1.- Queremos crear una **clase Menu** que nos facilite la interacción con el usuario a través de la consola. Será similar a otro que hicimos anteriormente, pero deberás implementarlo usando objetos List Especificación:

Sin constructor específico (no hay límite de opciones)

- void Add(String …) Añade una opción más al menú.
- int GetNumOpciones() Retorna el número de opciones disponibles.
- String GetOpcion(int n) Retorna el texto de la opción indicada ( ***0 < n < getNumOpciones()*** )
- String ToString() Retorna el texto (incluidos ret. de carro) de cómo debe imprimirse el menú.
- int Leer() Retorna la opción válida elegida a través de la consola . Asume hasta tres fallos, después lanza excepción. Si no tiene ninguna opción lanza excepción.

Una vez que se ejecute *ToString* o *Leer*, el menú no aceptará que se le añadan nuevas opciones.

2.- Queremos informatizar una lista de clientes. De ellos queremos saber el nombre y sus apellidos, que no deben ser nulos. A cada cliente, le asignaremos un DNI (que deberá ser válido).
Para ello deberás comenza por implementar la *clase Cliente*, heredando de la clase *Persona* vista en clase.

Usaremos un objeto de tipo Menu (de la clase anteriormente realizada) que nos permita ofrecer las opciones:

- *Crear un cliente*,
- *Buscar a un cliente*,
- *Mostrar los clientes leídos*,
- *Eliminar cliente de la lista*,
- y *Finalizar*.

***Nota:*** Utiliza objeto un objeto *List* para guardar los Clientes leídos.

3.- Implementa una función (static) que reciba como parámetro un numero positivo (entre 0 y 999) y que retorne un String con el número en forma de número romano.
- Ejemplo: NumeroTextual(248) retornará “CCXLVIII”.

4.- Implementa una función (static) que reciba como parámetro un numero positivo (entre 0 y 999) y que retorne un String con el número en forma textual.
- Ejemplo: NumeroTextual(258) retornará “Doscientos cincuenta y ocho”.

Dedícale un buen rato a planificar antes de ponerte a programar.
Te recomiendo que prepares una serie de tablas de datos String inicializadas con las formas textuales de los dígitos, según la posicion dentro del número.
- De este modo…

```
String[] unidades={“cero”, ”uno”, “dos”, ….
String[] decenas = {“dieci”, “veinti”, “treinta y ” ….
String[] teens = {“diez”, “once”, ….
```

5.-

Una cadena de ADN se representa como una secuencia circular de bases (adenina, timina, citosina y guanina) que es única para cada ser vivo, por ejemplo…

| | | |
|---|---|---|
| A | T | G |
| T | | C |
| A | T | G |

Dicha cadena se puede representar como un vector de caracteres recorriéndola en sentido horario desde la parte superior izquierda:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | T | G | C | G | T | A | T |

Se pide diseñar una clase que represente una secuencia de ADN que sobre escriba el método equals que devuelva true si dos cadenas de ADN coinciden.

MUY IMPORTANTE: La secuencia de ADN es cíclica, por lo que puede comenzar en cualquier posición. Por ejemplo, las dos secuencias siguientes coinciden:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | T | G | C | G | T | A | T |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | T | A | T | G | C | G | T |

Nombre de la clase: ADN
- Contructor único, recibe un String com letras (A, T, G, ó C) en máyusculas. Lanza excepcion si no es así.
- Implementar Equals y ToString, que sobreescriban adecuadamente los de Object.
- implementar estas funciones (no estáticas):
  - ADN Concatenar(ADN otraCadena)

***Nota:*** *Utiliza objetos List*

6.- Implementa las clases del siguiente diagrama para la gestión de un VideoClub.