAnimal 48](#_Toc165508281)

[ClassLibrary\Interface\IValidar 49](#_Toc165508282)

[ClassLibrary\Animal 49](#_Toc165508283)

[ClassLibrary\Bovino 51](#_Toc165508284)

[ClassLibrary\Capataz 52](#_Toc165508285)

[ClassLibrary\Empleado 53](#_Toc165508286)

[ClassLibrary\Ovino 54](#_Toc165508287)

[ClassLibrary\Peon 56](#_Toc165508288)

[ClassLibrary\Potrero 57](#_Toc165508289)

[ClassLibrary\Tarea 59](#_Toc165508290)

[ClassLibrary\Vacuna 60](#_Toc165508291)

[ClassLibrary\Vacunacion 61](#_Toc165508292)

# Diagrama de Clases

## [URL Imagen JPG ClassDiagram](https://raw.githubusercontent.com/kevinacuna0194/P2-N2B-Obligatorio1-Estancia/main/img/P2-N2B-Obligatorio1-ClassDiagram.jpg)

## [URL Imagen SVG ClassDiagram](https://raw.githubusercontent.com/kevinacuna0194/P2-N2B-Obligatorio1-Estancia/f5c264206146eddbe8e27f8e076482acf5f722d2/img/P2-N2B-Obligatorio1-ClassDiagram.svg)

# Tabla con la Información

## Datos precargados

### Peón

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Email | Password | Nombre | Fecha Ingreso | ¿Residente Estancia? |
| peon1@email.com | password1 | Juan | 2022, 1, 1 | True |
| peon2@email.com | password2 | María | 2022, 1, 2 | False |
| peon3@email.com | password3 | Carlos | 2022, 1, 3 | True |
| peon4@email.com | password4 | Ana | 2022, 1, 4 | False |
| peon5@email.com | password5 | Pedro | 2022, 1, 5 | True |
| peon6@email.com | password6 | Luis | 2022, 1, 6 | False |
| peon7@email.com | password7 | Sofía | 2022, 1, 7 | True |
| peon8@email.com | password8 | Elena | 2022, 1, 8 | False |
| peon9@email.com | password9 | Diego | 2022, 1, 9 | True |
| peon10@email.com | password10 | Laura | 2022, 1, 10 | False |

### Tarea

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nº | Descripción | Fecha Pactada | ¿Completada? | Fecha Cierre | Comentario |
| 1 | Preparar terreno para siembra | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(2) | Se necesita arar y fertilizar el terreno |
| 2 | Sembrar cultivo de maíz | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(4) | Sembrar el maíz en las parcelas asignadas |
| 3 | Regar cultivos | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(6) | Asegurarse de mantener una hidratación adecuada |
| 4 | Fertilizar cultivos | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(8) | Aplicar fertilizantes según las necesidades del suelo |
| 5 | Control de plagas | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(10) | Monitorear y aplicar tratamientos contra plagas |
| 6 | Podar árboles frutales | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(12) | Realizar poda de forma adecuada para promover el crecimiento |
| 7 | Cosechar cultivos | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(14) | Recolectar los cultivos en el momento óptimo |
| 8 | Inspección de cercas | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(16) | Revisar la integridad de las cercas del campo |
| 9 | Reparación de maquinaria agrícola | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(18) | Realizar mantenimiento y reparaciones según sea necesario |
| 10 | Control de malezas | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(20) | Eliminar malezas que puedan competir con los cultivos |
| 11 | Fertilizar terrenos vacíos | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(22) | Aplicar fertilizantes en áreas sin cultivos |
| 12 | Revisión de sistemas de riego | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(24) | Asegurarse de que los sistemas de riego estén funcionando correctamente |
| 13 | Control de humedad en suelo | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(26) | Monitorear niveles de humedad y ajustar riego según sea necesario |
| 14 | Cercar área de pastoreo | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(28) | Instalar cercas temporales para el pastoreo del ganado |
| 15 | Preparar suelos para próximas siembras | DateTime.Today.AddDays(1) | False | DateTime.Today.AddDays(30) | Arar y acondicionar suelos para futuras siembras |

### Capataz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Email | Password | Nombre | Fecha Ingreso | Cantidad Personas a Cargo |
| capataz1@email.com | password1 | Juan | 2022/1/1 | 10 |
| capataz2@email.com | password2 | María | 2022/1/2 | 8 |

### Ganado Ovino

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nº | Código Caravana | Sexo | Raza | Fecha Nacimiento | Costo Adquisición | Costo Alimentación | Peso Actual | ¿Es Híbrido? | Peso Lana | Precio Kilo Lana | Precio Kilo en Pie |
| 1 | Caravana1 | Macho | Raza1 | 2019, 01, 15 | 1500 | 200 | 30.5 | false | 5.2 | 15 | 25 |
| 2 | Caravana2 | Hembra | Raza2 | 2020, 03, 22 | 1600 | 220 | 35.2 | true | 6.8 | 18 | 21 |
| 3 | Caravana3 | Macho | Raza3 | 2021, 05, 10 | 1700 | 240 | 32.7 | false | 5.9 | 16 | 25 |
| 4 | Caravana4 | Hembra | Raza4 | 2022, 07, 03 | 1800 | 260 | 38.1 | true | 7.5 | 20 | 23 |
| 5 | Caravana5 | Macho | Raza5 | 2023, 09, 18 | 1900 | 280 | 36.8 | false | 6.3 | 17 | 27 |
| 6 | Caravana6 | Hembra | Raza6 | 2024, 11, 25 | 2000 | 300 | 42.4 | true | 8.1 | 22 | 24 |
| 7 | Caravana7 | Macho | Raza7 | 2025, 12, 10 | 2100 | 320 | 39.6 | false | 7.2 | 19 | 30 |
| 8 | Caravana8 | Hembra | Raza8 | 2026, 10, 06 | 2200 | 340 | 45.3 | true | 9.3 | 25 | 28 |
| 9 | Caravana9 | Macho | Raza9 | 2027, 08, 30 | 2300 | 360 | 43.9 | false | 8.5 | 23 | 33 |
| 10 | Caravana10 | Hembra | Raza10 | 2028, 07, 20 | 2400 | 380 | 49.7 | true | 10.2 | 27 | 22 |
| 11 | Caravana11 | Macho | Raza11 | 2018, 02, 14 | 1500 | 200 | 34.2 | false | 6.1 | 17 | 26 |
| 12 | Caravana12 | Hembra | Raza12 | 2017, 04, 03 | 1600 | 220 | 40.5 | true | 7.9 | 21 | 23 |
| 13 | Caravana13 | Macho | Raza13 | 2016, 06, 27 | 1700 | 240 | 37.8 | false | 7.0 | 18 | 29 |
| 14 | Caravana14 | Hembra | Raza14 | 2015, 08, 12 | 1800 | 260 | 44.6 | true | 8.8 | 24 | 25 |
| 15 | Caravana15 | Macho | Raza15 | 2014, 10, 05 | 1900 | 280 | 41.3 | false | 7.7 | 20 | 31 |
| 16 | Caravana16 | Hembra | Raza16 | 2013, 12, 20 | 2000 | 300 | 47.9 | true | 9.6 | 26 | 27 |
| 17 | Caravana17 | Macho | Raza17 | 2012, 07, 18 | 2100 | 320 | 45.2 | false | 8.4 | 22 | 34 |
| 18 | Caravana18 | Hembra | Raza18 | 2011, 05, 30 | 2200 | 340 | 51.8 | true | 10.7 | 28 | 29 |
| 19 | Caravana19 | Macho | Raza19 | 2010, 03, 25 | 2300 | 360 | 49.1 | false | 9.2 | 24 | 35 |
| 20 | Caravana20 | Hembra | Raza20 | 2009, 01, 15 | 2400 | 380 | 55.7 | true | 11.3 | 30 | 40 |
| 21 | Caravana21 | Macho | Raza21 | 2010, 1, 1 | 1500 | 200 | 30.5 | false | 5.2 | 15 | 45 |
| 22 | Caravana22 | Hembra | Raza22 | 2011, 1, 1 | 1600 | 220 | 35.2 | true | 6.8 | 18 | 50 |
| 23 | Caravana23 | Macho | Raza23 | 2012, 1, 1 | 1700 | 240 | 32.7 | false | 5.9 | 16 | 50 |
| 24 | Caravana24 | Hembra | Raza24 | 2013, 1, 1 | 1800 | 260 | 38.1 | true | 7.5 | 20 | 55 |
| 25 | Caravana25 | Macho | Raza25 | 2014, 1, 1 | 1900 | 280 | 36.8 | false | 6.3 | 17 | 60 |
| 26 | Caravana26 | Hembra | Raza26 | 2015, 1, 1 | 2000 | 300 | 42.4 | true | 8.1 | 22 | 65 |
| 27 | Caravana27 | Macho | Raza27 | 2016, 1, 1 | 2100 | 320 | 39.6 | false | 7.2 | 19 | 70 |
| 28 | Caravana28 | Hembra | Raza28 | 2017, 1, 1 | 2200 | 340 | 45.3 | true | 9.3 | 25 | 75 |
| 29 | Caravana29 | Macho | Raza29 | 2018, 1, 1 | 2300 | 360 | 43.9 | false | 8.5 | 23 | 80 |
| 30 | Caravana30 | Hembra | Raza30 | 2019, 1, 1 | 2400 | 380 | 49.7 | true | 10.2 | 27 | 85 |

### Ganado Bovino

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nº | Código Caravana | Sexo | Raza | Fecha Nacimiento | Costo Adquisición | Costo Alimentación | Peso Actual | ¿Es Híbrido? | Tipo Alimentación | Precio Kilo Lana |
| 1 | Caravana1 | Macho | Angus | 2019, 01, 15 | 1500 | 200 | 300 | false | Grano | 25 |
| 2 | Caravana2 | Hembra | Hereford | 2020, 03, 22 | 1600 | 220 | 320 | true | Pastura | 30 |
| 3 | Caravana3 | Macho | Simmental | 2021, 05, 10 | 1700 | 240 | 340 | false | Grano | 35 |
| 4 | Caravana4 | Hembra | Angus | 2022, 07, 03 | 1800 | 260 | 360 | true | Pastura | 40 |
| 5 | Caravana5 | Macho | Hereford | 2023, 09, 18 | 1900 | 280 | 380 | false | Grano | 45 |
| 6 | Caravana6 | Hembra | Simmental | 2024, 11, 25 | 2000 | 300 | 400 | true | Pastura | 50 |
| 7 | Caravana7 | Macho | Angus | 2025, 12, 10 | 2100 | 320 | 420 | false | Grano | 55 |
| 8 | Caravana8 | Hembra | Hereford | 2026, 10, 06 | 2200 | 340 | 440 | true | Pastura | 60 |
| 9 | Caravana9 | Macho | Simmental | 2027, 08, 30 | 2300 | 360 | 460 | false | Grano | 65 |
| 10 | Caravana10 | Hembra | Angus | 2028, 07, 20 | 2400 | 380 | 480 | true | Pastura | 70 |
| 11 | Caravana11 | Macho | Hereford | 2018, 02, 14 | 1500 | 200 | 300 | false | Grano | 75 |
| 12 | Caravana12 | Hembra | Simmental | 2017, 04, 03 | 1600 | 220 | 320 | true | Pastura | 80 |
| 13 | Caravana13 | Macho | Angus | 2016, 06, 27 | 1700 | 240 | 340 | false | Grano | 85 |
| 14 | Caravana14 | Hembra | Hereford | 2015, 08, 12 | 1800 | 260 | 360 | true | Pastura | 90 |
| 15 | Caravana15 | Macho | Simmental | 2014, 10, 05 | 1900 | 280 | 380 | false | Grano | 95 |
| 16 | Caravana16 | Hembra | Angus | 2013, 12, 20 | 2000 | 300 | 400 | true | Pastura | 100 |
| 17 | Caravana17 | Macho | Hereford | 2012, 07, 18 | 2100 | 320 | 420 | false | Grano | 105 |
| 18 | Caravana18 | Hembra | Simmental | 2011, 05, 30 | 2200 | 340 | 440 | true | Pastura | 110 |
| 19 | Caravana19 | Macho | Angus | 2010, 03, 25 | 2300 | 360 | 460 | false | Grano | 115 |
| 20 | Caravana20 | Hembra | Hereford | 2009, 01, 15 | 2400 | 380 | 480 | true | Pastura | 120 |
| 21 | Caravana21 | Macho | Simmental | 2023, 01, 15 | 1500 | 200 | 300 | false | Grano | 125 |
| 22 | Caravana22 | Hembra | Angus | 2020, 03, 22 | 1600 | 220 | 320 | true | Pastura | 130 |
| 23 | Caravana23 | Macho | Hereford | 2021, 05, 10 | 1700 | 240 | 340 | false | Grano | 135 |
| 24 | Caravana24 | Hembra | Simmental | 2022, 07, 03 | 1800 | 260 | 360 | true | Pastura | 140 |
| 25 | Caravana25 | Macho | Angus | 2023, 09, 18 | 1900 | 280 | 380 | false | Grano | 145 |
| 26 | Caravana26 | Hembra | Hereford | 2024, 11, 25 | 2000 | 300 | 400 | true | Pastura | 150 |
| 27 | Caravana27 | Macho | Simmental | 2025, 12, 10 | 2100 | 320 | 420 | false | Grano | 155 |
| 28 | Caravana28 | Hembra | Angus | 2026, 10, 06 | 2200 | 340 | 440 | true | Pastura | 160 |
| 29 | Caravana29 | Macho | Hereford | 2027, 08, 30 | 2300 | 360 | 460 | false | Grano | 165 |
| 30 | Caravana30 | Hembra | Simmental | 2028, 07, 20 | 2400 | 380 | 480 | true | Pastura | 170 |

