

**Unidad**

**6**

DIPLOMATURA EN PROGRAMACION .NET

Ejercicios

---

Tecnológica Nacional - Derechos Reservados

## Capítulo 11

# Corrientes de Entrada y Salida

## Capítulo 11

Para los ejercicios del capítulo, abrir la solución cuyo nombre es Capitulo11.

### Ejercicio 1

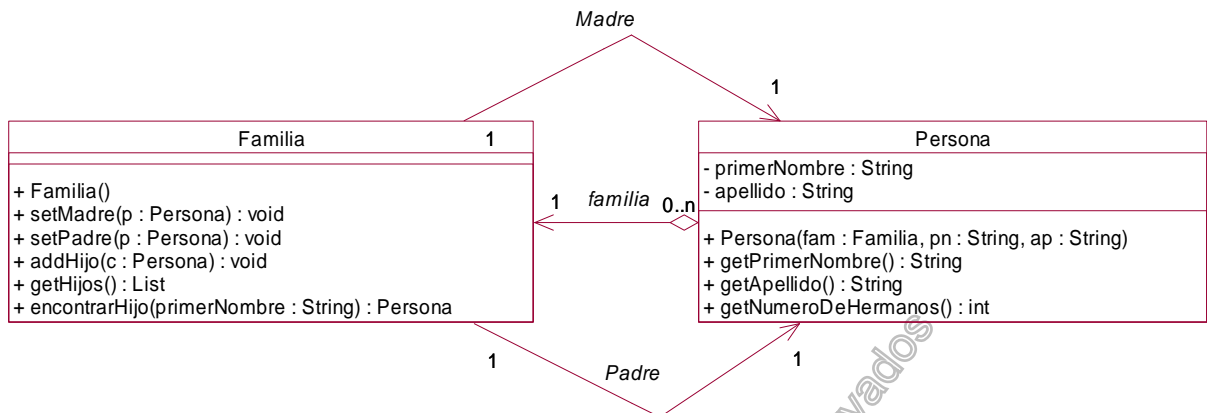
En este ejercicio se creará un programa para leer texto de la corriente estándar de entrada y escribir el mismo en un archivo, anteponiendo a cada ingreso el número de línea que corresponde a cada entrada del archivo.

1. Crear una instancia de la clase FileInfo con la ruta provista
2. Verificar si el archivo existe. En caso de ser así, borrarlo
3. Abrir una corriente de escritura usando la clase StreamWriter. La apertura se debe hacer con gestión de recursos automáticos para que cuando se cierre el bloque se llame al método Dispose.
4. Leer de la corriente estándar de ingreso por consola las líneas que ingrese el usuario (recordar que la línea se lee cuando se presiona la tecla Enter que significa "limpiar el búfer").
5. Escribir cada línea ingresada por consola por el usuario en el disco. Para ello, realizar las lecturas y escrituras dentro de un ciclo. Finalizar el ciclo cuando el tamaño de la línea ingresada es 0.
6. Todo el proceso de lectura / escritura se debe realizar controlando que no se produzca una excepción.
7. Abrir una corriente de lectura usando la clase StreamReader. La apertura se debe hacer con gestión de recursos automáticos para que cuando se cierre el bloque se llame al método Dispose.
8. Leer el archivo que se acaba de escribir en disco y mostrarlo por consola
9. Escribir cada línea ingresada por consola por el usuario en el disco. Para ello, realizar las lecturas y escrituras dentro de un ciclo. Finalizar el ciclo cuando el tamaño de la línea ingresada es 0.

### Ejercicio 2

#### Objetivo

En este ejercicio se van a leer y escribir objetos serializados a un archivo. El siguiente diagrama muestra las relaciones entre las clases del proyecto.



### Modificar la clase Familia

La clase Familia se ha implementado. Lo que se debe realizar es una modificación para que soporte serialización.

En el archivo de la clase Familia se encuentra un bloque de comentarios que empieza `/**` y termina con `*/` en C# y empieza y termina con `'-----'` en VB. Este comentario indica el lugar en el cual deberá proveer el código para el programa. Los archivos fuente para que construya el proyecto se encuentran en el proyecto de nombre ejercicio2.

1. Declarar que la clase Familia es serializable

### Modificar la clase Persona

Se debe modificar la clase Persona para que soporte serialización.

En el archivo de la clase Persona se encuentran bloques de comentarios que empiezan `/**` y terminan con `*/` en C# y empiezan y terminan con `'-----'` en VB. Estos comentarios indica el lugar en el cual deberá proveer el código para el programa.

