TRABAJO PRACTICO

N°4

FABRICA MILITAR DE TANQUES Y AVIONES

MACHACA GASTÓN

FUNCIONALIDAD PRETENDIDA:

Realizar una serie de producción de tanques y aviones en base a los materiales recibidos y la cantidad de personal para poder iniciar el proceso de ensamblaje.

Los tanques y aviones disponibles para la producción son:

- Tanque Tiger.
- Tanque Sherman.
- Tanque T-34.
- Avión Black Widow.
- Avión Henschel.
- Avión Ilyushin.

La fábrica está capacitada para producir en base a la cantidad de personal asignado. dependiendo el caso: el límite de producción de tanques y aviones será la mitad que usted usuario haya ingresado y se separaran en 2 secciones para producir tanques y aviones por separado (EJ: 100 de personal, 50 para tanques y 50 para aviones). Sí ingresa materiales de sobra, la producción se limitará a la capacidad de personal disponible y dejará de producir.

```
Fabrica<Tanque,Avion> fabricarTanquesyAviones = new Fabrica<Tanque,Avion>(100);
```

Dependiendo el tipo de modelo de tanque o avión a producir se van a requerir una serie de materiales:

TANQUE TIGER: CAÑON KWK, MOTOR MAYBACH Y ARMA MG34.

TANQUE SHERMAN: CAÑON M1, MOTOR R975 Y ARMA BROWNING.

TANQUE T-34: CAÑON F34, MOTOR DIESEL Y ARMA DT.

AVION BLACKWIDOW: CAÑON HISPANO, MOTOR PRATT Y ARMA BROWNING.

AVION ILYUSHIN: CAÑON NUDELMAN, MOTOR DAIMLER Y ARMA BEREZIN.

AVION HENSCHEL: CAÑON MG151, MOTOR GNOME Y ARMA MG17.

```
Tanque T1 = new Tanque(TipoCañon.KwK_43,10, TipoMotor.Maybach_HL_230_P30,10, TipoAmetralladoras.MG_34,10); // Tiger
Tanque T2 = new Tanque(TipoCañon.M1_76_mm,2, TipoMotor.R975_C4,2, TipoAmetralladoras.Browning_M1919A4,3); // Sherman
Tanque T3 = new Tanque(TipoCañon.F34_de_76_2_mm,3, TipoMotor.Diesel_V2_34,3, TipoAmetralladoras.DT,6); // T-34
```

```
Avion A1 = new Avion(Cañones.Hispano_M2,2,MotorRadial.Pratt_and_Whitney_R,5,Ametralladoras.Browning_M2,10);
Avion A2 = new Avion(Cañones.MG_151_20,10, MotorRadial.Gnome_Rhône_14M_5, 11, Ametralladoras.MG_17, 9);
Avion A3 = new Avion(Cañones.Nudelman_Suranov_NS_23, 1, MotorRadial.Daimler_Benz_DB_603,3, Ametralladoras.Berezin_UB,5);
```

Una vez ingresado los materiales, se inicia el proceso de fabricación:

Primero se trata de identificar el modelo del tanque en base a los materiales ingresados, así como también se realiza el mismo proceso para los aviones.

La fábrica se prepara para iniciar la producción y asigna al personal en su sección correspondiente (Administrar Capacidad).

Pero antes, se debe de corroborar si los materiales son lo suficientes para iniciar el proceso de producción, si por algún motivo, no llego a ingresar un material, no hay ningún problema, se enviará suministros para reemplazar lo faltante (Validar ingreso tanques-aviones).