### Vacuna

|  |  |  |  |
| --- | --- | --- | --- |
| Nº | Nombre | Descripción | Patógeno |
| 1 | Vacuna Antitetánica | Protege contra el tétanos | Clostridium tetani |
| 2 | Vacuna Anticlostridial | Protege contra las infecciones por clostridios | Clostridium perfringens |
| 3 | Vacuna Anticarbuncloso | Protege contra el ántrax | Bacillus anthracis |
| 4 | Vacuna Antileptospira | Protege contra la leptospirosis | Leptospira spp |
| 5 | Vacuna Antibrucelosis | Protege contra la brucelosis | Brucella abortus |
| 6 | Vacuna Antipasteurelosis | Protege contra la pasteurelosis | Pasteurella multocida |
| 7 | Vacuna Antirabia | Protege contra la rabia | Virus de la rabia |
| 8 | Vacuna Antiviral | Protege contra enfermedades virales | Diferentes virus |
| 9 | Vacuna Antiparasitaria | Protege contra parásitos internos y externos | Diferentes parásitos |
| 10 | Vacuna Anticoccidial | Protege contra la coccidiosis | Diferentes especies de coccidios |

### Potrero

|  |  |  |  |
| --- | --- | --- | --- |
| Nº | Descripción | Hectareas | Capacidad Máxima |
| 1 | Potrero 1 | 20 | 50 |
| 2 | Potrero 2 | 15 | 40 |
| 3 | Potrero 3 | 25 | 60 |
| 4 | Potrero 4 | 18 | 45 |
| 5 | Potrero 5 | 22 | 55 |
| 6 | Potrero 6 | 17 | 42 |
| 7 | Potrero 7 | 21 | 58 |
| 8 | Potrero 8 | 19 | 48 |
| 9 | Potrero 9 | 23 | 63 |
| 10 | Potrero 10 | 16 | 38 |

### Registro Vacunación

|  |  |  |  |
| --- | --- | --- | --- |
| Animal | Tipo Vacuna | Fecha | Vencimiento |
| Ovino (Caravana 1) | Vacuna Antitetánica | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 1) | Vacuna Anticlostridial | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 1) | Vacuna Anticarbuncloso | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 5) | Vacuna Anticarbuncloso | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 5) | Vacuna Antileptospira | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 10) | Vacuna Antibrucelosis | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 10) | Vacuna Antipasteurelosis | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 15) | Vacuna Antirabia | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 15) | Vacuna Antiviral | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 20) | Vacuna Antiparasitaria | DateTime.Now | DateTime.Now.AddMonths(6) |
| Ovino (Caravana 20) | Vacuna Anticoccidial | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 1) | Vacuna Antitetánica | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 1) | Vacuna Anticlostridial | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 1) | Vacuna Anticarbuncloso | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 5) | Vacuna Anticarbuncloso | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 5) | Vacuna Antileptospira | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 10) | Vacuna Antibrucelosis | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 10) | Vacuna Antipasteurelosis | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 15) | Vacuna Antirabia | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 15) | Vacuna Antiviral | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 20) | Vacuna Antiparasitaria | DateTime.Now | DateTime.Now.AddMonths(6) |
| Bovino (Caravana 20) | Vacuna Anticoccidial | DateTime.Now | DateTime.Now.AddMonths(6) |

### Asignación de Animales al Potrero

|  |  |
| --- | --- |
| Potrero | Animal |
| Potrero 1 | Ovino (Caravana 1) |
| Potrero 1 | Ovino (Caravana 2) |
| Potrero 1 | Ovino (Caravana 3) |
| Potrero 2 | Ovino (Caravana 4) |
| Potrero 2 | Ovino (Caravana 5) |
| Potrero 2 | Ovino (Caravana 6) |
| Potrero 3 | Ovino (Caravana 7) |
| Potrero 3 | Ovino (Caravana 8) |
| Potrero 3 | Ovino (Caravana 9) |
| Potrero 4 | Ovino (Caravana 10) |
| Potrero 4 | Ovino (Caravana 11) |
| Potrero 4 | Ovino (Caravana 12) |
| Potrero 5 | Ovino (Caravana 13) |
| Potrero 5 | Ovino (Caravana 14) |
| Potrero 5 | Ovino (Caravana 15) |
| Potrero 6 | Ovino (Caravana 16) |
| Potrero 6 | Ovino (Caravana 17) |
| Potrero 6 | Ovino (Caravana 18) |
| Potrero 7 | Ovino (Caravana 19) |
| Potrero 7 | Ovino (Caravana 20) |
| Potrero 7 | Ovino (Caravana 21) |
| Potrero 8 | Ovino (Caravana 22) |
| Potrero 8 | Ovino (Caravana 23) |
| Potrero 8 | Ovino (Caravana 24) |
| Potrero 9 | Ovino (Caravana 25) |
| Potrero 9 | Ovino (Caravana 26) |
| Potrero 9 | Ovino (Caravana 27) |
| Potrero 10 | Ovino (Caravana 28) |
| Potrero 10 | Ovino (Caravana 29) |
| Potrero 10 | Ovino (Caravana 30) |
| Potrero 1 | Ovino (Caravana 1) |
| Potrero 1 | Ovino (Caravana 2) |
| Potrero 1 | Ovino (Caravana 3) |
| Potrero 2 | Ovino (Caravana 4) |
| Potrero 2 | Ovino (Caravana 5) |
| Potrero 2 | Ovino (Caravana 6) |
| Potrero 3 | Ovino (Caravana 7) |
| Potrero 3 | Ovino (Caravana 8) |
| Potrero 3 | Ovino (Caravana 9) |
| Potrero 4 | Ovino (Caravana 10) |
| Potrero 4 | Ovino (Caravana 11) |
| Potrero 4 | Ovino (Caravana 12) |
| Potrero 5 | Ovino (Caravana 13) |
| Potrero 5 | Ovino (Caravana 14) |
| Potrero 5 | Ovino (Caravana 15) |
| Potrero 6 | Ovino (Caravana 16) |
| Potrero 6 | Ovino (Caravana 17) |
| Potrero 6 | Ovino (Caravana 18) |
| Potrero 7 | Ovino (Caravana 19) |
| Potrero 7 | Ovino (Caravana 20) |
| Potrero 7 | Ovino (Caravana 21) |
| Potrero 8 | Ovino (Caravana 22) |
| Potrero 8 | Ovino (Caravana 23) |
| Potrero 8 | Ovino (Caravana 24) |
| Potrero 9 | Ovino (Caravana 25) |
| Potrero 9 | Ovino (Caravana 26) |
| Potrero 9 | Ovino (Caravana 27) |
| Potrero 10 | Ovino (Caravana 28) |
| Potrero 10 | Ovino (Caravana 29) |
| Potrero 10 | Ovino (Caravana 30) |

### Asignación de Tareas al Peón

|  |  |
| --- | --- |
| Peón | Tarea |
| Peon1 (id 1) | tarea1 (Preparar terreno para siembra) |
| Peon1 (id 1) | tarea1 (Sembrar cultivo de maíz) |
| Peon1 (id 2) | tarea1 (Regar cultivos) |
| Peon1 (id 2) | tarea1 (Fertilizar cultivos) |
| Peon1 (id 3) | tarea1 (Control de plagas) |
| Peon1 (id 3) | tarea1 (Podar árboles frutales) |
| Peon1 (id 4) | tarea1 (Cosechar cultivos) |
| Peon1 (id 4) | tarea1 (Inspección de cercas) |
| Peon1 (id 5) | tarea1 (Reparación de maquinaria agrícola) |
| Peon1 (id 5) | tarea1 (Control de malezas) |
| Peon1 (id 6) | tarea1 (Fertilizar terrenos vacíos) |
| Peon1 (id 6) | tarea1 (Revisión de sistemas de riego) |
| Peon1 (id 7) | tarea1 (Control de humedad en suelo) |
| Peon1 (id 7) | tarea1 (Cercar área de pastoreo) |
| Peon1 (id 8) | tarea1 (Preparar suelos para próximas siembras) |
| Peon1 (id 8) | tarea1 (Preparar terreno para siembra) |
| Peon1 (id 9) | tarea1 (Sembrar cultivo de maíz) |
| Peon1 (id 9) | tarea1 (Regar cultivos) |
| Peon1 (id 10) | tarea1 (Fertilizar cultivos) |
| Peon1 (id 10) | tarea1 (Control de plagas) |

# Código Fuente

## ConsoleApp\Program.cs

using ClassLibrary;

using ClassLibrary.Enum;

namespace ConsoleApp

{

public class Program

{

private static Sistema? sistema;

static void Main(string[] args)

{

sistema = Sistema.Instancia;

string? input;

bool codigo = true;

while (codigo)

{

Bienvenida();

Menu();

input = Console.ReadLine();

Console.WriteLine();

switch (input)

{

/\*\* Listado de todos los animales \*\*/

case "1":

ListarAnimales();

break;

/\*\* mostrar todos los potreros con área mayor a dicha cantidad de hectáreas y una capacidad máxima superior al número dado \*\*/

case "2":

Console.Clear();

int hectareas = InputNumber("Ingresar Cantidad de Hectareas");

int numero = InputNumber("Ingresar Capacidad Máxima");

sistema.ListarPotrerosHectareasCapacidadMaxima(hectareas, numero);

break;

/\*\* Establecer el precio por kilogramo de lana de los ovinos \*\*/

case "3":

Console.Clear();

int PrecioPorKiloLana = InputNumber("Ingrese Precio por kilogramo de Lana de los Ovinos:");

sistema.PrecioPorKiloLana(PrecioPorKiloLana);

break;

/\*\* Alta de ganado bovino. \*\*/

case "4":

Console.Clear();

Sistema.Resaltar("▀▄▀▄▀▄ ALTA DE GANADO BOVINO ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

string codigoCaravana = InputText("Ingrese Código de Caravana: ").Trim();

Bovino bovino = sistema.ObtenerBovinoPorCodigoCaravana(codigoCaravana);

if (bovino is null)

{

Sexo sexo = InputSexo();

string raza = InputText("Ingrese Raza: ");

DateTime fechaNacimiento = InputDateTime("Ingrese Fecha de nacimiento: ");

int costoAdquisicion = InputNumber("Ingrese Costo Adquisición: ");

int costoAlimentacion = InputNumber("Ingrese Costo Alimentación: ");

int pesoActual = InputNumber("Ingrese Peso Actual: ");

bool esHibrido = InputBool("¿Es Híbrido? ");

TipoAlimentacion tipoAlimentacion = InputTipoAlimentacion();

int precioPorKiloBovinoEnPie = InputNumber("Ingrese Precio Por Kilo en Pie: ");

sistema.AltaBovino(codigoCaravana, sexo, raza, fechaNacimiento, costoAdquisicion, costoAlimentacion, pesoActual, esHibrido, tipoAlimentacion, precioPorKiloBovinoEnPie);

}

else

{

Sistema.Error("Existe un Bovino con Código de Caravana Ingresado. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

break;

case "5":

/\*\* Costo de Crianza Animal \*\*/

Console.Clear();

Sistema.Resaltar("▀▄▀▄▀▄ COSTO DE CRIANZA POR ANIMAL ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

string codigoCaravana1 = InputText("Ingrese Código de Caravana: ").Trim();

TipoAnimal tipoAnimal = InputTipoAnimal();

if (tipoAnimal == TipoAnimal.Ovino)

{

Ovino ovino = sistema.ObtenerOvinoPorCodigoCaravana(codigoCaravana1);

if (ovino is not null)

{

decimal costoCrianzaAnimal = sistema.CostoCrianzaAnimal(ovino);

Sistema.Exito($"Costo de Crianza del {TipoAnimal.Ovino} con (Código de Caravana: {codigoCaravana1}) = ${costoCrianzaAnimal}. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

else

{

Sistema.Error("No Existe un Ovino con Código de Caravana Ingresado. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

}

else if (tipoAnimal == TipoAnimal.Bovino)

{

Bovino bovino1 = sistema.ObtenerBovinoPorCodigoCaravana(codigoCaravana1);

if (bovino1 is not null)

{

decimal costoCrianzaAnimal = sistema.CostoCrianzaAnimal(bovino1);

Sistema.Exito($"Costo de Crianza {TipoAnimal.Bovino} (Código de Caravana: {codigoCaravana1}) = {costoCrianzaAnimal}. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

else

{

Sistema.Error("No Existe un Bovino con Código de Caravana Ingresado. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

}

break;

case "6":

/\*\* Potencial precio de venta en Ovinos \*\*/

Console.Clear();

Sistema.Resaltar("▀▄▀▄▀▄ POTENCIAL PRECIO DE VENTA OVINOS ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

string codigoCaravana2 = InputText("Ingrese Código de Caravana: ").Trim();

Ovino ovino1 = sistema.ObtenerOvinoPorCodigoCaravana(codigoCaravana2);

if (ovino1 is not null)

{

decimal precioVentaOvino = sistema.PrecioVentaOvino(ovino1);

Sistema.Exito($"Potencial Precio de venta Ovino (Código de Caravana: {codigoCaravana2}) = ${precioVentaOvino}. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

else

{

Sistema.Error("No Existe un Ovino con Código de Caravana Ingresado. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

break;

case "7":

/\*\* Potencial precio de venta en Bovinos \*\*/

Console.Clear();

Sistema.Resaltar("▀▄▀▄▀▄ POTENCIAL PRECIO DE VENTA BOVINOS ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

string codigoCaravana3 = InputText("Ingrese Código de Caravana: ").Trim();

Bovino bovino2 = sistema.ObtenerBovinoPorCodigoCaravana(codigoCaravana3);

if (bovino2 is not null)

{

decimal precioVentaOvino = sistema.PrecioVentaBovino(bovino2);

Sistema.Exito($"Potencial Precio de venta Bovino (Código de Caravana: {codigoCaravana3}) = ${precioVentaOvino}. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

else

{

Sistema.Error("No Existe un Bovino con Código de Caravana Ingresado. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

break;

case "8":

/\*\* Ganancias estimadas de la venta de un potrero \*\*/

Console.Clear();

Sistema.Resaltar("\n ▀▄▀▄▀▄ GANANCIAS ESTIMADAS DE VENTA POR POTRERO ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

