

Informe de Trabajo Práctico

Nº1 - Parte C: Fundamentos - Herencia, agregación, polimorfismo

Programación Orientada a Objetos
Ingeniería en Mecatrónica

Alumno: Juan Manuel BORQUEZ PEREZ
Legajo: 13567



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



**FACULTAD
DE INGENIERÍA**

► **1983/2023**
40 AÑOS DE DEMOCRACIA

1 Enunciado

- a) Prepare un directorio denominado `apellido_legajo_C` (reemplace con sus datos) que contenga 2 subdirectorios denominados `cons_4` y `cons_5`.
- b) Ubique en cada uno de ellos la implementación que corresponda según la consigna.
- c) Coloque en el directorio raíz (`apellido_legajo_C`) el documento con el informe general para esta entrega (siga el mismo formato de la anterior).
- d) Al finalizar, suba un archivo comprimido de la carpeta raíz creada (incluyendo todo su contenido) en la actividad denominada Entrega C, dentro del aula virtual.
- e) En cada consigna, Usted debe diseñar un modelo orientado a objetos que incorpore las clases mínimas indicadas e informarlo mediante un diagrama de clases.
- f) Fecha de entrega: 20 de setiembre de 2023.

1.1 Consigna 4

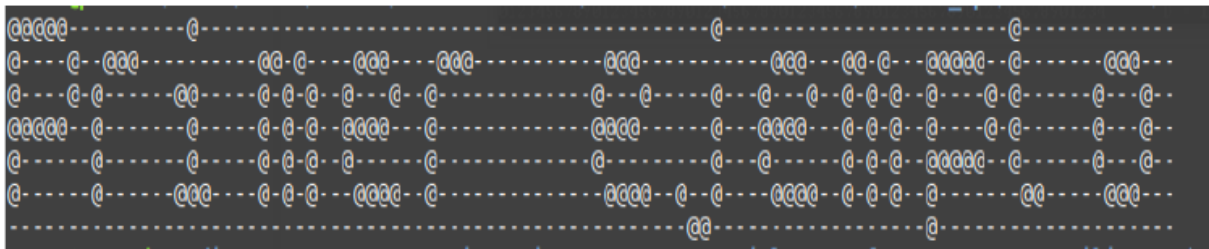
Implementar en lenguaje Python, bajo el paradigma orientado a objetos, un programa de utilidad que ofrezca un menú de usuario con la siguiente funcionalidad:

- Listar los archivos de arte ASCII disponibles en un directorio ubicado en el espacio de la aplicación.
- Seleccionar un archivo en particular mediante el tipeado de su nombre completo.
- Visualizar en la consola el contenido del archivo con el arte ASCII que contiene y un informe de análisis del mismo. Este informe debe indicar:
 - Qué caracteres se usaron en el archivo de arte ASCII.
 - Cuántos caracteres hay de cada uno y el total.
- Listar, en formato JSON, las coordenadas (solamente) que corresponden a un carácter indicado por el operador, considerando que `[0, 0]` se encuentra en la esquina superior izquierda.
- **[Desarrollo opcional]** Obtener la línea de texto del mensaje sin formato y su longitud, usando subprocesos y el binario del generador de arte ASCII.
- Terminar la ejecución.

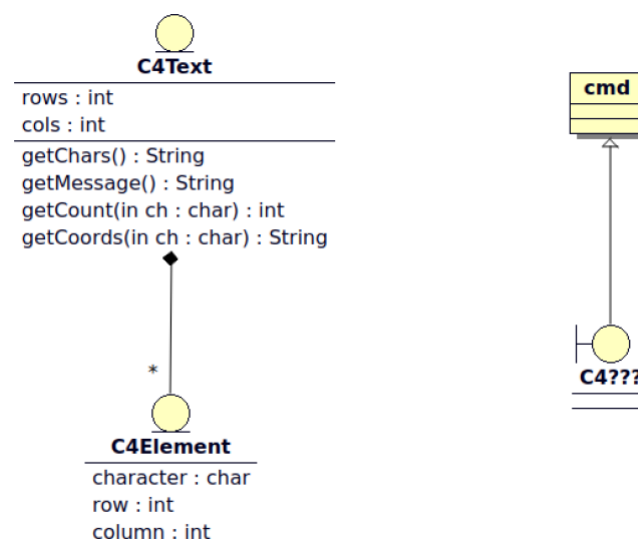
1.1.1 Observaciones y restricciones al diseño e implementación.