Ahora es cuando una vez verificado todo, se procede a iniciar el proceso de ensamblaje, tanto para tanques como para aviones (Ensamblado tanques-aviones).

```
T1.IdentificarTanque();
T2.IdentificarTanque();
T3.IdentificarTanque();

A1.IdentificarAvion();
A2.IdentificarAvion();
A3.IdentificarAvion();

fabricarTanquesyAviones.AdministrarCapacidad();

fabricarTanquesyAviones.ValidarIngresoTanques(T1);
fabricarTanquesyAviones.ValidarIngresoTanques(T2);
fabricarTanquesyAviones.ValidarIngresoTanques(T3);

fabricarTanquesyAviones.ValidarIngresoAviones(A1);
fabricarTanquesyAviones.ValidarIngresoAviones(A2);
fabricarTanquesyAviones.ValidarIngresoAviones(A3);

Console.WriteLine(fabricarTanquesyAviones.EmsambladoTanques(T1));
Console.WriteLine(fabricarTanquesyAviones.EmsambladoTanques(T2));
Console.WriteLine(fabricarTanquesyAviones.EmsambladoTanques(T3));

Console.WriteLine(fabricarTanquesyAviones.EmsambladoAviones(A1));
Console.WriteLine(fabricarTanquesyAviones.EmsambladoAviones(A2));
Console.WriteLine(fabricarTanquesyAviones.EmsambladoAviones(A3));
```

```
Se inicio el ensamblado de TANQUE TIGER en base a los materiales disponibles...
Se inicio el ensamblado de TANQUE SHERMAN en base a los materiales disponibles...
Se inicio el ensamblado de TANQUE T-34 en base a los materiales disponibles...
Se inicio el ensamblado de AERONAVE BLACK WIDOW en base a los materiales disponibles...
Se inicio el ensamblado de AERONAVE HENSCHEL en base a los materiales disponibles...
Se inicio el ensamblado de AERONAVE ILYUSHIN en base a los materiales disponibles...

Capacidad maxima de la fabrica: 100
Capacidad del area de creacion de Tanques actualmente sin usar: 35
Capacidad del area de creacion de Aeronaves actualmente sin usar: 38
Cantidad de Tanques Tigers creados: 10
Cantidad de Tanques Shermans creados: 2
Cantidad de Tanques T-34 creados: 3
Cantidad de Aeronaves Black Widow creadas:2
Cantidad de Aeronaves Ilyushin creadas:1
Cantidad de Aeronaves Henschel creadas:9
```

Como podrá ver, se realizó con éxito la fabricación de tanques y aviones.

Ahora, seguramente lo ideal seria contar el mecanismo de producción:

Cuando la fabrica recibe los materiales, si usted ingresa una cantidad de 100 cañones, 50 motores y 20 armas para un tanque Tiger, solo se harán 20 tanques Tiger, ahora el resto de materiales sobrantes se guardarán en la fábrica, pero como la capacidad de personal de la fabrica esta limitado a 100 a 200, si fueran pocos los materiales, no se requerirá todo el personal para el trabajo, sobraran y la fábrica lo informara.

GRACIAS POR SU ATENCION Y CONTINUE VIENDO EL INFORME.

Clase 15: EXCEPCIONES

Utilizo excepciones si por algún motivo, el usuario ingresa un valor invalido para el personal. Y lo readministre (asignando un nuevo número de manera aleatoria entre 100 a 200) y traiga personal para iniciar la producción.

Así como también si al tratar crear un tanque invalido, salte una excepción que te niegue la creación del mismo y te informe que el tanque a fabricar no se puede fabricar (no hay modelos disponibles).