CantidadAnimalesPorPotrero();

int id = InputNumber("Ingrese ID del Potrero:");

Potrero potrero = sistema.ObtenerPotreroPorId(id);

if (potrero is not null)

{

decimal ganaciaVentaPotrero = sistema.GanaciaVentaPotrero(potrero);

Sistema.Exito($"Ganancias Estimadas del Potrero con ID {potrero.Id}: ${ganaciaVentaPotrero}. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

else

{

Sistema.Error("No Existe un Potrero con ID Ingresado. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

break;

case "9":

ListarBovinos();

break;

case "10":

ListarOvinos();

break;

case "11":

ListarPotreros();

break;

case "12":

ListarAnimalesPorPotrero();

break;

case "13":

ListarVacunas();

break;

case "14":

ListarVacunasPorAnimal();

break;

case "15":

ListarCapataces();

break;

case "16":

ListarPeones();

break;

case "17":

ListarTareas();

break;

case "18":

ListarTareasPorPeon();

break;

case "0":

Sistema.Exito("Cerrando Aplicación de Consola...".ToUpper());

codigo = false;

break;

default:

Sistema.Error("Opción Inválida. Presione una Tecla Para Continuar. \n".ToUpper());

Console.ReadKey();

break;

}

}

}

#region Get; Set;

/\*\* Get; Set; \*\*/

#endregion Get; Set;

#region Métodos que Listan Información

/\*\* Métodos para Listar Información \*\*/

static void CantidadAnimalesPorPotrero()

{

try

{

foreach (Potrero potrero in sistema.Potreros)

{

Sistema.Resaltar($"▀▄▀▄▀▄ POTRERO ID: {potrero.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Sistema.Resaltar($"Total de Animales: {potrero.Animales.Count}. Capacidad Máxima: {potrero.CapacidadMaxima} \n", ConsoleColor.DarkGray);

}

}

catch (Exception ex)

{

Console.WriteLine();

Sistema.Error($"{ex.Message} \n");

}

}

static void ListarVacunasPorAnimal()

{

try

{

Console.Clear();

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE TAREAS POR PEÓN ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

int contador = 1;

foreach (Animal animal in sistema.Animales)

{

if (animal.Vacunaciones.Count > 0)

{

if (animal is Ovino)

{

Ovino ovino = (Ovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ OVINO {ovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

foreach (Vacunacion vacunacion in ovino.Vacunaciones)

{

Console.WriteLine($"➜ {vacunacion} \n");

}

}

else if (animal is Bovino)

{

Bovino bovino = (Bovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ BOVINO {bovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

foreach (Vacunacion vacunacion in bovino.Vacunaciones)

{

Console.WriteLine($"➜ {vacunacion} \n");

}

}

}

}

}

catch (Exception ex)

{

Console.WriteLine();

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Vacunas por Animal Listadas con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarTareasPorPeon()

{

try

{

Console.Clear();

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE TAREAS POR PEÓN ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

int contador = 1;

foreach (Empleado empleado in sistema.Empleados)

{

if (empleado is Peon)

{

Peon peon = (Peon)empleado;

Sistema.Resaltar($"▀▄▀▄▀▄ PEÓN {peon.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

foreach (Tarea tarea in peon.TareasAsignadas)

{

Console.WriteLine($"➜ {tarea} \n");

}

}

}

}

catch (Exception ex)

{

Console.WriteLine();

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Tareas por Peón Listadas con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarAnimalesPorPotrero()

{

try

{

Console.Clear();

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE ANIMALES POR POTRERO ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

int contador = 1;

foreach (Potrero potrero in sistema.Potreros)

{

Sistema.Resaltar($"▀▄▀▄▀▄ POTRERO {potrero.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

foreach (Animal animal in potrero.Animales)

{

if (animal is Ovino)

{

Ovino ovino = (Ovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ OVINO {ovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkGray);

Console.WriteLine($"➜ {animal} \n");

}

else if (animal is Bovino)

{

Bovino bovino = (Bovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ BOVINO {bovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkGray);

Console.WriteLine($"➜ {animal} \n");

}

}

}

}

catch (Exception ex)

{

Console.WriteLine();

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Animales por Potrero Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarPotreros()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE POTREROS ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

int contador = 1;

foreach (Potrero potrero in sistema.Potreros)

{

Sistema.Resaltar($"▀▄▀▄▀▄ POTRERO {potrero.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {potrero} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Potreros Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarVacunas()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE VACUNAS ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

int contador = 1;

foreach (Vacuna vacuna in sistema.Vacunas)

{

Sistema.Resaltar($"▀▄▀▄▀▄ VACUNA {vacuna.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {vacuna} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Vacunas Listadas con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarAnimales()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE ANIMALES ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

if (sistema.Animales.Count == 0) throw new ArgumentOutOfRangeException("Lista de Animales Vacía. Sistema\\ListarAnimales() \n");

int contador = 1;

foreach (Animal animal in sistema.Animales)

{

if (animal is Ovino)

{

Ovino ovino = (Ovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ OVINO {ovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {ovino} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

else if (animal is Bovino)

{

Bovino bovino = (Bovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ BOVINO {bovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {bovino} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Animales Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarOvinos()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE OVINOS ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

if (sistema.Animales.Count == 0) throw new ArgumentOutOfRangeException("Lista de Animales Vacía. Sistema\\ListarOvinos() \n");

int contador = 1;

foreach (Animal animal in sistema.Animales)

{

if (animal is Ovino)

{

Ovino ovino = (Ovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ OVINO {ovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {ovino} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Ovinos Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarBovinos()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE BOVINOS ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

if (sistema.Animales.Count == 0) throw new ArgumentOutOfRangeException("Lista de Animales Vacía. Sistema\\ListarBovinos() \n");

int contador = 1;

foreach (Animal animal in sistema.Animales)

{

if (animal is Bovino)

{

Bovino bovino = (Bovino)animal;

Sistema.Resaltar($"▀▄▀▄▀▄ BOVINO {bovino.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {bovino} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Bovinos Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarTareas()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE TAREAS ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

if (sistema.Tareas.Count == 0) throw new ArgumentOutOfRangeException("Lista de Tareas Vacía. Sistema\\ListarTareas() \n");

int contador = 1;

foreach (Tarea tarea in sistema.Tareas)

{

Sistema.Resaltar($"▀▄▀▄▀▄ TAREA {tarea.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {tarea} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Tareas Listadas con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarCapataces()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE CAPATACES ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

if (sistema.Empleados.Count == 0) throw new ArgumentOutOfRangeException("Lista de Empleados Vacía. Sistema\\ListarCapataces() \n");

int contador = 1;

foreach (Empleado empleado in sistema.Empleados)

{

if (empleado is Capataz)

{

Capataz capataz = (Capataz)empleado;

Sistema.Resaltar($"▀▄▀▄▀▄ CAPATAZ {capataz.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {capataz} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("Capataces Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

static void ListarPeones()

{

Console.Clear();

try

{

Sistema.Resaltar("▀▄▀▄▀▄ LISTADO DE PEONES ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

if (sistema.Empleados.Count == 0) throw new ArgumentOutOfRangeException("Lista de Empleados Vacía. Sistema\\ListarPeones() \n");

int contador = 1;

foreach (Empleado empleado in sistema.Empleados)

{

if (empleado is Peon)

{

Peon peon = (Peon)empleado;

Sistema.Resaltar($"▀▄▀▄▀▄ PEÓN {peon.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkBlue);

Console.WriteLine($"◢◤◢◤◢◤◢◤◢◤◢ ({contador++}) {peon} ◢◤◢◤◢◤◢◤◢◤◢ \n");

}

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

Sistema.Exito("peones Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

#endregion Métodos que Listan Información

#region Métodos Globales

/\*\* Métodos Globales \*\*/

static void SaltoDeLinea()

{

Console.WriteLine();

return;

}

static bool InputBool(string mensaje)

{

bool exito = false;

bool input = false;

while (!exito)

{

try

{

Sistema.Resaltar($"{mensaje} (S/N) \n", ConsoleColor.DarkBlue);

string inputString = Console.ReadLine().ToUpper();

if (inputString.Length != 1) throw new ArgumentOutOfRangeException("Cantidad de Dígitos Superior a la Esperada. Ingrese S o N. Program\\(string mensaje)");

switch (inputString)

{

case "S":

input = true;

exito = true;

break;

case "N":

input = false;

exito = true;

break;

default:

throw new ArgumentException("Opción Inválida. Program\\(string mensaje) \n");

}

}

catch (Exception ex)

{

SaltoDeLinea();

Sistema.Error($"{ex.Message} \n");

}

}

SaltoDeLinea();

return input;

}

static DateTime InputDateTime(string mensaje)

{

bool exito = false;

DateTime dateTime = DateTime.Now;

while (!exito)

{

try

{

Sistema.Resaltar($"{mensaje} (Day/Month/Year): \n", ConsoleColor.DarkBlue);

exito = DateTime.TryParse(Console.ReadLine(), out dateTime);

if (!exito) throw new ArgumentException("Formato de Fecha Incorrecto. Program\\InputDateTime(string mensaje)");

}

catch (Exception ex)

{

SaltoDeLinea();

Sistema.Error($"{ex.Message} \n");

}

}

SaltoDeLinea();

return dateTime;

}

static TipoAnimal InputTipoAnimal()

{

bool exito = false;

TipoAnimal tipoAnimal = new TipoAnimal();

while (!exito)

{

try

{

Sistema.Resaltar("Seleccione Tipo de Animal: \n", ConsoleColor.DarkBlue);

foreach (int numero in Enum.GetValues(typeof(TipoAnimal)))

{

Sistema.Resaltar($"{numero} ➟ {(TipoAnimal)numero} \n", ConsoleColor.DarkBlue);

}

string inputString = Console.ReadLine();

if (string.IsNullOrEmpty(inputString)) throw new ArgumentException("String Vacío. Program\\TipoAnimal()");

bool isNumber = int.TryParse(inputString, out int number);

if (isNumber && Enum.IsDefined(typeof(TipoAnimal), number))

{

tipoAnimal = (TipoAnimal)number;

exito = true;

}

else

{

throw new ArgumentOutOfRangeException("Opción Inválida. Program\\TipoAnimal()");

}

}

catch (Exception ex)

{

SaltoDeLinea();

Sistema.Error($"{ex.Message} \n");

}

}

SaltoDeLinea();

return tipoAnimal;

}

static TipoAlimentacion InputTipoAlimentacion()

{

bool exito = false;

TipoAlimentacion tipoAlimentacion = new TipoAlimentacion();

while (!exito)

{

try

{

Sistema.Resaltar("Seleccione Tipo de Alimentación: \n", ConsoleColor.DarkBlue);

foreach (int numero in Enum.GetValues(typeof(TipoAlimentacion)))

{

Sistema.Resaltar($"{numero} ➟ {(TipoAlimentacion)numero} \n", ConsoleColor.DarkBlue);

}

string inputString = Console.ReadLine();

if (string.IsNullOrEmpty(inputString)) throw new ArgumentException("String Vacío. Program\\InputTipoAlimentacion()");

bool isNumber = int.TryParse(inputString, out int number);

if (isNumber && Enum.IsDefined(typeof(TipoAlimentacion), number))

{

tipoAlimentacion = (TipoAlimentacion)number;

exito = true;

}

else

{

throw new ArgumentOutOfRangeException("Opción Inválida. Program\\InputTipoAlimentacion()");

}

}

catch (Exception ex)

{

SaltoDeLinea();

Sistema.Error($"{ex.Message} \n");

}

}

SaltoDeLinea();

return tipoAlimentacion;

}

static Sexo InputSexo()

{

bool exito = false;

Sexo sexo = new Sexo();

while (!exito)

{

try

{

Sistema.Resaltar("Seleccione Sexo del Animal: \n", ConsoleColor.DarkBlue);

foreach (int numero in Enum.GetValues(typeof(Sexo)))

{

Sistema.Resaltar($"{numero} ➟ {(Sexo)numero} \n", ConsoleColor.DarkBlue);

}

string inputString = Console.ReadLine();

if (string.IsNullOrEmpty(inputString)) throw new ArgumentException("String Vacío. Program\\InputSexo()");

bool isNumber = int.TryParse(inputString, out int number);

if (isNumber && Enum.IsDefined(typeof(Sexo), number))

{

sexo = (Sexo)number;

exito = true;

}

else

{

SaltoDeLinea();

throw new ArgumentOutOfRangeException("Opción Inválida. Program\\InputSexo(string mensaje)");

}

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

}

SaltoDeLinea();

return sexo;

}

static int InputNumber(string mensaje)

{

bool exito = false;

int inputNumero = 0;

bool isNumber = false;

while (!exito)

{

try

{

Sistema.Resaltar($"{mensaje} \n", ConsoleColor.DarkBlue);

isNumber = int.TryParse(Console.ReadLine(), out inputNumero);

if (!isNumber || inputNumero <= 0) throw new ArgumentOutOfRangeException("Número Incorrecto. Program\\InputText(string mensaje)");

exito = true;

}

catch (Exception ex)

{

SaltoDeLinea();

Sistema.Error($"{ex.Message} \n");

}

}

SaltoDeLinea();

return inputNumero;

}

static string InputText(string mensaje)

{

bool exito = false;

string? inputText = string.Empty;

while (!exito)

{

try

{

Sistema.Resaltar($"{mensaje} \n", ConsoleColor.DarkBlue);

inputText = Console.ReadLine();

if (string.IsNullOrEmpty(inputText)) throw new ArgumentException("InputText Vacío. InputString(string mensaje)");

exito = true;

}

catch (Exception ex)

{

Sistema.Error($"{ex.Message} \n");

}

}

SaltoDeLinea();

return inputText;

}

static void Menu()

{

Sistema.Resaltar("➡︎ 1 ➔ Listado de Todos los Animales \n".ToUpper(), ConsoleColor.DarkCyan);