- Algunos archivos de prueba, para colocar dentro del directorio ya mencionado, se encuentran en el aula virtual (denominados `ejemploN.txt`), sin embargo, usted puede agregar otros si lo cree conveniente. Para ello, puede usar el generador provisto (denominado `c4ascii.cpp`).
- Los archivos de arte ASCII responden al siguiente formato interno:

- Contenido como texto plano.
- 7 filas de N caracteres ASCII, dependiendo N de la longitud de la línea con el mensaje.
- Un carácter de frente (para el texto del mensaje) y un carácter de fondo
- Por ejemplo:



- Actualmente, el generador admite 1 carácter de frente; sin embargo, se estima que su funcionalidad será ampliada para usar diferentes caracteres.
- Las opciones de visualización e informe no deben operar sobre el sistema de archivos.
- El aplicativo debe ofrecer mensajes de ayuda cuando ocurran situaciones no previstas o el usuario seleccione una opción que dependa de otra.
- Aplique un esquema de diseño que separe la capa de modelo de las de vista y control.
- Para el manejo de errores y excepciones, utilizar subclases de Exception.
- Para la interfaz de usuario, usar formato CLI y reutilización mediante herencia.
- Incluya en su modelo OO las siguientes clases, agregando aquellos elementos que considere adecuados:



1.2 Consigna 5

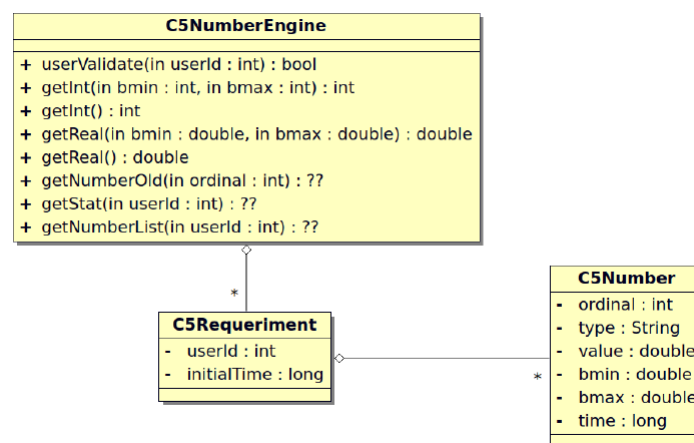
Utilizando UML para representar el modelo orientado a objetos y el lenguaje C++ para la implementación, desarrolle una pequeña herramienta para su uso en un sistema de comunicaciones en red conforme al requerimiento descripto:

- La aplicación deseada sigue una arquitectura cliente/servidor.
 - Los valores son puestos a disposición de los clientes mediante XML-RPC en un port específico para una IP dinámica, que se informa al momento del lanzamiento del servidor.
 - No se requiere una interfaz de usuario especial. Es una herramienta del tipo orden o comando del sistema(tanto para el servidor como para el cliente). Esto implica considerar el uso de argumentos de línea de comandos.
- El servidor debe proveer los siguientes servicios:
 - Validar cada petición en base a un identificador de usuario.
 - Permitir a un usuario iniciar un proceso de generación de números.
 - Generar un número real con 2 dígitos de precisión en un rango $[-CI, CS]$, donde CI y CS son los valores paramétricos de las cotas superior e inferior del rango, recibidos desde el cliente.
 - Generar un número entero en un rango $[-CI, CS]$ proporcionado por el cliente.
 - Informar el valor de un número generado por el usuario, con su marca de tiempo y el rango asociados.
 - Informar la cantidad de números generados por el usuario, su suma total y su promedio.
 - Listar los números generados por el usuario remoto desde que inició el proceso de generación, incluyendo el tipo, el instante de tiempo en que se generó, el rango de cada uno y, al final, el tiempo que llevó realizar todo el proceso desde que se hizo la primera solicitud.
- El cliente debe permitir:
 - Solicitar un número entero en un rango indicado explícitamente.
 - Solicitar un número real en un rango indicado explícitamente.
 - Solicitar un número entero sin indicar el rango. En este caso, el servidor utiliza el mismo rango de la petición entera anterior.
 - Solicitar un número real sin especificar el rango. En este caso, el servidor utiliza el mismo rango de la petición real anterior.
 - Solicitar la información asociada a un número generado anteriormente. En este caso, la petición se realiza enviando el ordinal deseado.
 - Solicitar la estadística básica.
 - Solicitar el listado de números generados.

- Mostrar en pantalla cada respuesta obtenida.
- Guardar la respuesta obtenida en un archivo con formato XML bien formado. El nombre del archivo se provee en cada petición.