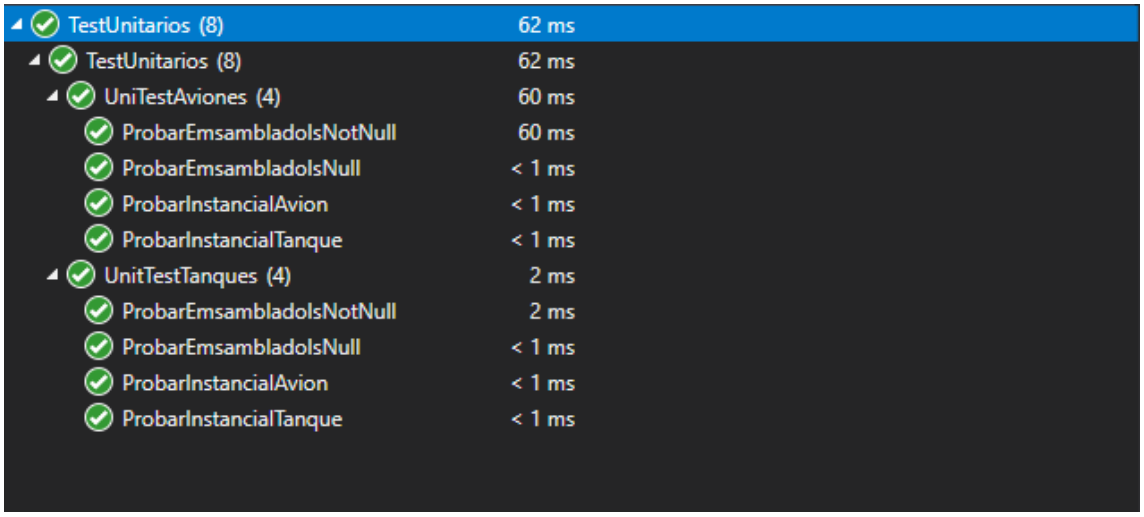
```
try
{
    Fabrica<Tanque, Avion> fabricaPrueba = new Fabrica<Tanque, Avion>(-1);
    fabricaPrueba.AdministrarCapacidad();
}
catch (ExcepcionCapacidad ex)
{
    Console.WriteLine(ex.Message);
}

Console.WriteLine("Pruebo excepcion, presione una tecla para continuar...");
Console.ReadKey();

try
{
    Tanque T4 = new Tanque(TipoCañon.KwK_43, 10, TipoMotor.R975_C4, 10, TipoAmetralladoras.MG_34, 10); // Tiger
    T4.IdentificarTanque();
}
catch (ExcepcionCompatibilidad ex)
{
    Console.WriteLine(ex.Message);
}

Console.ReadKey();
```

Clase 16: TEST UNITARIOS



Pruebo si la interfaz IAvion está en la jerarquía de herencia de avion o si no está.

```
/// <summary>
/// Prueba de Instancia de un Avion
/// </summary>
[TestMethod]
0 referencias
public void ProbarInstanciaIAvion()
{
    Avion A1 = new Avion(Cañones.Hispano_M2, 2, MotorRadial.Pratt_and_Whitney_R, 5, Ametralladoras.Browning_M2, 10);
    Assert.IsInstanceOfType(A1, typeof(IAvion));
}

/// <summary>
/// Prueba de Instancia de un Avion erronea.
/// </summary>
[TestMethod]
0 referencias
public void ProbarInstanciaITanque()
{
    Avion A1 = new Avion(Cañones.Hispano_M2, 2, MotorRadial.Pratt_and_Whitney_R, 5, Ametralladoras.Browning_M2, 10);
    Assert.IsNotInstanceOfType(A1, typeof(ITanque));
}
```

También pruebo si la creación del avión fue un éxito o si no existen los materiales para iniciar el proceso de fabricación.

```
/// <summary>
/// Prueba de Emsamblado de Avion no null.
/// </summary>
[TestMethod]
0 referencias
public void ProbarEmsambladoIsNotNull()
{
    Avion A1 = new Avion(Cañones.Hispano_M2, 2, MotorRadial.Pratt_and_Whitney_R, 5, Ametralladoras.Browning_M2, 10);
    Fabrica<Tanque, Avion> fabrica = new Fabrica<Tanque, Avion>(100);

    Assert.IsNotNull(fabrica.EmsambladoAviones(A1));
}

/// <summary>
/// Prueba de Emsamblado de Avion null.
/// </summary>
[TestMethod]
0 referencias
public void ProbarEmsambladoIsNull()
{
    Avion A1 = new Avion();
    Fabrica<Tanque, Avion> fabrica = new Fabrica<Tanque, Avion>(100);

    Assert.IsNotNull(fabrica.EmsambladoAviones(A1));
}
```