Sistema.Resaltar("➡︎ 2 ➔ Listado de Potreros con Área Mayor a Cantidad de Hectáreas Proporcionada y Capacidad Máxima Superior al Número Dado \n".ToUpper(), ConsoleColor.DarkCyan);

Sistema.Resaltar("➡︎ 3 ➔ Establecer el Precio por Kilogramo de Lana de los Ovinos \n".ToUpper(), ConsoleColor.DarkCyan);

Sistema.Resaltar("➡︎ 4 ➔ Alta de Ganado Bovino \n".ToUpper(), ConsoleColor.DarkCyan);

Sistema.Resaltar("➡︎ 5 ➔ Costo de Crianza por Animal \n".ToUpper(), ConsoleColor.DarkCyan);

Sistema.Resaltar("➡︎ 6 ➔ Potencial Precio de Venta Ovino \n".ToUpper(), ConsoleColor.DarkCyan);

Sistema.Resaltar("➡︎ 7 ➔ Potencial Precio de Venta Bovino \n".ToUpper(), ConsoleColor.DarkCyan);

Sistema.Resaltar("➡︎ 8 ➔ Ganancias Estimadas de Venta por Potrero \n".ToUpper(), ConsoleColor.DarkCyan);

Console.WriteLine("➡︎ 9 ➔ Listar Ganado Bovino \n".ToUpper());

Console.WriteLine("➡︎ 10 ➔ Listar Ganado Ovino \n".ToUpper());

Console.WriteLine("➡︎ 11 ➔ Listar Potreros \n".ToUpper());

Console.WriteLine("➡︎ 12 ➔ Listar Animales por Potrero \n".ToUpper());

Console.WriteLine("➡︎ 13 ➔ Listar Vacunas \n".ToUpper());

Console.WriteLine("➡︎ 14 ➔ Listar Vacunas por Animal \n".ToUpper());

Console.WriteLine("➡︎ 15 ➔ Listar Capataces \n".ToUpper());

Console.WriteLine("➡︎ 16 ➔ Listar Peones \n".ToUpper());

Console.WriteLine("➡︎ 17 ➔ Listar Tareas \n".ToUpper());

Console.WriteLine("➡︎ 18 ➔ Listar Tareas Por Peón \n".ToUpper());

Console.WriteLine("➡︎ 0 ➔ Salir \n".ToUpper());

}

static void Bienvenida()

{

Console.Clear();

SaltoDeLinea();

Sistema.Resaltar("🐄 🐑 ▁ ▂ ▄ ▅ ▆ ▇ █ ESTANCIA █ ▇ ▆ ▅ ▄ ▂ ▁ 🐑 🐄", ConsoleColor.DarkGreen);

Sistema.Resaltar("░▒▓█ Compra y Engorde de Bovinos y Ovinos █▓▒░".ToUpper(), ConsoleColor.DarkGreen);

Console.WriteLine();

Sistema.Resaltar("◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠ MENÚ ◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠◡◠", ConsoleColor.DarkYellow);

Console.WriteLine();

}

#endregion Métodos Globales

}

}

## ClassLibrary\Sistema.cs

using ClassLibrary.Enum;

namespace ClassLibrary

{

public class Sistema

{

/\*\* Atributos \*\*/

private List<Empleado> \_empleados = new List<Empleado>();

private List<Animal> \_animales = new List<Animal>();

private List<Tarea> \_tareas = new List<Tarea>();

private List<Vacuna> \_vacunas = new List<Vacuna>();

private List<Potrero> \_potreros = new List<Potrero>();

/\*\* Singleton \*\*/

private static Sistema? \_instancia;

/\*\* Constructor \*\*/

private Sistema()

{

PrecargarEmpleado();

PrecargarAnimal();

PrecargarTarea();

PrecargarVacuna();

PrecargarPotrero();

VacunarBovino();

VacunarOvino();

AsignarAnimalAlPotrero();

AsignarTareaAlPeon();

}

#region Get; Set;

/\*\* Get; Set; \*\*/

public static Sistema Instancia

{

get { if (\_instancia == null) \_instancia = new Sistema(); return \_instancia; }

}

public List<Empleado> Empleados

{

get { return \_empleados; }

}

public List<Animal> Animales

{

get { return \_animales; }

}

public List<Tarea> Tareas

{

get { return \_tareas; }

}

public List<Vacuna> Vacunas

{

get { return \_vacunas; }

}

public List<Potrero> Potreros

{

get { return \_potreros; }

}

#endregion Get; Set;

#region Métodos paraCalcular

/\*\* Métodos paraCalcular \*\*/

public decimal GanaciaVentaPotrero(Potrero potrero)

{

decimal costoTotal = 0;

decimal precioVentaAnimal = 0;

decimal costoCrianza = 0;

try

{

if (potrero is null) throw new ArgumentNullException("Object Null. GanaciaVentaPotrero(Potrero potrero)");

foreach(Animal animal in potrero.Animales)

{

if (animal is Ovino)

{

Ovino ovino = (Ovino)animal;

precioVentaAnimal += PrecioVentaOvino(ovino);

costoCrianza += CostoCrianzaAnimal(ovino);

}

else if (animal is Bovino)

{

Bovino bovino = (Bovino)animal;

precioVentaAnimal += PrecioVentaBovino(bovino);

costoCrianza += CostoCrianzaAnimal(bovino);

}

}

costoTotal = precioVentaAnimal - costoCrianza;

}

catch (Exception ex)

{

Console.WriteLine();

Error($"{ex.Message} \n");

}

return costoTotal;

}

public decimal PrecioVentaBovino(Bovino bovino)

{

decimal costoTotal = 0;

try

{

if (bovino is null) throw new ArgumentNullException("Object Null. Sistema.cs\\PrecioVentaOvino(Ovino ovino)");

decimal precioVenta = ((decimal)bovino.PesoActual \* bovino.PrecioPorKiloBovinoEnPie);

decimal recargoTipoAlimentacion = (bovino.TipoAlimentacion == TipoAlimentacion.Grano) ? precioVenta \* 0.30m : 0;

decimal recargoHembra = (bovino.Sexo == Sexo.Hembra) ? precioVenta \* 0.10m : 0;

costoTotal = precioVenta + recargoTipoAlimentacion + recargoHembra;

}

catch (Exception ex)

{

Console.WriteLine();

Error($"{ex.Message} \n");

}

return costoTotal;

}

public decimal PrecioVentaOvino(Ovino ovino)

{

decimal costoTotal = 0;

try

{

if (ovino is null) throw new ArgumentNullException("Object Null. Sistema.cs\\PrecioVentaOvino(Ovino ovino)");

decimal precioVenta = ((decimal)ovino.PesoLanaEstimado \* ovino.PrecioPorKiloLana) + (ovino.PrecioPorKiloOvinoEnPie \* (decimal)ovino.PesoActual);

decimal descuento = (ovino.EsHibrido) ? precioVenta \* 0.05m : 0;

costoTotal = precioVenta - descuento;

}

catch (Exception ex)

{

Console.WriteLine();

Error($"{ex.Message} \n");

}

return costoTotal;

}

public decimal CostoCrianzaAnimal(Animal? animal)

{

decimal costoTotal = 0;

try

{

if (animal is null) throw new ArgumentNullException("Object Null. Sistema.cs\\CostoCrianzaAnimal(Animal animal)");

decimal costoCrianzaAnimal = animal.CostoAdquisicion + animal.CostoAlimentacion;

int cantidadVacunas = animal.Vacunaciones.Count;

decimal costoVacunas = cantidadVacunas \* 200;

costoTotal = costoCrianzaAnimal + costoVacunas;

}

catch (Exception ex)

{

Console.WriteLine();

Error($"{ex.Message} \n");

}

return costoTotal;

}

#endregion Métodos paraCalcular

#region Métodos para Buscar Información

/\*\* Métodos para Buscar Información \*\*/

public void ListarPotrerosHectareasCapacidadMaxima(double hectareas, int numero)

{

if (hectareas <= 0 || numero <= 0) throw new ArgumentOutOfRangeException("Parámetros incorrectos. ListarPotrerosHectareasCapacidadMaxima(double hectareas, int numero) \n");

int contador = 1;

List<Potrero> potreros = new List<Potrero>();

foreach (Potrero potrero in \_potreros)

{

if (potrero.Hectareas > hectareas && potrero.CapacidadMaxima > numero)

{

potreros.Add(potrero);

}

}

if (potreros.Count == 0)

{

Error("No Se Encontraron Registros. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

return;

}

foreach (Potrero potrero in potreros)

{

Sistema.Resaltar($"▀▄▀▄▀▄ POTRERO {potrero.Id} ▄▀▄▀▄▀ \n", ConsoleColor.DarkYellow);

Console.WriteLine($"({contador++}) {potrero} ");

}

Exito("Potreros Listados con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

public Potrero ObtenerPotreroPorId(int id)

{

Potrero potrero = null;

try

{

if (id <= 0) throw new ArgumentOutOfRangeException("ID 0. ObtenerPotreroPorId(int id)");

int index = 0;

while (index < \_potreros.Count && potrero is null)

{

if (\_potreros[index].Id == id)

{

potrero = \_potreros[index];

}

index++;

}

}

catch (Exception ex)

{

Console.WriteLine();

Error($"{ex.Message} \n");

}

return potrero;

}

public Vacuna ObtenerVacunaPorNombre(string nombre)

{

if (nombre is null) throw new ArgumentNullException("String Vacío. ObtenerVacunaPorNombre(string nombre)");

Vacuna vacuna = null;

int index = 0;

while (index < \_vacunas.Count && vacuna is null)

{

if (\_vacunas[index].Nombre == nombre) vacuna = \_vacunas[index];

index++;

}

return vacuna;

}

public Ovino ObtenerOvinoPorCodigoCaravana(string codigoCaravana)

{

if (string.IsNullOrEmpty(codigoCaravana)) throw new ArgumentNullException("String Vacío. ObtenerOvinoPorCodigoCaravana(string codigoCaravana)");

Ovino ovino = null;

for (int index = 0; index < \_animales.Count; index++)

{

Animal animal = \_animales[index];

if (animal.CodigoCaravana == codigoCaravana && animal is Ovino)

{

ovino = (Ovino)animal;

}

}

return ovino;

}

public Bovino ObtenerBovinoPorCodigoCaravana(string codigoCaravana)

{

Bovino bovino = null;

try

{

if (string.IsNullOrEmpty(codigoCaravana)) throw new ArgumentNullException("String Vacío. ObtenerBovinoPorCodigoCaravana(string codigoCaravana)");

for (int index = 0; index < \_animales.Count; index++)

{

Animal animal = \_animales[index];

if (animal.CodigoCaravana == codigoCaravana && animal is Bovino)

{

bovino = (Bovino)animal;

}

}

}

catch (Exception ex)

{

Console.WriteLine();

Error(ex.Message);

}

return bovino;

}

public Tarea ObtenerTareaPorId(int id)

{

if (id <= 0) throw new ArgumentException("ID 0. ObtenerTareaPorId(int id)");

Tarea tarea = null;

int index = 0;

while (index < \_tareas.Count && tarea is null)

{

if (\_tareas[index].Id == id) tarea = \_tareas[index];

index++;

}

return tarea;

}

public Peon ObtenerPeonPorId(int id)

{

if (id <= 0) throw new ArgumentException("ID 0. ObtenerPeonPorId(int id)");

Peon peon = null;

int index = 0;

while (index < \_empleados.Count && peon is null)

{

Empleado empleado = \_empleados[index];

if (empleado is Peon)

{

Peon pawn = (Peon)empleado;

if (pawn.Id == id)

{

peon = pawn;

}

}

index++;

}

return peon;

}

#endregion Métodos para Buscar Información

#region Métodos para Agregar o Modificar Información

/\*\* Métodos para Agregan o Modificar Información \*\*/

public void AsignarTareaAlPeon()

{

try

{

Peon peon1 = ObtenerPeonPorId(1);

Peon peon2 = ObtenerPeonPorId(2);

Peon peon3 = ObtenerPeonPorId(3);

Peon peon4 = ObtenerPeonPorId(4);

Peon peon5 = ObtenerPeonPorId(5);

Peon peon6 = ObtenerPeonPorId(6);

Peon peon7 = ObtenerPeonPorId(7);

Peon peon8 = ObtenerPeonPorId(8);

Peon peon9 = ObtenerPeonPorId(9);

Peon peon10 = ObtenerPeonPorId(10);

if (peon1 is null || peon2 is null || peon3 is null || peon4 is null || peon5 is null || peon6 is null || peon7 is null || peon8 is null || peon9 is null || peon10 is null)

{

throw new InvalidOperationException("Object Null. Sistema.cs\\AsignarTareaAlPeon()");

}

Tarea tarea1 = ObtenerTareaPorId(1);

Tarea tarea2 = ObtenerTareaPorId(2);

Tarea tarea3 = ObtenerTareaPorId(3);

Tarea tarea4 = ObtenerTareaPorId(4);

Tarea tarea5 = ObtenerTareaPorId(5);

Tarea tarea6 = ObtenerTareaPorId(6);

Tarea tarea7 = ObtenerTareaPorId(7);

Tarea tarea8 = ObtenerTareaPorId(8);

Tarea tarea9 = ObtenerTareaPorId(9);

Tarea tarea10 = ObtenerTareaPorId(10);

Tarea tarea11 = ObtenerTareaPorId(11);

Tarea tarea12 = ObtenerTareaPorId(12);

Tarea tarea13 = ObtenerTareaPorId(13);

Tarea tarea14 = ObtenerTareaPorId(14);

Tarea tarea15 = ObtenerTareaPorId(15);

if (tarea1 is null || tarea2 is null || tarea3 is null || tarea4 is null || tarea5 is null || tarea6 is null || tarea7 is null || tarea8 is null || tarea9 is null || tarea10 is null || tarea11 is null || tarea12 is null || tarea13 is null || tarea14 is null || tarea15 is null)

{

throw new InvalidOperationException("Object Null. Sistema.cs\\AsignarTareaAlPeon()");