1.2.1 Observaciones y restricciones al diseño e implementación.

- Si bien son pocos usuarios, cada uno se identifica por un número de 3 dígitos.
- Los identificadores de los usuarios habilitados se encuentran almacenados en un archivo de texto predefinido del lado servidor.
- Un usuario puede acceder solo a los datos generados por él.
- La marca de tiempo de generación de cada número corresponde a la cantidad de segundos desde que el usuario inició el proceso.
- Contemplar el control de errores en los escenarios básicos de operación usando excepciones.
- Incluya los mensajes al usuario que correspondan, así como ayudas de operación o notificación de errores.
- Aplique un diseño de 2 capas con separación entre el modelo y la presentación/control.
- Por simplicidad, se sugiere el uso de la librería XML-RPC para C++, disponible desde la URL <http://xmlrpcpp.sourceforge.net/>. Considere la lectura de la documentación disponible en la página del proyecto. Revise el video explicativo de clase, sobre el montaje de un aplicativo C/S usando dicha librería, en <https://youtu.be/B4vzzSq0DvA> (aproximadamente desde 4:40 minutos). Puede ver un aplicativo similar en <https://youtu.be/2an54kpvg3c>. Sin embargo, queda a su preferencia el uso de una librería similar.
- Incluya en su modelo OO las siguientes clases, además de cualquier otro elemento que considere adecuado:



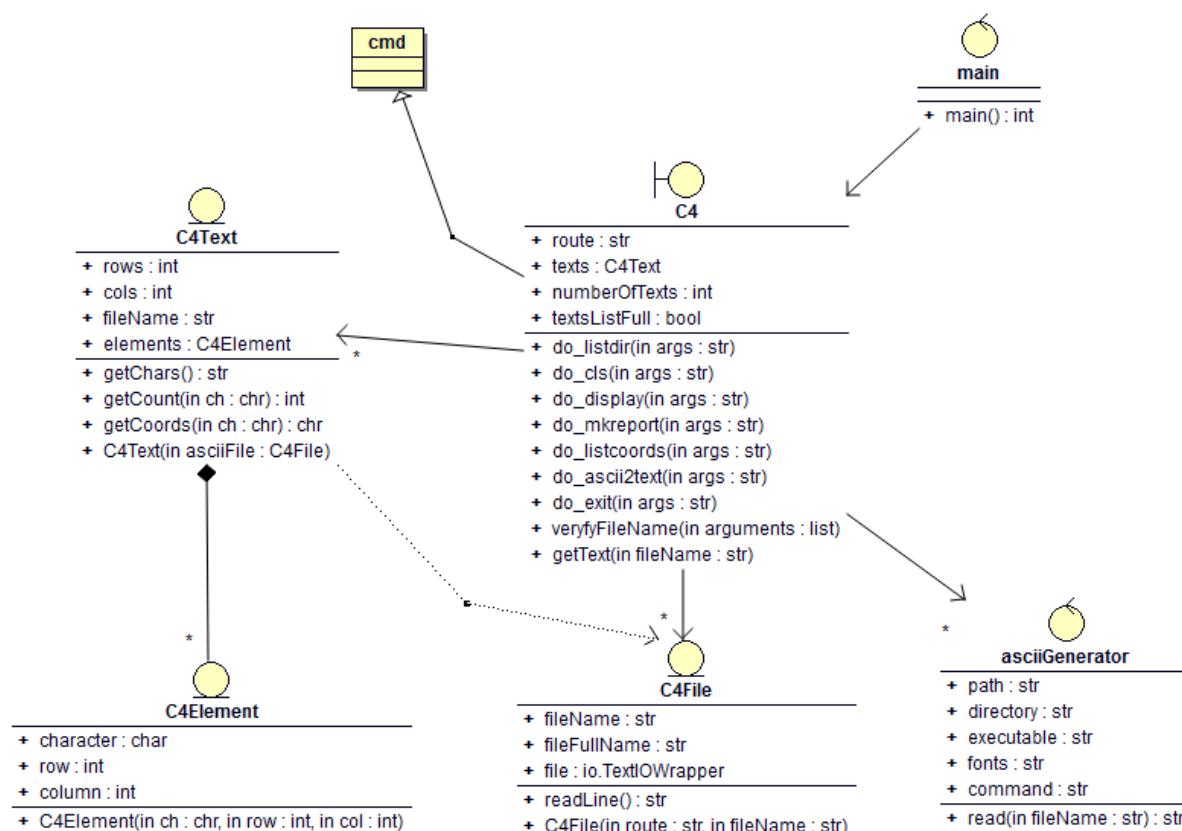
2 Esquema General de la Solución

2.1 Consigna 4

Esquema lógico general de la solución para la consigna 4.