Así como también probé la interfaz de aviones, replico lo mismo para los tanques e identifico si está en la jerarquía de herencia o no.

```
/// <summary>
/// Prueba de Instancia de un Tanque
/// </summary>
[TestMethod]
0 referencias
public void ProbarInstanciaITanque()
{
    Tanque T1 = new Tanque(TipoCañon.KwK_43, 10, TipoMotor.Maybach_HL_230_P30, 10, TipoAmetralladoras.MG_34, 10); // Tiger
    Assert.IsInstanceOfType(T1, typeof(ITanque));
}

/// <summary>
/// Prueba de Instancia de un Tanque erronea.
/// </summary>
[TestMethod]
0 referencias
public void ProbarInstanciaIAvion()
{
    Tanque T1 = new Tanque(TipoCañon.KwK_43, 10, TipoMotor.Maybach_HL_230_P30, 10, TipoAmetralladoras.MG_34, 10); // Tiger
    Assert.IsNotInstanceOfType(T1, typeof(IAvion));
}
```

Y por último si al iniciar el ensamblado en la fábrica de un tanque este no es null.

```
    /// <summary>
    /// Prueba de Emsamblado de Tanque no null.
    /// </summary>
    [TestMethod]
    0 referencias
    public void ProbarEmsambladoIsNotNull()
    {
        Tanque T1 = new Tanque(TipoCañon.KwK_43, 10, TipoMotor.Maybach_HL_230_P30, 10, TipoAmetralladoras.MG_34, 10); // Tiger
        Fabrica<Tanque, Avion> fabrica = new Fabrica<Tanque, Avion>(100);

        Assert.IsNotNull(fabrica.EmsambladoTanques(T1));
    }

    /// <summary>
    /// Prueba de Emsamblado de Tanque null.
    /// </summary>
    [TestMethod]
    0 referencias
    public void ProbarEmsambladoIsNull()
    {
        Tanque T1 = new Tanque();
        Fabrica<Tanque, Avion> fabrica = new Fabrica<Tanque, Avion>(100);

        Assert.IsNotNull(fabrica.EmsambladoTanques(T1));
    }
```

Clase 17: TIPOS GENERICOS y Clase 19: ARCHIVOS Y SERIALIZACIÓN

Para los genéricos lo que hice fue hacer que se pueda crear un informe genérico ya sea para tanques, aviones o la misma fabrica de los materiales que se utilizaron para la producción. En el caso de los tanques los materiales que se pasaron para la creación, lo mismo para avión y para fabrica el informe contendrá la cantidad de materiales sobrantes de la producción en general de la fábrica.

Y para archivos que toda esa información la recopile y la guarde en la carpeta test del proyecto (más exacto en la carpeta test/bin/debug) en formato XML y lo serialice.

```
/// <summary>
/// Metodo que se encarga de generar archivo XML de materiales de un tanque,avion o la fabrica.
/// </summary>
/// <param name="dato">Generico</param>
2 referencias
public void GenerarInformeXML(T dato)
{
    mensaje = typeof(T).Name;
    XmlSerializer s = new XmlSerializer(typeof(T));

    using (TextWriter tw = new StreamWriter($"{mensaje}.xml"))
    {
        s.Serialize(tw, dato);
    }
}
```

```
Informes<Tanque> informeTanques = new Informes<Tanque>();
Informes<Avion> informeAviones = new Informes<Avion>();

informeTanques.GenerarInformeXML(T1);
informeAviones.GenerarInformeXML(A1);
```

Clase 18: INTERFACES

Hice 2 interfaces para poder obtener en todo momento el modelo de tanque o avión que se este fabricando.

```
4 referencias
public interface IAvion
{
    7 referencias
    Aeronaves ObtenerDatosAviones { get; }
}

4 referencias
public interface ITanque
{
    7 referencias
    TipoTanque ObtenerDatosTanques { get; }
}
```