}

peon1.AsignarTarea(tarea1);

peon1.AsignarTarea(tarea2);

peon2.AsignarTarea(tarea3);

peon2.AsignarTarea(tarea4);

peon3.AsignarTarea(tarea5);

peon3.AsignarTarea(tarea6);

peon4.AsignarTarea(tarea7);

peon4.AsignarTarea(tarea8);

peon5.AsignarTarea(tarea9);

peon5.AsignarTarea(tarea10);

peon6.AsignarTarea(tarea11);

peon6.AsignarTarea(tarea12);

peon7.AsignarTarea(tarea13);

peon7.AsignarTarea(tarea14);

peon8.AsignarTarea(tarea15);

peon8.AsignarTarea(tarea1);

peon9.AsignarTarea(tarea2);

peon9.AsignarTarea(tarea3);

peon10.AsignarTarea(tarea4);

peon10.AsignarTarea(tarea5);

}

catch (Exception ex)

{

Console.WriteLine();

Error($"{ex.Message} \n");

return;

}

}

public void AsignarAnimalAlPotrero()

{

try

{

Potrero potrero1 = ObtenerPotreroPorId(1);

Potrero potrero2 = ObtenerPotreroPorId(2);

Potrero potrero3 = ObtenerPotreroPorId(3);

Potrero potrero4 = ObtenerPotreroPorId(4);

Potrero potrero5 = ObtenerPotreroPorId(5);

Potrero potrero6 = ObtenerPotreroPorId(6);

Potrero potrero7 = ObtenerPotreroPorId(7);

Potrero potrero8 = ObtenerPotreroPorId(8);

Potrero potrero9 = ObtenerPotreroPorId(9);

Potrero potrero10 = ObtenerPotreroPorId(10);

if (potrero1 is null || potrero2 is null || potrero3 is null || potrero4 is null || potrero5 is null || potrero6 is null || potrero7 is null || potrero8 is null || potrero9 is null || potrero10 is null)

{

throw new ArgumentException("Object Null. Sistema.cs\\AsignarAnimalAlPotrero()");

}

List<Ovino> ovinos = new List<Ovino>();

List<Bovino> bovinos = new List<Bovino>();

for (int index = 0; index < \_animales.Count; index++)

{

Animal animal = \_animales[index];

if (animal is Ovino)

{

Ovino ovino = (Ovino)animal;

ovinos.Add(ovino);

}

else if (animal is Bovino)

{

Bovino bovino = (Bovino)animal;

bovinos.Add(bovino);

}

}

/\*\* Asignar Potrero Ovino \*\*/

foreach (Ovino ovino in ovinos)

{

if (ovino.CodigoCaravana == "Caravana1" || ovino.CodigoCaravana == "Caravana2" || ovino.CodigoCaravana == "Caravana3")

{

potrero1.AsignarPotrero(ovino, potrero1);

}

else if (ovino.CodigoCaravana == "Caravana4" || ovino.CodigoCaravana == "Caravana5" || ovino.CodigoCaravana == "Caravana6")

{

potrero2.AsignarPotrero(ovino, potrero2);

}

else if (ovino.CodigoCaravana == "Caravana7" || ovino.CodigoCaravana == "Caravana8" || ovino.CodigoCaravana == "Caravana9")

{

potrero3.AsignarPotrero(ovino, potrero3);

}

else if (ovino.CodigoCaravana == "Caravana10" || ovino.CodigoCaravana == "Caravana11" || ovino.CodigoCaravana == "Caravana12")

{

potrero4.AsignarPotrero(ovino, potrero4);

}

else if (ovino.CodigoCaravana == "Caravana13" || ovino.CodigoCaravana == "Caravana14" || ovino.CodigoCaravana == "Caravana15")

{

potrero5.AsignarPotrero(ovino, potrero5);

}

else if (ovino.CodigoCaravana == "Caravana16" || ovino.CodigoCaravana == "Caravana17" || ovino.CodigoCaravana == "Caravana18")

{

potrero6.AsignarPotrero(ovino, potrero6);

}

else if (ovino.CodigoCaravana == "Caravana19" || ovino.CodigoCaravana == "Caravana20" || ovino.CodigoCaravana == "Caravana21")

{

potrero7.AsignarPotrero(ovino, potrero7);

}

else if (ovino.CodigoCaravana == "Caravana22" || ovino.CodigoCaravana == "Caravana23" || ovino.CodigoCaravana == "Caravana24")

{

potrero8.AsignarPotrero(ovino, potrero8);

}

else if (ovino.CodigoCaravana == "Caravana25" || ovino.CodigoCaravana == "Caravana26" || ovino.CodigoCaravana == "Caravana27")

{

potrero9.AsignarPotrero(ovino, potrero9);

}

else if (ovino.CodigoCaravana == "Caravana28" || ovino.CodigoCaravana == "Caravana29" || ovino.CodigoCaravana == "Caravana30")

{

potrero10.AsignarPotrero(ovino, potrero10);

}

else

{

throw new ArgumentException("Error Al Asignar Ovino al Potrero");

}

}

/\*\* Asignar Potrero Bovino \*\*/

foreach (Bovino bovino in bovinos)

{

if (bovino.CodigoCaravana == "Caravana1" || bovino.CodigoCaravana == "Caravana2" || bovino.CodigoCaravana == "Caravana3")

{

potrero1.AsignarPotrero(bovino, potrero1);

}

else if (bovino.CodigoCaravana == "Caravana4" || bovino.CodigoCaravana == "Caravana5" || bovino.CodigoCaravana == "Caravana6")

{

potrero2.AsignarPotrero(bovino, potrero2);

}

else if (bovino.CodigoCaravana == "Caravana7" || bovino.CodigoCaravana == "Caravana8" || bovino.CodigoCaravana == "Caravana9")

{

potrero3.AsignarPotrero(bovino, potrero3);

}

else if (bovino.CodigoCaravana == "Caravana10" || bovino.CodigoCaravana == "Caravana11" || bovino.CodigoCaravana == "Caravana12")

{

potrero4.AsignarPotrero(bovino, potrero4);

}

else if (bovino.CodigoCaravana == "Caravana13" || bovino.CodigoCaravana == "Caravana14" || bovino.CodigoCaravana == "Caravana15")

{

potrero5.AsignarPotrero(bovino, potrero5);

}

else if (bovino.CodigoCaravana == "Caravana16" || bovino.CodigoCaravana == "Caravana17" || bovino.CodigoCaravana == "Caravana18")

{

potrero6.AsignarPotrero(bovino, potrero6);

}

else if (bovino.CodigoCaravana == "Caravana19" || bovino.CodigoCaravana == "Caravana20" || bovino.CodigoCaravana == "Caravana21")

{

potrero7.AsignarPotrero(bovino, potrero7);

}

else if (bovino.CodigoCaravana == "Caravana22" || bovino.CodigoCaravana == "Caravana23" || bovino.CodigoCaravana == "Caravana24")

{

potrero8.AsignarPotrero(bovino, potrero8);

}

else if (bovino.CodigoCaravana == "Caravana25" || bovino.CodigoCaravana == "Caravana26" || bovino.CodigoCaravana == "Caravana27")

{

potrero9.AsignarPotrero(bovino, potrero9);

}

else if (bovino.CodigoCaravana == "Caravana28" || bovino.CodigoCaravana == "Caravana29" || bovino.CodigoCaravana == "Caravana30")

{

potrero10.AsignarPotrero(bovino, potrero10);

}

else

{

throw new ArgumentException("Error Al Asignar Bovino al Potrero");

}

}

}

catch (Exception ex)

{

Console.WriteLine();

Error($"{ex.Message} \n");

return;

}

}

public void AltaBovino(string codigoCaravana, Sexo sexo, string raza, DateTime fechaNacimiento, decimal costoAdquisicion, decimal costoAlimentacion, double pesoActual, bool esHibrido, TipoAlimentacion tipoAlimentacion, decimal precioPorKiloBovinoEnPie)

{

try

{

Bovino bovino = new Bovino(codigoCaravana, sexo, raza, fechaNacimiento, costoAdquisicion, costoAlimentacion, pesoActual, esHibrido, tipoAlimentacion, precioPorKiloBovinoEnPie);

if (\_animales.Contains(bovino)) throw new ArgumentException("Existe un Bovino con Código de Caravana Ingresado");

AltaAnimal(bovino);

Exito("Bovino Agregado Correctamente. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

catch (Exception ex)

{

Error($"{ex.Message} \n");

return;

}

}

public void PrecioPorKiloLana(int precioPorKiloLana)

{

if (precioPorKiloLana == 0) throw new ArgumentException("precioPorKiloLana = 0. Sistema\\PrecioPorKilogramoLana(int precioPorKiloLana) \n");

foreach (Animal animal in \_animales)

{

if (animal is Ovino)

{

Ovino ovino = (Ovino)animal;

ovino.PrecioPorKiloLana = precioPorKiloLana;

}

}

Exito("Precio por Kilogramo de Lana de los Ovinos Modificado con Éxito. Presione una Tecla Para Continuar. \n");

Console.ReadKey();

}

public void VacunarOvino()

{

Ovino ovino1 = ObtenerOvinoPorCodigoCaravana("Caravana1");

Ovino ovino2 = ObtenerOvinoPorCodigoCaravana("Caravana5");

Ovino ovino3 = ObtenerOvinoPorCodigoCaravana("Caravana10");

Ovino ovino4 = ObtenerOvinoPorCodigoCaravana("Caravana15");

Ovino ovino5 = ObtenerOvinoPorCodigoCaravana("Caravana20");

if (ovino1 is null || ovino2 is null || ovino3 is null || ovino4 is null || ovino5 is null) throw new ArgumentNullException("Object Null Sistema\\VacunarOvino() \n");

Vacuna vacuna1 = ObtenerVacunaPorNombre("Vacuna Antitetánica");

Vacuna vacuna2 = ObtenerVacunaPorNombre("Vacuna Anticlostridial");

Vacuna vacuna3 = ObtenerVacunaPorNombre("Vacuna Anticarbuncloso");

Vacuna vacuna4 = ObtenerVacunaPorNombre("Vacuna Antileptospira");

Vacuna vacuna5 = ObtenerVacunaPorNombre("Vacuna Antibrucelosis");

Vacuna vacuna6 = ObtenerVacunaPorNombre("Vacuna Antipasteurelosis");

Vacuna vacuna7 = ObtenerVacunaPorNombre("Vacuna Antirabia");

Vacuna vacuna8 = ObtenerVacunaPorNombre("Vacuna Antiviral");

Vacuna vacuna9 = ObtenerVacunaPorNombre("Vacuna Antiparasitaria");

Vacuna vacuna10 = ObtenerVacunaPorNombre("Vacuna Anticoccidial");

if (vacuna1 is null || vacuna2 is null || vacuna3 is null || vacuna4 is null || vacuna5 is null || vacuna6 is null || vacuna7 is null || vacuna8 is null || vacuna9 is null || vacuna10 is null) throw new ArgumentNullException("Object Null Sistema\\VacunarOvino() \n");

ovino1.Vacunar(vacuna1, DateTime.Now, DateTime.Now.AddMonths(6));

ovino1.Vacunar(vacuna2, DateTime.Now, DateTime.Now.AddMonths(6));

ovino1.Vacunar(vacuna3, DateTime.Now, DateTime.Now.AddMonths(6));

ovino2.Vacunar(vacuna3, DateTime.Now, DateTime.Now.AddMonths(6));

ovino2.Vacunar(vacuna4, DateTime.Now, DateTime.Now.AddMonths(6));

ovino3.Vacunar(vacuna5, DateTime.Now, DateTime.Now.AddMonths(6));

ovino3.Vacunar(vacuna6, DateTime.Now, DateTime.Now.AddMonths(6));

ovino4.Vacunar(vacuna7, DateTime.Now, DateTime.Now.AddMonths(6));

ovino4.Vacunar(vacuna8, DateTime.Now, DateTime.Now.AddMonths(6));

ovino5.Vacunar(vacuna9, DateTime.Now, DateTime.Now.AddMonths(6));

ovino5.Vacunar(vacuna10, DateTime.Now, DateTime.Now.AddMonths(6));

}

public void VacunarBovino()

{

Animal bovino1 = ObtenerBovinoPorCodigoCaravana("Caravana1");

Bovino bovino2 = ObtenerBovinoPorCodigoCaravana("Caravana5");

Bovino bovino3 = ObtenerBovinoPorCodigoCaravana("Caravana10");

Bovino bovino4 = ObtenerBovinoPorCodigoCaravana("Caravana15");

Bovino bovino5 = ObtenerBovinoPorCodigoCaravana("Caravana20");

if (bovino1 is null || bovino2 is null || bovino3 is null || bovino4 is null || bovino5 is null) throw new ArgumentNullException("Object Null. Sistema\\VacunarBovino() \n");

Vacuna vacuna1 = ObtenerVacunaPorNombre("Vacuna Antitetánica");

Vacuna vacuna2 = ObtenerVacunaPorNombre("Vacuna Anticlostridial");

Vacuna vacuna3 = ObtenerVacunaPorNombre("Vacuna Anticarbuncloso");

Vacuna vacuna4 = ObtenerVacunaPorNombre("Vacuna Antileptospira");

Vacuna vacuna5 = ObtenerVacunaPorNombre("Vacuna Antibrucelosis");

Vacuna vacuna6 = ObtenerVacunaPorNombre("Vacuna Antipasteurelosis");

Vacuna vacuna7 = ObtenerVacunaPorNombre("Vacuna Antirabia");

Vacuna vacuna8 = ObtenerVacunaPorNombre("Vacuna Antiviral");

Vacuna vacuna9 = ObtenerVacunaPorNombre("Vacuna Antiparasitaria");