La mayoría de la información relativa a las clases mostradas en el siguiente esquema está indicada como comentarios en los métodos y descripciones de las clases. A continuación, se realiza una descripción general de la solución:

- La clase ‘main’ representa el archivo principal de la solución.
- ‘C4’ es la principal clase de la solución que tiene los métodos CLI y permite la interacción con el usuario a través de prompts y salidas por pantalla.
- Los objetos de la clase ‘C4File’ están asociados a archivos de arte ASCII particulares. Se trata de objetos temporales utilizados únicamente durante la construcción de los objetos de la clase ‘C4Text’ para realizar la lectura de los archivos de arte ASCII.
- Los objetos de la clase ‘C4Text’ son representaciones del arte ASCII contenida en los archivos. Permiten acceder a información y operaciones generales de los mismos.
- Los objetos de la clase ‘C4Element’ son objetos persistentes contenidos dentro de objetos de la clase ‘C4Text’.
- ‘asciiGenerator’ es una clase con información y un método particular para el acceso al procedimiento remoto de la aplicación de ASCII art auxiliar. Tiene información de clase persistente común a todos los objetos de esta clase. Cada objeto se crea con el propósito de acceder al método.



2.2 Consigna 5

Esquema lógico general de la solución para la consigna 5

La mayoría de la información relativa a las clases mostradas en los siguiente esquema está indicada como comentarios en los métodos y descripciones de las clases. A continuación, se realiza una descripción general de la solución:

Se llevó a cabo un diagrama de clases para la parte del cliente y varios diagramas para la parte del servidor. Todos los diagramas y la distribución general de la aplicación, junto con las relaciones, se pueden acceder desde el proyecto de UML que se encuentra en la carpeta UML dentro de la consigna 5 en `./cons_5/C5/UML/C5`. El diagrama para la parte del cliente es menos cargado y se presenta en una sola captura en la figura 1, a la que se puede acceder desde el panel de navegación en BOUML como se indica en la captura 3.

La parte del servidor implica un conjunto de relaciones más complejas y en mayor cantidad si se tienen en cuenta todas las modificaciones realizadas. El diagrama completo para la parte del servidor no se presenta de forma legible en una sola captura (ver figura 2) y, por lo tanto, se decidió armar varios diagramas de clases para las partes más importantes del servidor. A estos diagramas se puede acceder desde el panel de navegación en UML como se indica en la captura 4.

Por un lado, en la figura 5 se encuentra el diagrama que representa las relaciones entre las clases principales dadas en la consigna. El diagrama que se muestra en las figuras 6 y 7 representa las relaciones entre clases asociadas a las excepciones personalizadas que se han definido. Las figuras 8 y 9 representan las relaciones entre clases asociadas al servidor

RPC.

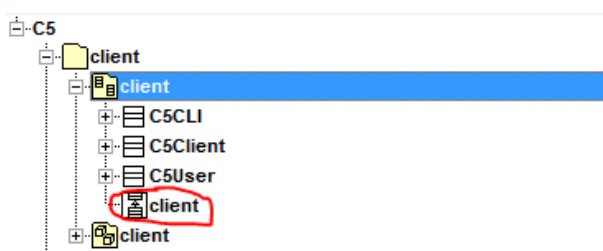


Figure 1: Diagrama de clases para la parte del cliente.

Para ver el esquema completo, consulta el Anexo ??, donde se muestran en varias fotos todos los diagramas de clases.