Clase 21: SQL y Clase 22: Base de datos

En esta clase se hace uso de la conexión SQL para pasar los datos de los materiales para producir y almacenarlos en la base de datos, por cada tanque o avión que se crea, se guarda en la base de datos. El .bak se encuentra en la carpeta base de datos del proyecto.

```
/// <summary>
/// Constructor sin parametros que inicializa el atributo de conexion
/// </summary>
5 referencias
public SQL()
{
    this.conexion = new SqlConnection("Data Source =.; Initial Catalog = MachacaGastonTP4; Integrated Security = True");
}
```

```
2 referencias
public bool InsertarTanque(Tanque tanque)
{
    bool retorno;

    int AmetralladoraSherman = tanque.CantidadAmetralladorasShermans;
    int AmetralladoraTigers = tanque.CantidadAmetralladorasTigers;
    int AmetralladoraT34 = tanque.CantidadAmetralladoras_T34;
    int CañonesShermans = tanque.CantidadCañonesShermans;
    int CañonesTiger = tanque.CantidadCañonesTigers;
    int CañonesT34 = tanque.CantidadCañones_T34;
    int MotoresSherman = tanque.CantidadMotoresShermans;
    int MotoresTiger = tanque.CantidadMotoresTigers;
    int MotoresT34 = tanque.CantidadMotores_T34;
    Tanques.TipoTanque Tipo = tanque.ObtenerDatosTanques;
```

```
3 referencias
public bool InsertarAvion(Avion avion)
{
    bool retorno;

    int AmetralladorasBlackWidow = avion.CantidadAmetralladorasBlackWindows;
    int AmetralladorasIlyushin = avion.CantidadAmetralladorasIlyushins;
    int AmetralladorasHenschel = avion.CantidadAmetralladorasHenschels;
    int CañonesBlackWidow = avion.CantidadCañonesBlackWindows;
    int CañonesIlyushin = avion.CantidadCañonesIlyushins;
    int CañonesHenschel = avion.CantidadCañonesHenschels;
    int MotoresBlackWidow = avion.CantidadMotoresBlackWindows;
    int MotoresIlyushin = avion.CantidadMotoresIlyushins;
    int MotoresHenschel = avion.CantidadMotoresHenschels;
    Aviones.Aeronaves Tipo = avion.ObtenerDatosAviones;
```

Lamentablemente la línea es muy extensa y la resolución no se llega a ver con detalle, pero sigo pasando por cada producto creado, sus materiales.

```
string sql = "INSERT INTO Tabla_Tanques (TipoDelTanque, AmetralladoraTigers, AmetralladoraT34, AmetralladoraSherman, CañonTigers, CañonT34, CañonSherman, MotoresTigers, MotoresT34, MotoresSherm
sql += "VALUES (@TipoDelTanque, @AmetralladoraTigers, @AmetralladoraT34, @AmetralladoraSherman, @CañonTigers, @CañonT34, @CañonSherman, @MotoresTigers, @MotoresT34, @MotoresSherman)";
```

```
string sql = "INSERT INTO Tabla_Aviones (TipoDeAvion, AmetralladorasBlackWindow, AmetralladorasIlyushin, AmetralladorasHenschel, CañonesBlackWidow, CañonesIlyushin, CañonesHenschel, MotoresBlack
sql += "VALUES (@TipoDeAvion, @AmetralladorasBlackWindow, @AmetralladorasIlyushin, @AmetralladorasHenschel, @CañonesBlackWidow, @CañonesIlyushin, @CañonesHenschel, @MotoresBlackWidow, @MotoresIl
```

Clase 23: Hilos

Para este tema lo que se hace es crear un hilo de formulario por cada elemento que se debe de agregar. Si usted está en el menú de producción se dará cuenta que cada vez que intente pasar una cantidad del material solicitado aparecerá un formulario personalizado por cada elemento del combo box.

Tal es así que dependiendo de si seleccionaste motores, cañones o ametralladoras, podrás ver el mismo formulario, pero tomando la cantidad de diferentes tipos de material necesario para la producción.