Vacuna vacuna10 = ObtenerVacunaPorNombre("Vacuna Anticoccidial");

if (vacuna1 is null || vacuna2 is null || vacuna3 is null || vacuna4 is null || vacuna5 is null || vacuna6 is null || vacuna7 is null || vacuna8 is null || vacuna9 is null || vacuna10 is null) throw new ArgumentNullException("Object Null Sistema\\VacunarBovino() \n");

bovino1.Vacunar(vacuna1, DateTime.Now, DateTime.Now.AddMonths(6));

bovino1.Vacunar(vacuna2, DateTime.Now, DateTime.Now.AddMonths(6));

bovino1.Vacunar(vacuna3, DateTime.Now, DateTime.Now.AddMonths(6));

bovino2.Vacunar(vacuna3, DateTime.Now, DateTime.Now.AddMonths(6));

bovino2.Vacunar(vacuna4, DateTime.Now, DateTime.Now.AddMonths(6));

bovino3.Vacunar(vacuna5, DateTime.Now, DateTime.Now.AddMonths(6));

bovino3.Vacunar(vacuna6, DateTime.Now, DateTime.Now.AddMonths(6));

bovino4.Vacunar(vacuna7, DateTime.Now, DateTime.Now.AddMonths(6));

bovino4.Vacunar(vacuna8, DateTime.Now, DateTime.Now.AddMonths(6));

bovino5.Vacunar(vacuna9, DateTime.Now, DateTime.Now.AddMonths(6));

bovino5.Vacunar(vacuna10, DateTime.Now, DateTime.Now.AddMonths(6));

}

public void AltaPotrero(Potrero potrero)

{

try

{

if (potrero is null) throw new ArgumentNullException("Object Null Sistema\\AltaPotrero(Potrero potrero) \n");

potrero.Validar();

if (\_potreros.Contains(potrero)) throw new ArgumentException("Potrero ya Existe en Sistema\\List<Potrero> \_potreros \n");

\_potreros.Add(potrero);

}

catch (Exception ex)

{

Error(ex.Message);

}

}

public void PrecargarPotrero()

{

AltaPotrero(new Potrero("Potrero 1", 20, 50));

AltaPotrero(new Potrero("Potrero 2", 15, 40));

AltaPotrero(new Potrero("Potrero 3", 25, 60));

AltaPotrero(new Potrero("Potrero 4", 18, 45));

AltaPotrero(new Potrero("Potrero 5", 22, 55));

AltaPotrero(new Potrero("Potrero 6", 17, 42));

AltaPotrero(new Potrero("Potrero 7", 21, 58));

AltaPotrero(new Potrero("Potrero 8", 19, 48));

AltaPotrero(new Potrero("Potrero 9", 23, 63));

AltaPotrero(new Potrero("Potrero 10", 16, 38));

}

public void AltaVacuna(Vacuna vacuna)

{

try

{

if (vacuna is null) throw new ArgumentNullException("Object Null Sistema\\AltaVacuna(Vacuna vacuna) \n");

vacuna.Validar();

if (\_vacunas.Contains(vacuna)) throw new ArgumentException("Vacuna ya Existe en Sistema\\List<Vacuna> \_vacunas \n");

\_vacunas.Add(vacuna);

}

catch (Exception ex)

{

Error(ex.Message);

}

}

public void PrecargarVacuna()

{

AltaVacuna(new Vacuna("Vacuna Antitetánica", "Protege contra el tétanos", "Clostridium tetani"));

AltaVacuna(new Vacuna("Vacuna Anticlostridial", "Protege contra las infecciones por clostridios", "Clostridium perfringens"));

AltaVacuna(new Vacuna("Vacuna Anticarbuncloso", "Protege contra el ántrax", "Bacillus anthracis"));

AltaVacuna(new Vacuna("Vacuna Antileptospira", "Protege contra la leptospirosis", "Leptospira spp."));

AltaVacuna(new Vacuna("Vacuna Antibrucelosis", "Protege contra la brucelosis", "Brucella abortus"));

AltaVacuna(new Vacuna("Vacuna Antipasteurelosis", "Protege contra la pasteurelosis", "Pasteurella multocida"));

AltaVacuna(new Vacuna("Vacuna Antirabia", "Protege contra la rabia", "Virus de la rabia"));

AltaVacuna(new Vacuna("Vacuna Antiviral", "Protege contra enfermedades virales", "Diferentes virus"));

AltaVacuna(new Vacuna("Vacuna Antiparasitaria", "Protege contra parásitos internos y externos", "Diferentes parásitos"));

AltaVacuna(new Vacuna("Vacuna Anticoccidial", "Protege contra la coccidiosis", "Diferentes especies de coccidios"));

}

public void AltaAnimal(Animal animal)

{

if (animal is null) throw new ArgumentNullException("Object Null Sistema\\AltaAnimal(Animal animal) \n");

animal.Validar();

if (\_animales.Contains(animal)) throw new ArgumentException("Ovino ya Existe en Sistema\\List<Animal> \_animales \n");

\_animales.Add(animal);

}

public void PrecargarAnimal()

{

/\*\* Ovinos \*\*/

AltaAnimal(new Ovino("Caravana1", Sexo.Macho, "Raza1", new DateTime(2019, 01, 15), 1500, 200, 30.5, false, 5.2, 15, 20));

AltaAnimal(new Ovino("Caravana2", Sexo.Hembra, "Raza2", new DateTime(2020, 03, 22), 1600, 220, 35.2, true, 6.8, 18, 22));

AltaAnimal(new Ovino("Caravana3", Sexo.Macho, "Raza3", new DateTime(2021, 05, 10), 1700, 240, 32.7, false, 5.9, 16, 21));

AltaAnimal(new Ovino("Caravana4", Sexo.Hembra, "Raza4", new DateTime(2022, 07, 03), 1800, 260, 38.1, true, 7.5, 20, 25));

AltaAnimal(new Ovino("Caravana5", Sexo.Macho, "Raza5", new DateTime(2023, 09, 18), 1900, 280, 36.8, false, 6.3, 17, 23));

AltaAnimal(new Ovino("Caravana6", Sexo.Hembra, "Raza6", new DateTime(2024, 11, 25), 2000, 300, 42.4, true, 8.1, 22, 27));

AltaAnimal(new Ovino("Caravana7", Sexo.Macho, "Raza7", new DateTime(2025, 12, 10), 2100, 320, 39.6, false, 7.2, 19, 24));

AltaAnimal(new Ovino("Caravana8", Sexo.Hembra, "Raza8", new DateTime(2026, 10, 06), 2200, 340, 45.3, true, 9.3, 25, 30));

AltaAnimal(new Ovino("Caravana9", Sexo.Macho, "Raza9", new DateTime(2027, 08, 30), 2300, 360, 43.9, false, 8.5, 23, 28));

AltaAnimal(new Ovino("Caravana10", Sexo.Hembra, "Raza10", new DateTime(2028, 07, 20), 2400, 380, 49.7, true, 10.2, 27, 33));

AltaAnimal(new Ovino("Caravana11", Sexo.Macho, "Raza11", new DateTime(2018, 02, 14), 1500, 200, 34.2, false, 6.1, 17, 22));

AltaAnimal(new Ovino("Caravana12", Sexo.Hembra, "Raza12", new DateTime(2017, 04, 03), 1600, 220, 40.5, true, 7.9, 21, 26));

AltaAnimal(new Ovino("Caravana13", Sexo.Macho, "Raza13", new DateTime(2016, 06, 27), 1700, 240, 37.8, false, 7.0, 18, 23));

AltaAnimal(new Ovino("Caravana14", Sexo.Hembra, "Raza14", new DateTime(2015, 08, 12), 1800, 260, 44.6, true, 8.8, 24, 29));

AltaAnimal(new Ovino("Caravana15", Sexo.Macho, "Raza15", new DateTime(2014, 10, 05), 1900, 280, 41.3, false, 7.7, 20, 25));

AltaAnimal(new Ovino("Caravana16", Sexo.Hembra, "Raza16", new DateTime(2013, 12, 20), 2000, 300, 47.9, true, 9.6, 26, 31));

AltaAnimal(new Ovino("Caravana17", Sexo.Macho, "Raza17", new DateTime(2012, 07, 18), 2100, 320, 45.2, false, 8.4, 22, 27));

AltaAnimal(new Ovino("Caravana18", Sexo.Hembra, "Raza18", new DateTime(2011, 05, 30), 2200, 340, 51.8, true, 10.7, 28, 34));

AltaAnimal(new Ovino("Caravana19", Sexo.Macho, "Raza19", new DateTime(2010, 03, 25), 2300, 360, 49.1, false, 9.2, 24, 29));

AltaAnimal(new Ovino("Caravana20", Sexo.Hembra, "Raza20", new DateTime(2009, 01, 15), 2400, 380, 55.7, true, 11.3, 30, 35));

AltaAnimal(new Ovino("Caravana21", Sexo.Macho, "Raza21", new DateTime(2010, 1, 1), 1500, 200, 30.5, false, 5.2, 15, 40));

AltaAnimal(new Ovino("Caravana22", Sexo.Hembra, "Raza22", new DateTime(2011, 1, 1), 1600, 220, 35.2, true, 6.8, 18, 45));

AltaAnimal(new Ovino("Caravana23", Sexo.Macho, "Raza23", new DateTime(2012, 1, 1), 1700, 240, 32.7, false, 5.9, 16, 50));

AltaAnimal(new Ovino("Caravana24", Sexo.Hembra, "Raza24", new DateTime(2013, 1, 1), 1800, 260, 38.1, true, 7.5, 20, 55));

AltaAnimal(new Ovino("Caravana25", Sexo.Macho, "Raza25", new DateTime(2014, 1, 1), 1900, 280, 36.8, false, 6.3, 17, 60));

AltaAnimal(new Ovino("Caravana26", Sexo.Hembra, "Raza26", new DateTime(2015, 1, 1), 2000, 300, 42.4, true, 8.1, 22, 65));

AltaAnimal(new Ovino("Caravana27", Sexo.Macho, "Raza27", new DateTime(2016, 1, 1), 2100, 320, 39.6, false, 7.2, 19, 70));

AltaAnimal(new Ovino("Caravana28", Sexo.Hembra, "Raza28", new DateTime(2017, 1, 1), 2200, 340, 45.3, true, 9.3, 25, 75));

AltaAnimal(new Ovino("Caravana29", Sexo.Macho, "Raza29", new DateTime(2018, 1, 1), 2300, 360, 43.9, false, 8.5, 23, 80));

AltaAnimal(new Ovino("Caravana30", Sexo.Hembra, "Raza30", new DateTime(2019, 1, 1), 2400, 380, 49.7, true, 10.2, 27, 85));

/\*\* Bovinos \*\*/

AltaAnimal(new Bovino("Caravana1", Sexo.Macho, "Angus", new DateTime(2019, 01, 15), 1500, 200, 300, false, TipoAlimentacion.Grano, 25));

AltaAnimal(new Bovino("Caravana2", Sexo.Hembra, "Hereford", new DateTime(2020, 03, 22), 1600, 220, 320, true, TipoAlimentacion.Pastura, 30));

AltaAnimal(new Bovino("Caravana3", Sexo.Macho, "Simmental", new DateTime(2021, 05, 10), 1700, 240, 340, false, TipoAlimentacion.Grano, 35));

AltaAnimal(new Bovino("Caravana4", Sexo.Hembra, "Angus", new DateTime(2022, 07, 03), 1800, 260, 360, true, TipoAlimentacion.Pastura, 40));

AltaAnimal(new Bovino("Caravana5", Sexo.Macho, "Hereford", new DateTime(2023, 09, 18), 1900, 280, 380, false, TipoAlimentacion.Grano, 45));

AltaAnimal(new Bovino("Caravana6", Sexo.Hembra, "Simmental", new DateTime(2024, 11, 25), 2000, 300, 400, true, TipoAlimentacion.Pastura, 50));

AltaAnimal(new Bovino("Caravana7", Sexo.Macho, "Angus", new DateTime(2025, 12, 10), 2100, 320, 420, false, TipoAlimentacion.Grano, 55));

AltaAnimal(new Bovino("Caravana8", Sexo.Hembra, "Hereford", new DateTime(2026, 10, 06), 2200, 340, 440, true, TipoAlimentacion.Pastura, 60));

AltaAnimal(new Bovino("Caravana9", Sexo.Macho, "Simmental", new DateTime(2027, 08, 30), 2300, 360, 460, false, TipoAlimentacion.Grano, 65));

AltaAnimal(new Bovino("Caravana10", Sexo.Hembra, "Angus", new DateTime(2028, 07, 20), 2400, 380, 480, true, TipoAlimentacion.Pastura, 70));

AltaAnimal(new Bovino("Caravana11", Sexo.Macho, "Hereford", new DateTime(2018, 02, 14), 1500, 200, 300, false, TipoAlimentacion.Grano, 75));

AltaAnimal(new Bovino("Caravana12", Sexo.Hembra, "Simmental", new DateTime(2017, 04, 03), 1600, 220, 320, true, TipoAlimentacion.Pastura, 80));

AltaAnimal(new Bovino("Caravana13", Sexo.Macho, "Angus", new DateTime(2016, 06, 27), 1700, 240, 340, false, TipoAlimentacion.Grano, 85));

AltaAnimal(new Bovino("Caravana14", Sexo.Hembra, "Hereford", new DateTime(2015, 08, 12), 1800, 260, 360, true, TipoAlimentacion.Pastura, 90));

AltaAnimal(new Bovino("Caravana15", Sexo.Macho, "Simmental", new DateTime(2014, 10, 05), 1900, 280, 380, false, TipoAlimentacion.Grano, 95));

AltaAnimal(new Bovino("Caravana16", Sexo.Hembra, "Angus", new DateTime(2013, 12, 20), 2000, 300, 400, true, TipoAlimentacion.Pastura, 100));

AltaAnimal(new Bovino("Caravana17", Sexo.Macho, "Hereford", new DateTime(2012, 07, 18), 2100, 320, 420, false, TipoAlimentacion.Grano, 105));

AltaAnimal(new Bovino("Caravana18", Sexo.Hembra, "Simmental", new DateTime(2011, 05, 30), 2200, 340, 440, true, TipoAlimentacion.Pastura, 110));