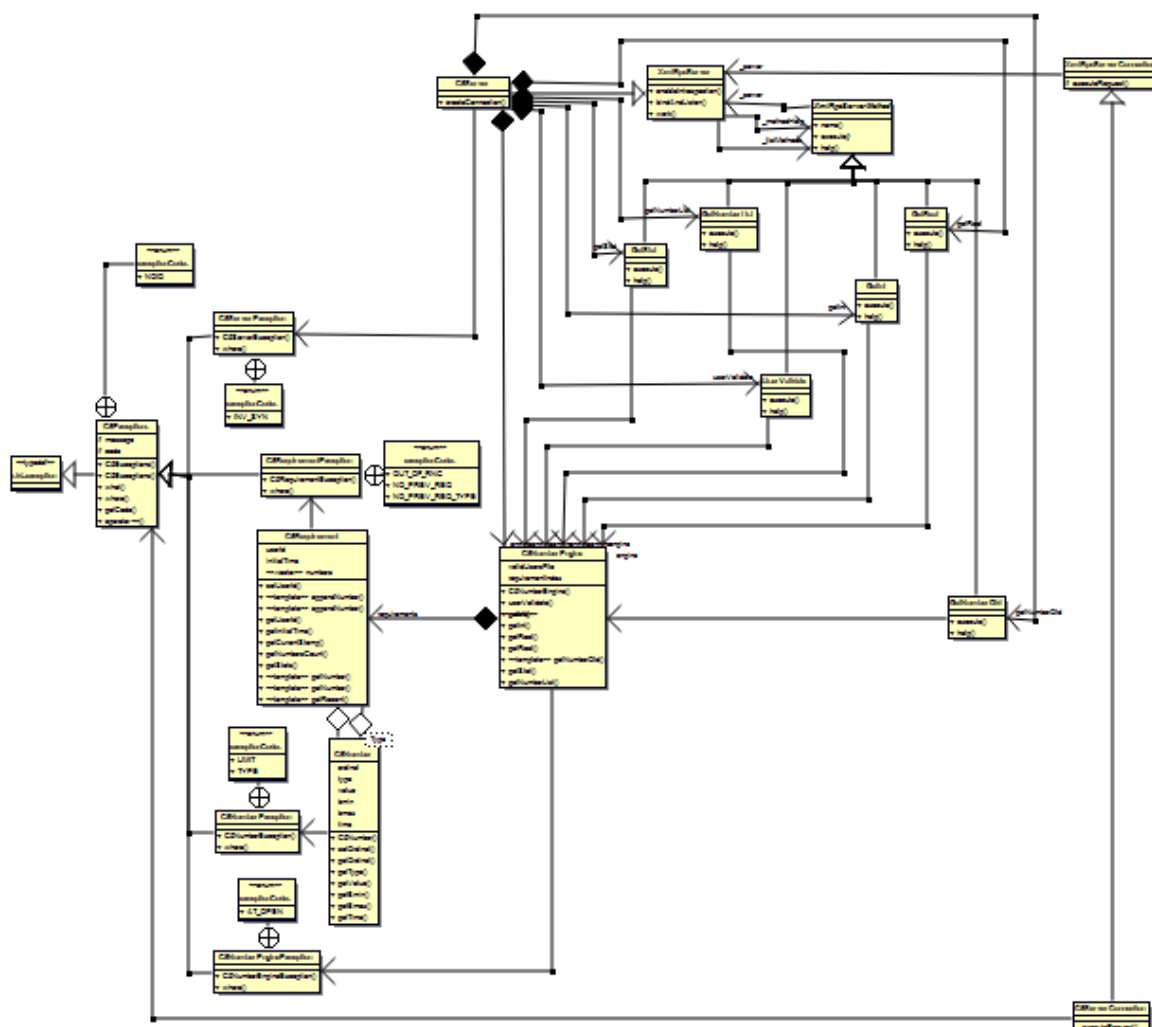


Figure 2: Diagrama completo para la parte del servidor.

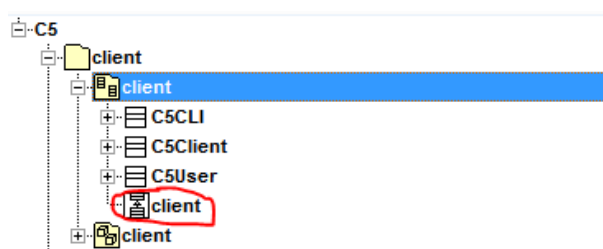


Figure 3: Captura del panel de navegación en BOUML para acceder al diagrama del cliente.

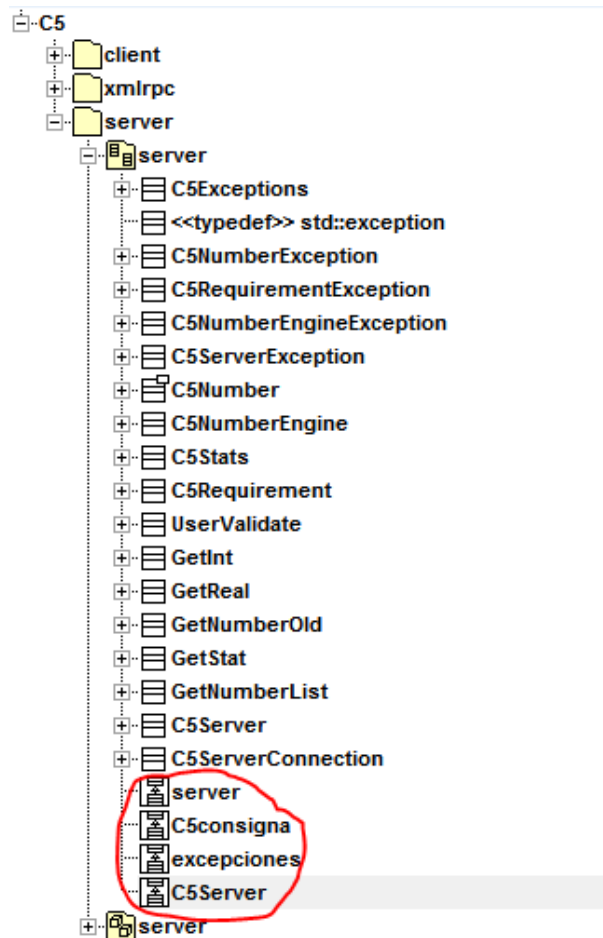


Figure 4: Captura del panel de navegación en UML para acceder a los diagramas del servidor.

