Un proceso es un programa en ejecución que tiene asignados recursos tales como memoria e hilos.

Todos los hilos de un mismo proceso comparten los mismos recursos asignados por el sistema operativo.

Puede ser una petición a una base de datos o sino algún cálculo matemática a una unidad de procesamiento (ambos muy costosos).

Programacion multi-hilo

Por defecto, cada proceso tiene un único hilo.

La programación multi-hilo (multithreaded programming) permite que un proceso se ejecute sobre múltiples hilos y cada uno de esos hilos esté realizando una tarea distinta en paralelo.

La programación en paralelo (parallel programming) es un sub-tipo de programación multi-hilo.
Se utiliza para dividir una gran carga de trabajo en partes independientes y ejecutarlas en paralelo, maximizando el uso de los núcleos de la CPU.

Task

● Task es la clase que utilizaremos para ejecutar métodos en un nuevo hilo.

● Task nos permitirá manejar también el estado de dicho hilo.

● Task es la evolución de la clase Thread de .Net Framework. Lo mejora, facilita la utilización, la hace más sencilla.

Task task = Task.Run(ImprimirHora);
Tiene una sobrecarga que es un CancellationToken, que permite cancelar la tarea secundaria.

RECUPERAR DATOS DE UNA BASE DE DATOS (LEER - GET)

```
    private static string stringConnection;
    private static SqlConnection connection;

stringConnection = "Server=.;Database=20220621SP;Trusted_Connection=True;";

    try
       {
          using (connection = new SqlConnection(stringConnection))
          {
```

```
            string sentencia = $"SELECT * FROM {tabla} WHERE id = @id";

            SqlCommand cmd = new SqlCommand(sentencia, connection);
            connection.Open();

            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
               return reader.GetString(1); //1 es el id
            }
            throw new DataBaseManagerException("ID innexistente");

        }
      }
      catch (Exception ex)
      {

        throw new DataBaseManagerException("Error al leer el nombre", ex);
      }
```

ESCRIBIR EN UNA BASE DE DATOS

```
private static string stringConnection;
private static SqlConnection connection;

stringConnection = "Server=.;Database=20220621SP;Trusted_Connection=True;";

      try
        {
           using (connection = new SqlConnection(stringConnection))
           {
              string sentencia = "INSERT INTO RESULTADOS (escuderia, posicion,
horaLlegada) VALUES (@escuderia, @posicion, @horaLlegada)";
              SqlCommand cmd = new SqlCommand(sentencia, connection);
              //evitar la inyección
              cmd.Parameters.AddWithValue("escuderia", autoF1.Escuderia);
              cmd.Parameters.AddWithValue("posicion", autoF1.Posicion);
              cmd.Parameters.AddWithValue("horaLlegada",DateTime.Now.ToString());

              conexion.Open();
              cmd.ExecuteNonQuery();
           }
        }
        catch (Exception)
        {
           throw new Exception("Error al guardar el vehíchulo");
        }
```

SERIALIZAR EN XML

```
        string rutaEscritorio =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
        string archivo = "Bombero.xml";
        ruta = Path.Join(rutaEscritorio, archivo);

    public void SERIALIZAR(Bombero info) //write
    {
        try
        {
            using (StreamWriter sw = new StreamWriter(ruta))
            {
                XmlSerializer xmlSerializer = new XmlSerializer(typeof(Bombero));
                xmlSerializer.Serialize(sw, info); //info es lo que quiero guardar
            }
        }
        catch (Exception)
        {

            throw new Exception("Error al guardar.");
        }

    }
```

DESERIALIZAR EN XML

```
        string rutaEscritorio = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
        string archivo = "Bombero.xml";
        ruta = Path.Join(rutaEscritorio, archivo);

    public Bombero Leer() //read
    {
        try
        {
            if (Path.GetExtension(ruta) == ".xml")
            {
                using (StreamReader sr = new StreamReader(ruta))
                {
                    XmlSerializer xmlSerializer = new XmlSerializer(typeof(Bombero));
                    return (Bombero)xmlSerializer.Deserialize(sr);
                }
            }
            else
            {
                throw new Exception("El archivo no es .xml");
            }
        }
        catch (NullReferenceException)
```

```
        {
            throw new NullReferenceException("No hay ningun bombero en la lista.");
        }
        catch (Exception)
        {
            throw new Exception("Error al leer el archivo");
        }
    }
```

SERIALIZADO EN .JSON

```
ArchivoTexto archivoTexto = new ArchivoTexto();
archivoTexto.Escribir(JsonSerializer.Serialize(dato, typeof(Persona)), path);
```

DESERIALIZAR EN JSON

```
ArchivoTexto archivoTexto = new ArchivoTexto();
string texto = archivoTexto.Leer(path);
return JsonSerializer.Deserialize<T>(texto);
```