AltaAnimal(new Bovino("Caravana19", Sexo.Macho, "Angus", new DateTime(2010, 03, 25), 2300, 360, 460, false, TipoAlimentacion.Grano, 115));

AltaAnimal(new Bovino("Caravana20", Sexo.Hembra, "Hereford", new DateTime(2009, 01, 15), 2400, 380, 480, true, TipoAlimentacion.Pastura, 120));

AltaAnimal(new Bovino("Caravana21", Sexo.Macho, "Simmental", new DateTime(2023, 01, 15), 1500, 200, 300, false, TipoAlimentacion.Grano, 125));

AltaAnimal(new Bovino("Caravana22", Sexo.Hembra, "Angus", new DateTime(2020, 03, 22), 1600, 220, 320, true, TipoAlimentacion.Pastura, 130));

AltaAnimal(new Bovino("Caravana23", Sexo.Macho, "Hereford", new DateTime(2021, 05, 10), 1700, 240, 340, false, TipoAlimentacion.Grano, 135));

AltaAnimal(new Bovino("Caravana24", Sexo.Hembra, "Simmental", new DateTime(2022, 07, 03), 1800, 260, 360, true, TipoAlimentacion.Pastura, 140));

AltaAnimal(new Bovino("Caravana25", Sexo.Macho, "Angus", new DateTime(2023, 09, 18), 1900, 280, 380, false, TipoAlimentacion.Grano, 145));

AltaAnimal(new Bovino("Caravana26", Sexo.Hembra, "Hereford", new DateTime(2024, 11, 25), 2000, 300, 400, true, TipoAlimentacion.Pastura, 150));

AltaAnimal(new Bovino("Caravana27", Sexo.Macho, "Simmental", new DateTime(2025, 12, 10), 2100, 320, 420, false, TipoAlimentacion.Grano, 155));

AltaAnimal(new Bovino("Caravana28", Sexo.Hembra, "Angus", new DateTime(2026, 10, 06), 2200, 340, 440, true, TipoAlimentacion.Pastura, 160));

AltaAnimal(new Bovino("Caravana29", Sexo.Macho, "Hereford", new DateTime(2027, 08, 30), 2300, 360, 460, false, TipoAlimentacion.Grano, 165));

AltaAnimal(new Bovino("Caravana30", Sexo.Hembra, "Simmental", new DateTime(2028, 07, 20), 2400, 380, 480, true, TipoAlimentacion.Pastura, 170));

}

public void AltaTarea(Tarea tarea)

{

try

{

if (tarea is null) throw new ArgumentNullException("Object Null AltaTarea() \n");

tarea.Validar();

if (\_tareas.Contains(tarea)) throw new ArgumentException("Tarea ya Existe en \_tareas \n");

\_tareas.Add(tarea);

}

catch (Exception ex)

{

Error(ex.Message);

}

}

public void PrecargarTarea()

{

AltaTarea(new Tarea("Preparar terreno para siembra", DateTime.Today.AddDays(1), false, DateTime.Today.AddDays(2), "Se necesita arar y fertilizar el terreno"));

AltaTarea(new Tarea("Sembrar cultivo de maíz", DateTime.Today.AddDays(3), false, DateTime.Today.AddDays(4), "Sembrar el maíz en las parcelas asignadas"));

AltaTarea(new Tarea("Regar cultivos", DateTime.Today.AddDays(5), false, DateTime.Today.AddDays(6), "Asegurarse de mantener una hidratación adecuada"));

AltaTarea(new Tarea("Fertilizar cultivos", DateTime.Today.AddDays(7), false, DateTime.Today.AddDays(8), "Aplicar fertilizantes según las necesidades del suelo"));

AltaTarea(new Tarea("Control de plagas", DateTime.Today.AddDays(9), false, DateTime.Today.AddDays(10), "Monitorear y aplicar tratamientos contra plagas"));

AltaTarea(new Tarea("Podar árboles frutales", DateTime.Today.AddDays(11), false, DateTime.Today.AddDays(12), "Realizar poda de forma adecuada para promover el crecimiento"));

AltaTarea(new Tarea("Cosechar cultivos", DateTime.Today.AddDays(13), false, DateTime.Today.AddDays(14), "Recolectar los cultivos en el momento óptimo"));

AltaTarea(new Tarea("Inspección de cercas", DateTime.Today.AddDays(15), false, DateTime.Today.AddDays(16), "Revisar la integridad de las cercas del campo"));

AltaTarea(new Tarea("Reparación de maquinaria agrícola", DateTime.Today.AddDays(17), false, DateTime.Today.AddDays(18), "Realizar mantenimiento y reparaciones según sea necesario"));

AltaTarea(new Tarea("Control de malezas", DateTime.Today.AddDays(19), false, DateTime.Today.AddDays(20), "Eliminar malezas que puedan competir con los cultivos"));

AltaTarea(new Tarea("Fertilizar terrenos vacíos", DateTime.Today.AddDays(21), false, DateTime.Today.AddDays(22), "Aplicar fertilizantes en áreas sin cultivos"));

AltaTarea(new Tarea("Revisión de sistemas de riego", DateTime.Today.AddDays(23), false, DateTime.Today.AddDays(24), "Asegurarse de que los sistemas de riego estén funcionando correctamente"));

AltaTarea(new Tarea("Control de humedad en suelo", DateTime.Today.AddDays(25), false, DateTime.Today.AddDays(26), "Monitorear niveles de humedad y ajustar riego según sea necesario"));

AltaTarea(new Tarea("Cercar área de pastoreo", DateTime.Today.AddDays(27), false, DateTime.Today.AddDays(28), "Instalar cercas temporales para el pastoreo del ganado"));

AltaTarea(new Tarea("Preparar suelos para próximas siembras", DateTime.Today.AddDays(29), false, DateTime.Today.AddDays(30), "Arar y acondicionar suelos para futuras siembras"));

}

public void AltaEmpleado(Empleado empleado)

{

try

{

if (empleado is null) throw new ArgumentNullException("Object Null AltaEmpleado(Empleado empleado) \n");

empleado.Validar();

if (\_empleados.Contains(empleado)) throw new ArgumentException("Capataz ya Existe en Sistema\\List<Empleado> \_empleados \n");

\_empleados.Add(empleado);

}

catch (Exception ex)

{

Error(ex.Message);

}

}

public void PrecargarEmpleado()

{

/\*\* Peones \*\*/

AltaEmpleado(new Peon("peon1@email.com", "password1", "Juan", new DateTime(2022, 1, 1), true));

AltaEmpleado(new Peon("peon2@email.com", "password2", "María", new DateTime(2022, 1, 2), false));

AltaEmpleado(new Peon("peon3@email.com", "password3", "Carlos", new DateTime(2022, 1, 3), true));

AltaEmpleado(new Peon("peon4@email.com", "password4", "Ana", new DateTime(2022, 1, 4), false));

AltaEmpleado(new Peon("peon5@email.com", "password5", "Pedro", new DateTime(2022, 1, 5), true));

AltaEmpleado(new Peon("peon6@email.com", "password6", "Luis", new DateTime(2022, 1, 6), false));

AltaEmpleado(new Peon("peon7@email.com", "password7", "Sofía", new DateTime(2022, 1, 7), true));

AltaEmpleado(new Peon("peon8@email.com", "password8", "Elena", new DateTime(2022, 1, 8), false));

AltaEmpleado(new Peon("peon9@email.com", "password9", "Diego", new DateTime(2022, 1, 9), true));

AltaEmpleado(new Peon("peon10@email.com", "password10", "Laura", new DateTime(2022, 1, 10), false));

/\*\* Capataces \*\*/

AltaEmpleado(new Capataz("capataz1@email.com", "password1", "Juan", new DateTime(2022, 1, 1), 10));

AltaEmpleado(new Capataz("capataz2@email.com", "password2", "María", new DateTime(2022, 1, 2), 8));

}

#endregion #region Métodos para Agregar o Modificar Información

#region Métodos Globales

/\*\* Métodos Globales \*\*/

public static void Resaltar(string mensaje, ConsoleColor color1)

{

Console.ForegroundColor = color1;

Console.WriteLine(mensaje);

Console.ForegroundColor = ConsoleColor.Gray;

}

public static void Exito(string mensaje)

{

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine(mensaje);

Console.ForegroundColor = ConsoleColor.Gray;

}

public static void Error(string mensaje)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.Error.WriteLine(mensaje);

Console.ForegroundColor = ConsoleColor.Gray;

}

#endregion Métodos Globales

}

}

## ClassLibrary\Enum\Sexo

namespace ClassLibrary.Enum

{

// Enumeración para el sexo del animal

public enum Sexo

{

Macho = 0, Hembra = 1

}

}

## ClassLibrary\Enum\TipoAlimentacion

namespace ClassLibrary.Enum

{

// Enumeración para el tipo de alimentación del bovino

public enum TipoAlimentacion

{

Grano = 0, Pastura = 1

}

}

## ClassLibrary\Enum\TipoAnimal

namespace ClassLibrary.Enum

{

public enum TipoAnimal

{

Ovino = 0, Bovino = 1

}

}

## ClassLibrary\Interface\IValidar

namespace ClassLibrary.Interface

{

internal interface IValidar

{

bool Validar();

}

}

## ClassLibrary\Animal

using ClassLibrary.Enum;

using ClassLibrary.Interface;

namespace ClassLibrary

{

// Clase base para todos los animales

public abstract class Animal : IValidar

{

protected string \_codigoCaravana;

protected Sexo \_sexo;

protected string \_raza;

protected DateTime \_fechaNacimiento;

protected decimal \_costoAdquisicion;

protected decimal \_costoAlimentacion;

protected double \_pesoActual;

protected bool \_esHibrido;

protected List<Vacunacion> \_vacunaciones = new List<Vacunacion>();

// Potrero al que está asignado el animal

internal Potrero \_potreroAsignado;

// Constructor Clase Base

protected Animal(string codigoCaravana, Sexo sexo, string raza, DateTime fechaNacimiento, decimal costoAdquisicion, decimal costoAlimentacion, double pesoActual, bool esHibrido)

{

\_codigoCaravana = codigoCaravana;

\_sexo = sexo;

\_raza = raza;

\_fechaNacimiento = fechaNacimiento;

\_costoAdquisicion = costoAdquisicion;

\_costoAlimentacion = costoAlimentacion;

\_pesoActual = pesoActual;

\_esHibrido = esHibrido;

}

/\*\* Get; Set; \*\*/

public Potrero PotreroAsignado

{

set { \_potreroAsignado = value; }

}

public List<Vacunacion> Vacunaciones

{

get { return \_vacunaciones; }

}

public bool EsHibrido

{

get { return \_esHibrido; }

}

public double PesoActual

{

get { return \_pesoActual; }

}

public decimal CostoAlimentacion

{

get { return \_costoAlimentacion; }

}

public decimal CostoAdquisicion

{

get { return \_costoAdquisicion; }

}

public Sexo Sexo

{

get { return \_sexo; }

}

public string CodigoCaravana

{

get { return \_codigoCaravana; }

}

/\*\* Vacunar un Aniamal \*\*/

public void Vacunar(Vacuna vacuna, DateTime fecha, DateTime vencimiento)

{

// Crear una nueva instancia de Vacunación

Vacunacion nuevaVacunacion = new Vacunacion(vacuna, fecha, vencimiento);

// Agregar la nueva vacunación a la lista de vacunaciones del animal

\_vacunaciones.Add(nuevaVacunacion);

}

/\*\* Métodos Globales \*\*/

public virtual bool Validar()

{

if (!String.IsNullOrEmpty(\_codigoCaravana) && \_codigoCaravana.Length == 8 && !String.IsNullOrEmpty(\_raza) && \_fechaNacimiento < DateTime.Today && \_costoAdquisicion > 0 && \_costoAlimentacion > 0 && \_pesoActual > 0) return true;

return false;

}

public override string ToString()

{

string mensaje;

mensaje = $"Código Caravana: {\_codigoCaravana} ➟ ";

mensaje += $"Sexo: {\_sexo} ➟ ";

mensaje += $"Raza: {\_raza} ➟ ";

mensaje += $"Fecha de Nacimiento: {\_fechaNacimiento} ➟ ";

mensaje += $"Costo de Adquisición: {\_costoAdquisicion} ➟ ";

mensaje += $"Costo de Alimentación: {\_costoAlimentacion} ➟ ";

mensaje += $"Peso Actual: {\_pesoActual} ➟ ";

mensaje += $"¿Es Híbrido?: {\_esHibrido} ➟ ";

mensaje += $"\n \n Registro de Vacunación: \n";

if (\_vacunaciones.Count > 0)

{

foreach (Vacunacion vacunacion in \_vacunaciones)

{

mensaje += $"\n ➜ {vacunacion} \n";

}

}

else

{

mensaje += $"\n ➜ No Hay Registros de Vacunación \n";

}

return mensaje;

}

public override bool Equals(object? obj)

{

Animal animal = obj as Animal;

return animal is not null && \_codigoCaravana == animal.\_codigoCaravana;

}

}

}

## ClassLibrary\Bovino

using ClassLibrary.Enum;

namespace ClassLibrary

{

// Clase para los bovinos, derivada de Animal

public class Bovino : Animal

{

private static int idContador = 1;

private int \_id;

private TipoAlimentacion \_tipoAlimentacion;

private decimal \_precioPorKiloBovinoEnPie;

// Constructor Clase Derivada

public Bovino(string codigoCaravana, Sexo sexo, string raza, DateTime fechaNacimiento, decimal costoAdquisicion, decimal costoAlimentacion, double pesoActual, bool esHibrido, TipoAlimentacion tipoAlimentacion, decimal precioPorKiloBovinoEnPie) : base(codigoCaravana, sexo, raza, fechaNacimiento, costoAdquisicion, costoAlimentacion, pesoActual, esHibrido)

{

\_id = idContador++;

\_tipoAlimentacion = tipoAlimentacion;