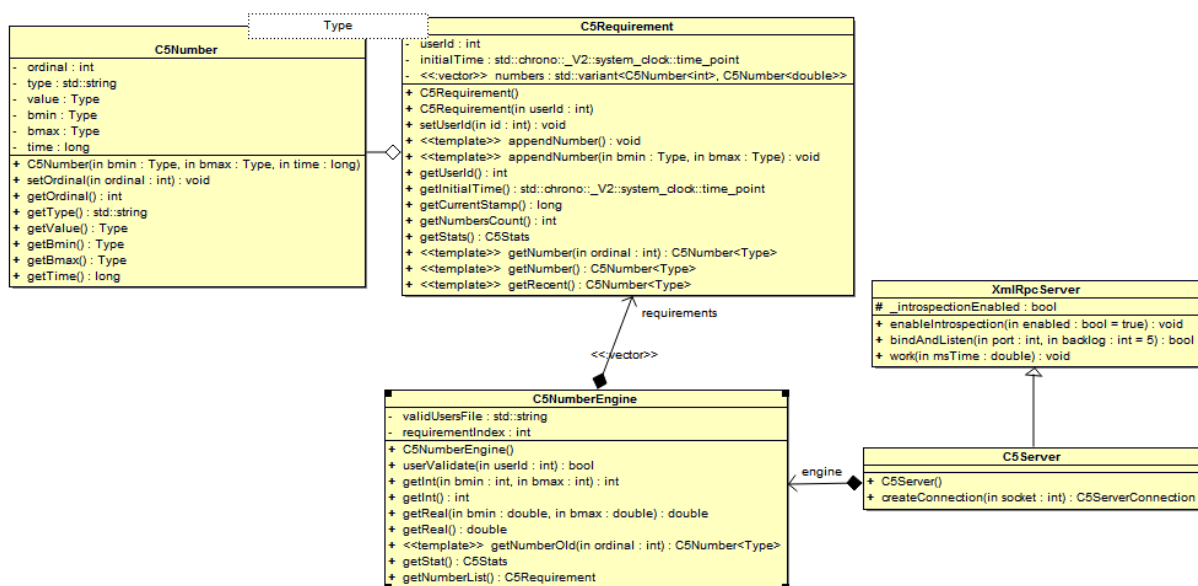


Figure 5: Diagrama de relaciones entre las clases principales.

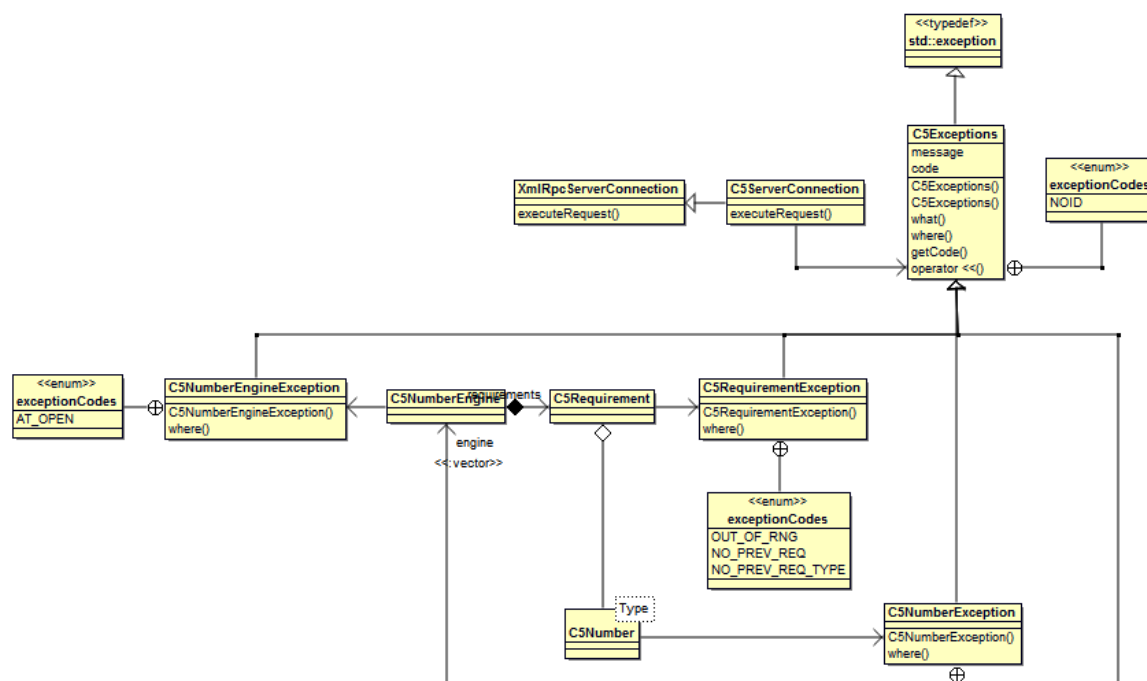


Figure 6: Diagrama de relaciones entre clases asociadas a excepciones personalizadas.

