Trabajo Practico N°4

El programa es un sistema de ventas orientado al age of empires II . En la cual utilizando a los celtas podés armar tu civilización arrancando con un X cantidad de cada recurso y permitiendo comprar unidades o edificios militares. Toda entidad que se puede comprar está almacenada en una base de datos a los cuales voy a acceder mediante una consulta por el nombre de la entidad que se quiera comprar. Arrojará los costos de la misma y se podrá comprar en caso de cumplir las condiciones manejadas como excepciones. Cada venta es almacenada en la base de datos.

En caso de haber problemas con la base de datos .mdf hacer lo siguiente

Crear una base de datos llamada EntidadesDB y correr los siguientes 3 scripts :

```
USE [EntidadesDB]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[EntidadesMilitares](
	[Nombre] [varchar](20) NOT NULL,
	[Comida] [int] NOT NULL,
	[Madera] [int] NOT NULL,
	[Oro] [int] NOT NULL,
	[Piedra] [int] NOT NULL,
	[Tipo] [varchar](10) NOT NULL
) ON [PRIMARY]
GO
```

```
USE [EntidadesDB]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[UnidadesVendidas](
	[Nombre] [varchar](20) NOT NULL,
	[Tipo] [varchar](30) NOT NULL
) ON [PRIMARY]
GO
```

```
INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Espada larga',60,0,20,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Piquero',35,25,0,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Caballeria Ligera',80,0,0,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Jinete',60,0,75,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Camello',55,0,60,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Ballestero',0,25,45,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Guerrillero',25,35,0,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES

('Arquero a caballo',0,40,60,0,'Unidad');
```

```
INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES
('Woad raider',65,0,25,0,'Unidad');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES
('Cuartel',0,175,0,0,'Edificio');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES
('Establo',0,175,0,0,'Edificio');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES
('Galeria',0,175,0,0,'Edificio');

INSERT INTO EntidadesMilitares(nombre,comida,madera,oro,piedra,tipo) VALUES
('Castillo',0,0,0,650,'Edificio');
```

Temas implementados en:

- **Excepciones:** Se manejan las posibles excepciones propias del programa en la biblioteca de exceptions.
- **Test Unitarios:** Proyecto de test unitarios dónde se prueban algunas excepciones y que se instancie cierto campo
- **Tipos Genéricos:** El método EjecutarVenta (El que almacena en la base de datos a cada venta) es genérico y también se aplica en la clase XML;
- **Interfaces:** Se aplica la interfaz IMostrarDatos ya que no fue necesario aplicarlo en alguna relación de herencia (La clase padre no expone ningún dato que no exponga su derivada, por lo cual, es un método propio de cada clase)
- **Archivos y serialización:** Una vez finalizado el programa y al apretar el botón " Cerrar" saldrá un MessageBox Preguntando si deseo guardar la civilización creada en un archivo xml y un archivo de texto.
- **SQL y base de datos:** Todas las entidades que están a la venta están almacenados en la base de datos EntidadesDB a la cual voy a poder acceder mediante una consulta que

realiza el “botón Buscar”. De la misma forma, cada venta realizada se almacenará en la misma base de datos pero en otra tabla.

- **Hilos:** En el load del formulario, se ejecuta un nuevo hilo que se encarga de mantener los datos de los recursos disponibles actualizados cada vez que realizo una compra.
- **Eventos:** Cuando se ejecuta la venta, junto al botón comprar, se dispara un evento que se encarga de vaciar los textbox y el buscador.
- **Metodos de extensión:** La clase civilización posee un método extendido

Algunas aclaraciones: La clase edificio y unidad son bastantes similares… lo único en lo que difieren es que la cantidad de unidades para comprar tiene una limitación que es el limite de población. Por lo cual, no aplique una fuerte relación de herencia y polimorfismo con la clase base.