\_precioPorKiloBovinoEnPie = precioPorKiloBovinoEnPie;

}

/\*\* Get; Set; \*\*/

public int Id

{

get { return \_id; }

}

public decimal PrecioPorKiloBovinoEnPie

{

get { return \_precioPorKiloBovinoEnPie; }

}

public TipoAlimentacion TipoAlimentacion

{

get { return \_tipoAlimentacion; }

}

/\*\* Métodos Globales \*\*/

public override bool Validar()

{

base.Validar();

if (\_precioPorKiloBovinoEnPie > 0) return true;

return false;

}

public override string ToString()

{

string mensaje = base.ToString();

mensaje += $"\n Tipo de Alimentación: {\_tipoAlimentacion} ➟ ";

mensaje += $"Precio por Kilo de Bovino en Pie: {\_precioPorKiloBovinoEnPie}";

return mensaje;

}

public override bool Equals(object? obj)

{

Bovino bovino = obj as Bovino;

return bovino is not null && \_codigoCaravana == bovino.\_codigoCaravana;

}

}

}

## ClassLibrary\Capataz

using ClassLibrary.Interface;

namespace ClassLibrary

{

public class Capataz : Empleado

{

private static int idContador = 1;

private int \_id;

private int \_cantidadPersonasACargo;

/\*\* Constructor \*\*/

public Capataz(string email, string password, string nombre, DateTime fechaIngreso, int cantidadPersonasACargo) : base(email, password, nombre, fechaIngreso)

{

\_id = idContador++;

\_cantidadPersonasACargo = cantidadPersonasACargo;

}

// Propiedades de lectura y escritura

public int Id

{

get { return \_id; }

}

// Métodos

public override bool Validar()

{

base.Validar();

if (\_cantidadPersonasACargo > 0) return true;

return false;

}

public override string ToString()

{

string mensaje = base.ToString();

mensaje += $"Cantidad de Personas a Cargo: ${\_cantidadPersonasACargo}";

return mensaje;

}

public override bool Equals(Object obj)

{

Capataz capataz = obj as Capataz;

return capataz is not null && this.\_nombre.Equals(capataz.\_nombre) && this.\_email.Equals(capataz.\_email);

}

}

}

## ClassLibrary\Empleado

using ClassLibrary.Enum;

using ClassLibrary.Interface;

namespace ClassLibrary

{

// Clase base para todos los empleados

public abstract class Empleado : IValidar

{

protected string \_email;

protected string \_password;

protected string \_nombre;

protected DateTime \_fechaIngreso;

// Constructor

protected Empleado(string email, string password, string nombre, DateTime fechaIngreso)

{

\_email = email;

\_password = password;

\_nombre = nombre;

\_fechaIngreso = fechaIngreso;

}

// Métodos

public virtual bool Validar()

{

if (!string.IsNullOrEmpty(\_email) && !string.IsNullOrEmpty(\_password) && \_password.Length >= 8 && !string.IsNullOrEmpty(\_nombre) && \_fechaIngreso > DateTime.Today) return true;

return false;

}

public override string ToString()

{

string mensaje;

mensaje = $"Email: {\_email} ➟ ";

mensaje += $"Password: {\_password} ➟ ";

mensaje += $"Nombre: {\_nombre} ➟ ";

mensaje += $"Fecha de Ingreso: {\_fechaIngreso} ➟ ";

return mensaje;

}

}

}

## ClassLibrary\Ovino

using ClassLibrary.Enum;

namespace ClassLibrary

{

// Clase para los ovinos, derivada de Animal

public class Ovino : Animal

{

private static int idContador = 1;

private int \_id;

private double \_pesoLanaEstimado;

private decimal \_precioPorKiloLana;

private decimal \_precioPorKiloOvinoEnPie;

// Constructor clase Derivada

public Ovino(string codigoCaravana, Sexo sexo, string raza, DateTime fechaNacimiento, decimal costoAdquisicion, decimal costoAlimentacion, double pesoActual, bool esHibrido, double pesoLanaEstimado, decimal precioPorKiloLana, decimal precioPorKiloEnPie) : base(codigoCaravana, sexo, raza, fechaNacimiento, costoAdquisicion, costoAlimentacion, pesoActual, esHibrido)

{

\_id = idContador++;

\_pesoLanaEstimado = pesoLanaEstimado;

\_precioPorKiloLana = precioPorKiloLana;

\_precioPorKiloOvinoEnPie = precioPorKiloEnPie;

}

/\*\* Get; Set; \*\*/

public int Id

{

get { return \_id; }

}

public decimal PrecioPorKiloOvinoEnPie

{

get { return \_precioPorKiloOvinoEnPie; }

}

public decimal PrecioPorKiloLana

{

get { return \_precioPorKiloLana; }

set { \_precioPorKiloLana = value; }

}

public double PesoLanaEstimado

{

get { return \_pesoLanaEstimado; }

}

/\*\* Métodos Globales \*\*/

public override bool Validar()

{

base.Validar();

if (\_pesoLanaEstimado > 0 && \_precioPorKiloLana > 0 && \_precioPorKiloOvinoEnPie > 0) return true;

return false;

}

public override string ToString()

{

string mensaje = base.ToString();

mensaje += $"\n Peso Lana Estimado: ${\_pesoLanaEstimado} ➟ ";

mensaje += $"Precio por Kilo de Lana: ${\_precioPorKiloLana} ➟ ";

mensaje += $"Precio por Kilo de Ovino en Pie: ${\_precioPorKiloOvinoEnPie}";

return mensaje;

}

public override bool Equals(object? obj)

{

Ovino ovino = obj as Ovino;

return ovino is not null && this.\_codigoCaravana.Equals(ovino.\_codigoCaravana);

}

}

}

## ClassLibrary\Peon

namespace ClassLibrary

{

// Clase para el Peon, derivada de Animal

public class Peon : Empleado

{

private static int idContador = 1;

private int \_id;

private bool \_residenteEstancia;

private List<Tarea> \_tareasAsignadas = new List<Tarea>();

// Constructor

public Peon(string email, string password, string nombre, DateTime fechaIngreso, bool residenteEstancia) : base(email, password, nombre, fechaIngreso)

{

\_id = idContador++;

\_residenteEstancia = residenteEstancia;

}

/\*\* Get; Set; \*\*/

public List<Tarea> TareasAsignadas

{

get { return \_tareasAsignadas; }

set { \_tareasAsignadas = value; }

}

public int Id

{

get { return \_id; }

}

/\*\* Métodos Que Agregan o Modifican Información \*\*/

// Método para asignar tarea al peón

public void AsignarTarea(Tarea tarea)

{

try

{

if (tarea is null) throw new ArgumentException("Object Null. Peon.cs\\AsignarTarea(Tarea tarea)");

tarea.Validar();

\_tareasAsignadas.Add(tarea);

}

catch (Exception ex)

{

Sistema.Error(ex.Message);

}

}

/\*\* Métodos Globales \*\*/

public override bool Validar()

{

return base.Validar();

}

public override string ToString()

{

string mensaje = base.ToString();

mensaje += $"¿Es Residente de la Estancia?: {\_residenteEstancia} ➟ ";

mensaje += $"\n \n Tareas Asignadas: \n";

if (\_tareasAsignadas.Count > 0)

{

foreach (Tarea tarea in \_tareasAsignadas)

{

mensaje += $"\n ➜ {tarea} \n";

}

}

else

{

mensaje += $"\n ➜ No Hay Tareas Asignadas ";

}

return mensaje;

}

public override bool Equals(object? obj)

{

Peon peon = obj as Peon;

return peon is not null && \_nombre.Equals(peon.\_nombre) && \_email.Equals(peon.\_email);

}

}

}

## ClassLibrary\Potrero

using ClassLibrary.Interface;

namespace ClassLibrary

{

public class Potrero : IValidar

{

private static int idContador = 1;

private int \_id;

private string \_descripcion;

private double \_hectareas;

private int \_capacidadMaxima;

private List<Animal> \_animales = new List<Animal>();

public Potrero(string descripcion, double hectareas, int capacidadMaxima)

{

\_id = idContador++;

\_descripcion = descripcion;

\_hectareas = hectareas;

\_capacidadMaxima = capacidadMaxima;

}

#region Get; Set;

/\*\* Get; Set; \*\*/

public List<Animal> Animales

{

get { return \_animales; }

}

public int Id

{

get { return \_id; }

}

public double Hectareas

{

get { return \_hectareas; }

}

public int CapacidadMaxima

{

get { return \_capacidadMaxima; }

}

#endregion Get; Set;

#region Métodos que Agregan o Modifican Información

/\*\* Métodos que Agregan o Modifican Información \*\*/

public void AsignarPotrero(Animal animal, Potrero potrero)

{

try

{

// Verificar si el potrero tiene capacidad para más animales

if (potrero.\_animales.Count >= potrero.\_capacidadMaxima) throw new InvalidOperationException("El Potrero Está Lleno.");

// Agregar el animal al potrero y asignarle el potrero

potrero.\_animales.Add(animal);

animal.PotreroAsignado = potrero;

}

catch (Exception ex)

{

Console.WriteLine();

Sistema.Error($"{ex.Message} \n");

}

}

#endregion Métodos que Agregan o Modifican Información

#region Métodos Globales

/\*\* Métodos Globales \*\*/

public bool Validar()

{

if (!String.IsNullOrEmpty(\_descripcion) && \_hectareas > 0 && \_capacidadMaxima > 0 && \_animales.Count > 0) return true;

return false;

}

public override string ToString()

{

string mensaje;

mensaje = $"ID Potrero: {\_id} ➟ ";

mensaje += $"Descripción: {\_descripcion} ➟ ";

mensaje += $"Hectareas: {\_hectareas} ➟ ";

mensaje += $"Capacidad Máxima: {\_capacidadMaxima} ➟ ";

mensaje += $"\n \n Animales en Potrero: \n";

if (\_animales.Count > 0)

{

foreach (Animal animal in \_animales)

{

mensaje += $"\n ➜ {animal} \n";

}

}

else

{

mensaje += $"\n ➜ No hay registros de Animales en el Potrero \n";

}

return mensaje;

}

#endregion Métodos Globales

}

}

## ClassLibrary\Tarea

using ClassLibrary.Interface;

namespace ClassLibrary

{

public class Tarea : IValidar

{

private static int idContador = 1;

private int \_id;

private string \_descripcion;

private DateTime \_fechaPactada;

private bool \_completada;

private DateTime \_fechaCierre;

private string \_comentario;

public Tarea(string descripcion, DateTime fechaPactada, bool completada, DateTime fechaCierre, string comentario)

{

\_id = idContador++;

\_descripcion = descripcion;

\_fechaPactada = fechaPactada;

\_completada = completada;

\_fechaCierre = fechaCierre;

\_comentario = comentario;

}

/\*\* Get; Set; \*\*/

public int Id

{

get { return \_id; }

}

public bool Validar()

{

if (!String.IsNullOrEmpty(\_descripcion) && \_fechaPactada < DateTime.Today && \_fechaCierre < DateTime.Today && !String.IsNullOrEmpty(\_comentario)) return true;

return false;

}

public override string ToString()

{

string mensaje = string.Empty;

mensaje += $"ID Tarea: {\_id} ➟ ";

mensaje += $"Descripción: {\_descripcion} ➟ ";

mensaje += $"Fecha Pactada: {\_fechaPactada} ➟ ";

mensaje += $"¿Completada?: {\_completada} ➟ ";

mensaje += $"fecha de Cierre: {\_fechaCierre} ➟ ";

mensaje += $"Comentario: {\_comentario}";

return mensaje;

}

public override bool Equals(object? obj)

{

Tarea tarea = obj as Tarea;

return (tarea is not null) && this.\_id == tarea.\_id;

}

}

}

## ClassLibrary\Vacuna

using ClassLibrary.Interface;

namespace ClassLibrary

{

public class Vacuna : IValidar

{

private static int idContador = 1;

private int \_id;

private string \_nombre;

private string \_descripcion;

private string \_patogeno;

// Constructor

public Vacuna(string nombre, string descripcion, string patogeno)

{

\_id = idContador++;

\_nombre = nombre;

\_descripcion = descripcion;

\_patogeno = patogeno;

}

/\*\* Get; Set; \*\*/

public int Id

{

get { return \_id; }

}

public string Nombre

{

get { return \_nombre; }

}

/\*\* Métodos \*\*/

public bool Validar()

{

if (!String.IsNullOrEmpty(\_nombre) && !String.IsNullOrEmpty(\_descripcion) && !String.IsNullOrEmpty(\_patogeno)) return true;

return false;

}

public override string ToString()

{

string mensaje;

mensaje = $"Nombre: {\_nombre} ➟ ";

mensaje += $"Descripción: {\_descripcion} ➟ ";

mensaje += $"patógeno: {\_patogeno}";

return mensaje;

}

}

}

## ClassLibrary\Vacunacion

using ClassLibrary.Interface;

namespace ClassLibrary

{

public class Vacunacion : IValidar

{

private Vacuna \_tipoVacuna;

private DateTime \_fecha;

private DateTime \_vencimiento;

public Vacunacion(Vacuna tipoVacuna, DateTime fecha, DateTime vencimiento)

{

\_tipoVacuna = tipoVacuna;

\_fecha = fecha;

\_vencimiento = vencimiento;

}

public Vacuna TipoVacuna

{

get { return \_tipoVacuna; }

set { \_tipoVacuna = value; }

}

public DateTime Fecha

{

get { return \_fecha; }

set { \_fecha = value; }

}

public DateTime Vencimiento

{

get { return \_vencimiento; }

set { \_vencimiento = value; }

}

public bool Validar()

{

if (\_tipoVacuna is not null) return true;

return false;

}

public override string ToString()

{

string mensaje;

mensaje = $"Tipo de Vacuna: {\_tipoVacuna} ➟ ";

mensaje += $"Fecha: {\_fecha} ➟ ";

mensaje += $"Vencimiento: {\_vencimiento}";

return mensaje;

}

}

}