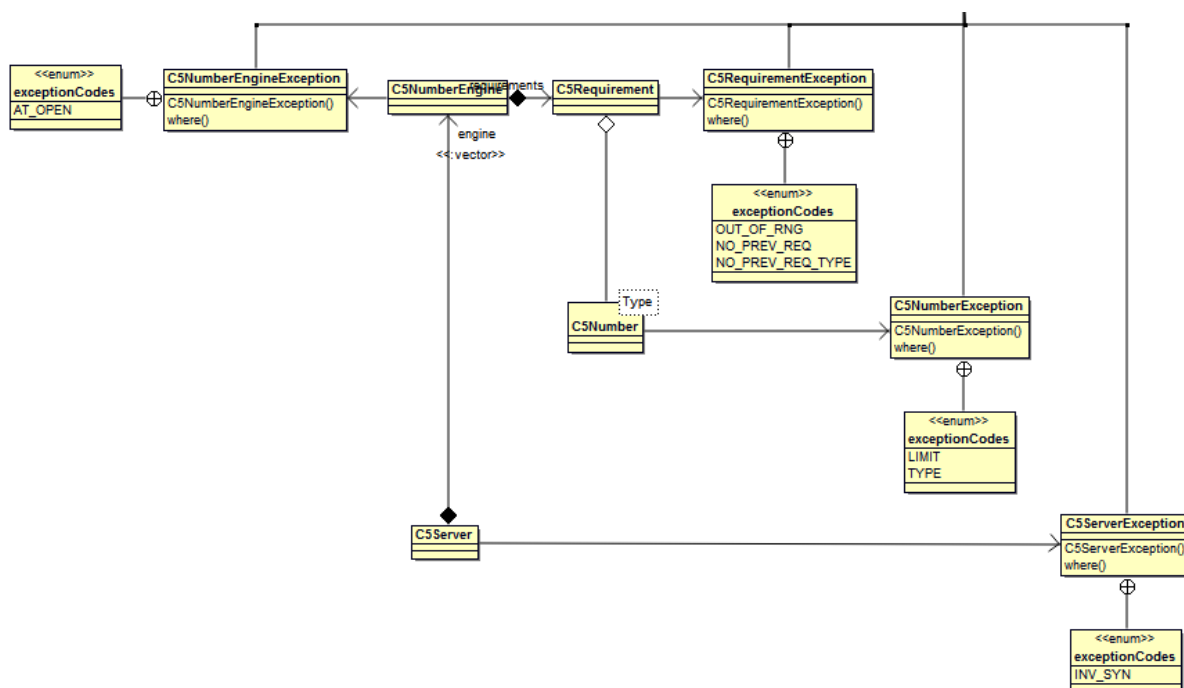


Figure 7: Diagrama de relaciones entre clases asociadas a excepciones personalizadas.