2. Declarar que la clase Persona es serializable
3. El atributo `numeroDeHermanos` se calcula en el procesamiento, de manera que no necesita ser serializado. Crear la declaración de `numeroDeHermanos` de manera que el mismo no se serialice.
4. La propiedad `NumeroDeHermanos` retorna el número de hermanos para esa persona. El atributo se calcula cuando se procesa a la persona en particular. Esta propiedad deberá revisar si el atributo ya fue inicializado (que no sea cero) para luego realizar los cálculos. El cálculo es: restar uno al número total de hijos de esta familia. Por último, retornar el valor calculado

### Completar el programa principal

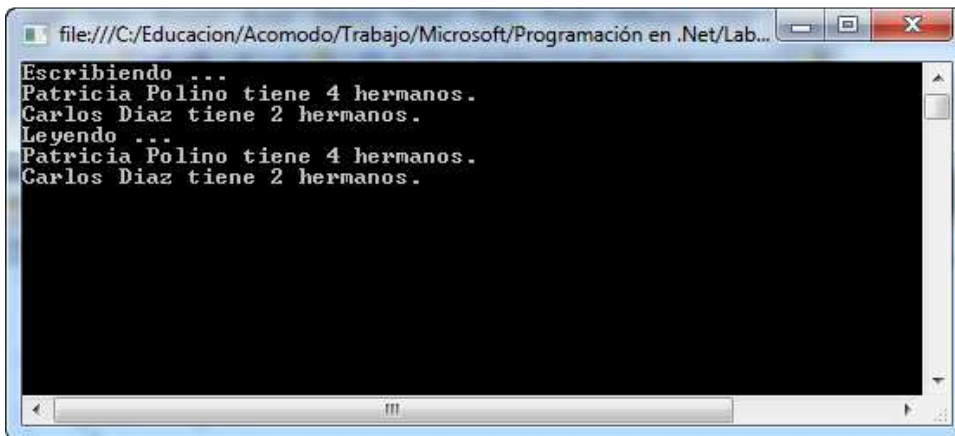
Este programa crea tres familias y la gente que le pertenece. Inicializa las relaciones entre los familiares y sus padres. Luego consulta el “número de hermanos” para Patricia Polino y Carlos Diaz para demostrar el cálculo en un atributo no serializable.

En el archivo de la clase que contiene el método Main se encuentran bloques de comentarios que empiezan `/**` y terminan con `*/` en C# y empiezan y terminan con `' -----` en VB. Estos comentarios indica el lugar en el cual deberá proveer el código para el programa.

5. En el método `EscribirFamilias`, crear una corriente de salida utilizando `FileStream`, donde el modo de apertura es `FileMode.OpenOrCreate` y el acceso es `FileAccess.Write`. Nombrar al archivo que contiene a los objetos familias: `familias.ser`. El proceso se debe realizar con gestión automática de recursos.
6. Agregar el código para escribir las tres familias a la corriente de salida de objetos. El formato de salida de la serialización es binario.

El programa debe leer las tres familias y recuperar el objeto `Persona` para Patricia Folino y Carlos Diaz. Para ello utiliza el método `LeerFamilias` en donde además se consulta por el número de hermanos para cada uno de ellos para demostrar el uso de atributo no serializable `numeroDeHermanos` el cual no se guardó en disco y deberá recalcularse.

7. En el método `LeerFamilias`, crear una corriente de entrada utilizando `FileStream`, donde el modo de apertura es `FileMode.Open` y el acceso es `FileAccess.Read`. Leer desde el archivo que contiene a los objetos familias creados anteriormente: `familias.ser`. El proceso se debe realizar con gestión automática de recursos.
8. Agregar el código para leer las tres familias a la corriente de entrada de objetos.
9. Compilar y ejecutar el programa. Se deberá ver la siguiente salida



```
file:///C:/Educacion/Acomodo/Trabajo/Microsoft/Programación .Net/Lab...
Escribiendo ...
Patricia Polino tiene 4 hermanos.
Carlos Diaz tiene 2 hermanos.
Leyendo ...
Patricia Polino tiene 4 hermanos.
Carlos Diaz tiene 2 hermanos.
```

### Ejercicio 3

En este ejercicio modificará el proyecto Banco para que el reporte generado cree un archivo en disco en lugar de una salida por pantalla para el reporte.

En los distintos archivos fuente a modificar se encuentran bloques de comentarios que empiezan `/**` y terminan con `*/` en C# y empiezan y terminan con `'-----'` en VB. Estos comentarios indica el lugar en el cual deberá proveer el código para el programa.

#### Modificar la clase ReporteCliente

1. Crear una instancia de la clase FileInfo con la ruta provista
2. Verificar si el archivo existe. En caso de ser así, borrarlo
3. Abrir una corriente de escritura usando la clase StreamWriter. La apertura se debe hacer con gestión de recursos automáticos para que cuando se cierre el bloque se llame al método Dispose.
4. Modificar todas las salidas por consola para que se hagan sobre la corriente que se abrió para escritura.
5. Verificar que no existan errores de compilación y ejecutar el programa
6. Verificar que la salida obtenida sea la misma que se obtenía por pantalla