Se crean 6 hilos: 3 para los materiales del tanque y 3 para los del avión.

```
Thread hiloMotorTanque;
Thread hiloCañonTanque;
Thread hiloArmaTanque;

Thread hiloMotorAvion;
Thread hiloCañonAvion;
Thread hiloArmaAvion;
```

Este mismo código de réplica exactamente igual para los demás materiales, pero cada hilo será independiente.

```
/// <summary>
/// Inicia el hilo tanques-cañon al cambiar la seleccion.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void CañonTanques_SelectedIndexChanged(object sender, EventArgs e)
{
    cañonTanqueActual = CañonTanques.Text;

    if (capacidadMax != 0)
    {
        if (this.CañonTanques.SelectedIndex >= 0)
        {

            if (this.hiloCañonTanque != null)
            {
                this.hiloCañonTanque.Abort();
            }

            this.hiloCañonTanque = new Thread(AbrirFormCañonesTanques);
            this.hiloCañonTanque.IsBackground = true;

            if (hiloCañonTanque.IsAlive == false)
            {
                hiloCañonTanque.Start();
            }
        }
    }
}
```

Clase 24: Eventos y delegados

En esta clase lo que hacemos es usar el delegado en la clase Tanque y también el evento en cuestión.

Para lo que se utiliza eventos es que por cada vez que uno ingresa un tanque este se guarde en un informe semanal en el escritorio que te muestre día y hora de cada tipo de tanque que estés creando, así como también los materiales del mismo.

```
public delegate void DelegadoEvento(string mensaje);
```

```
public event DelegadoEvento Evento;
```

Acción del evento en la creación del tanque en el formulario "ProduccionEmsamblado".

```
tanque.Evento += ManejadorEvento;
tanque.PasarInformacion(tanque);
```

Si por algún motivo no se acciona el evento, este no invocara al método de guardado y salteara el paso.

```
public void PasarInformacion(string info)
{
    if(!object.ReferenceEquals(this.Evento,null))
    {
        this.Evento.Invoke(info);
    }
}
```

Guardo el informe en el escritorio con el nombre de "Informe.log".

```
/// <summary>
/// Evento que se acciona al confirmar produccion y enlista la produccion de tanques.
/// </summary>
/// <param name="produccion"></param>
1 referencia
protected void ManejadorEvento(string produccion)
{
    StreamWriter mensaje = new StreamWriter(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\Informe.log", true);

    mensaje.WriteLine("\nDia y hora: " + DateTime.Now + "\n"+ produccion);

    mensaje.Close();
}
```

Clase 25: Métodos de Extensión

Para este tema, extendí las clases ExcepcionCapacidad, ExcepcionConfirmar que incluye a ExcepcionCompatibilidad para que informe con un mensaje acorde a la excepción.

```
public static class Extensiones
{
    2 referencias
    public static string CantidadEmpleados(this ExcepcionCapacidad capacidad)
    {
        string mensaje = "Debe haber al menos 100 a 200 empleados...";

        return mensaje;
    }

    1 referencia
    public static string ConfirmarEmpleados(this ExcepcionConfirmar confirmar)
    {
        string mensaje = "No se ha seleccionado todos los parametros solicitados...";

        return mensaje;
    }

    2 referencias
    public static string MostrarCompatibilidad(this ExcepcionCompatibilidad confirmar)
    {
        StringBuilder mensaje = new StringBuilder();

        mensaje.AppendLine("Tanque Tiger: \nRequiere: \nMotor - Maybach || Cañon - KwK 43 || Ametralladora - MG 34");
        mensaje.AppendLine("\nTanque Sherman: \nRequiere: \nMotor - R975 || Cañon - M1 76 || Ametralladora - Browning");
        mensaje.AppendLine("\nTanque T-34: \nRequiere: \nMotor - Diesel || Cañon - F34 || Ametralladora - DT");
        mensaje.AppendLine("\nAvion BlackWidow \nRequiere: \nMotor - Pratt || Cañon - Hispano || Ametralladora - Browning");
        mensaje.AppendLine("\nAvion Ilyushin \nRequiere: \nMotor - Daimler || Cañon - Nudelman || Ametralladora - Berezin");
        mensaje.AppendLine("\nAvion Henschel \nRequiere: \nMotor - Gnome || Cañon - MG 151 20 || Ametralladora - MG 17");

        return mensaje.ToString();
    }
```