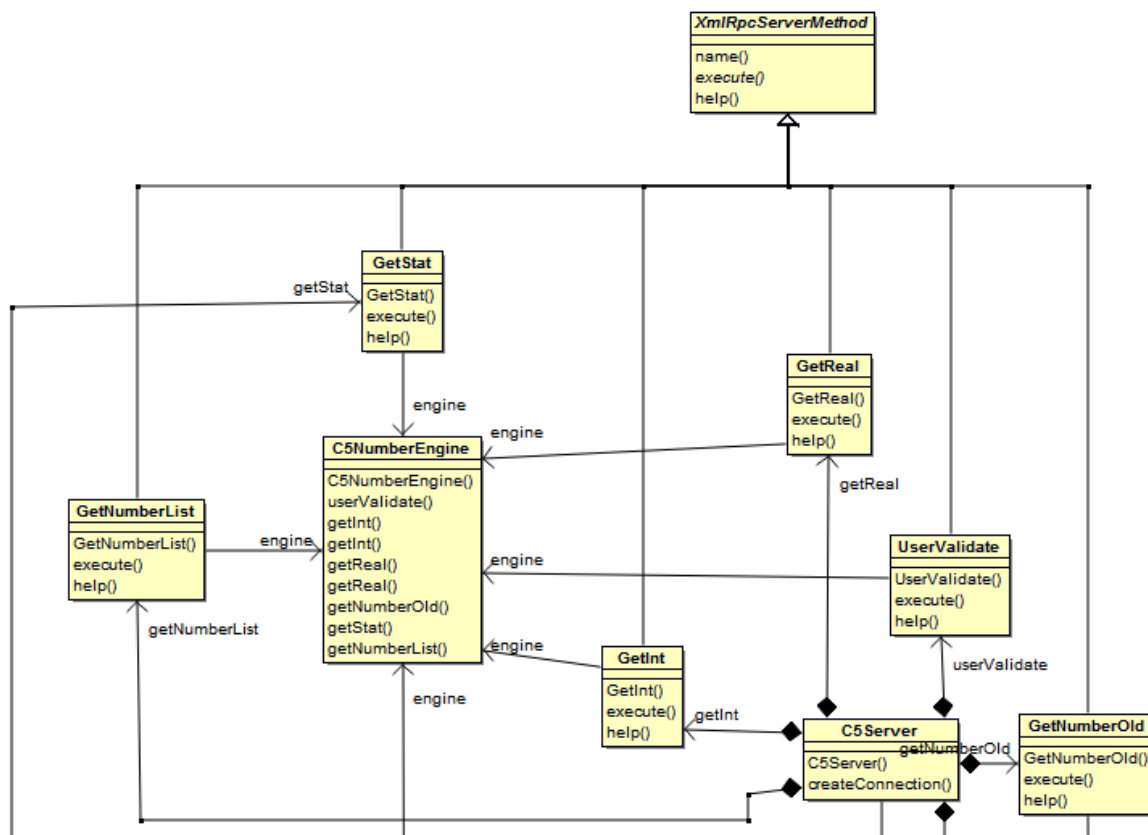


Figure 8: Diagrama de relaciones entre clases asociadas al servidor RPC.

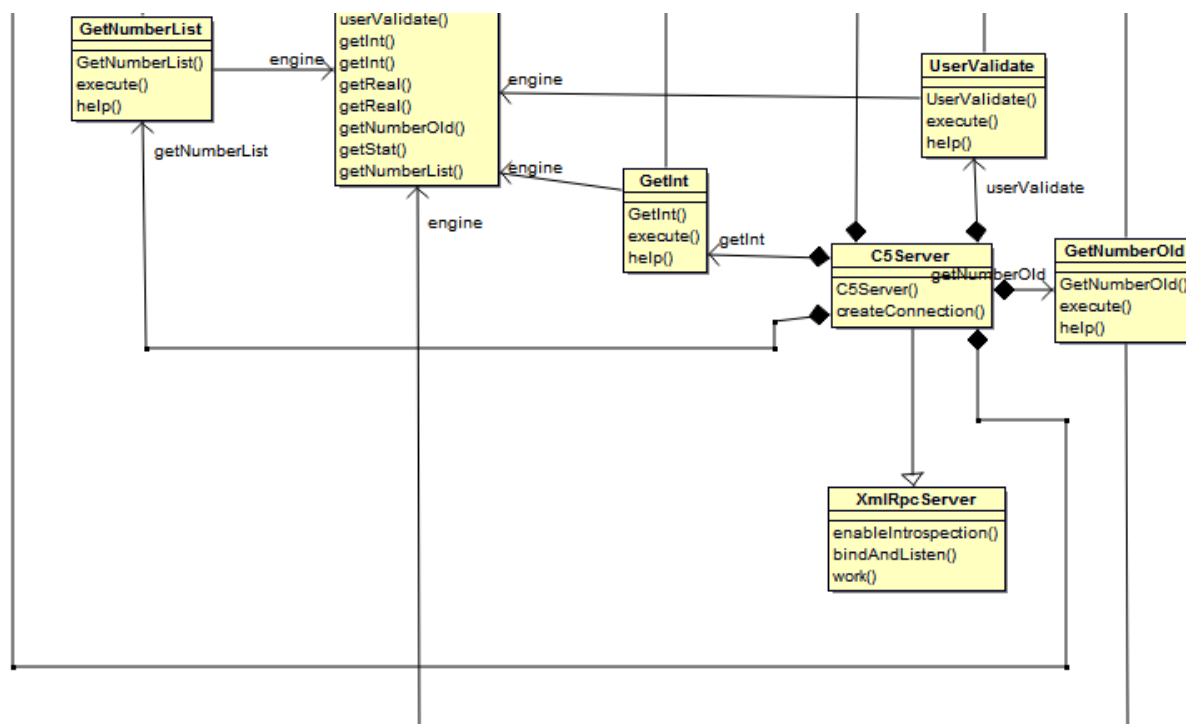
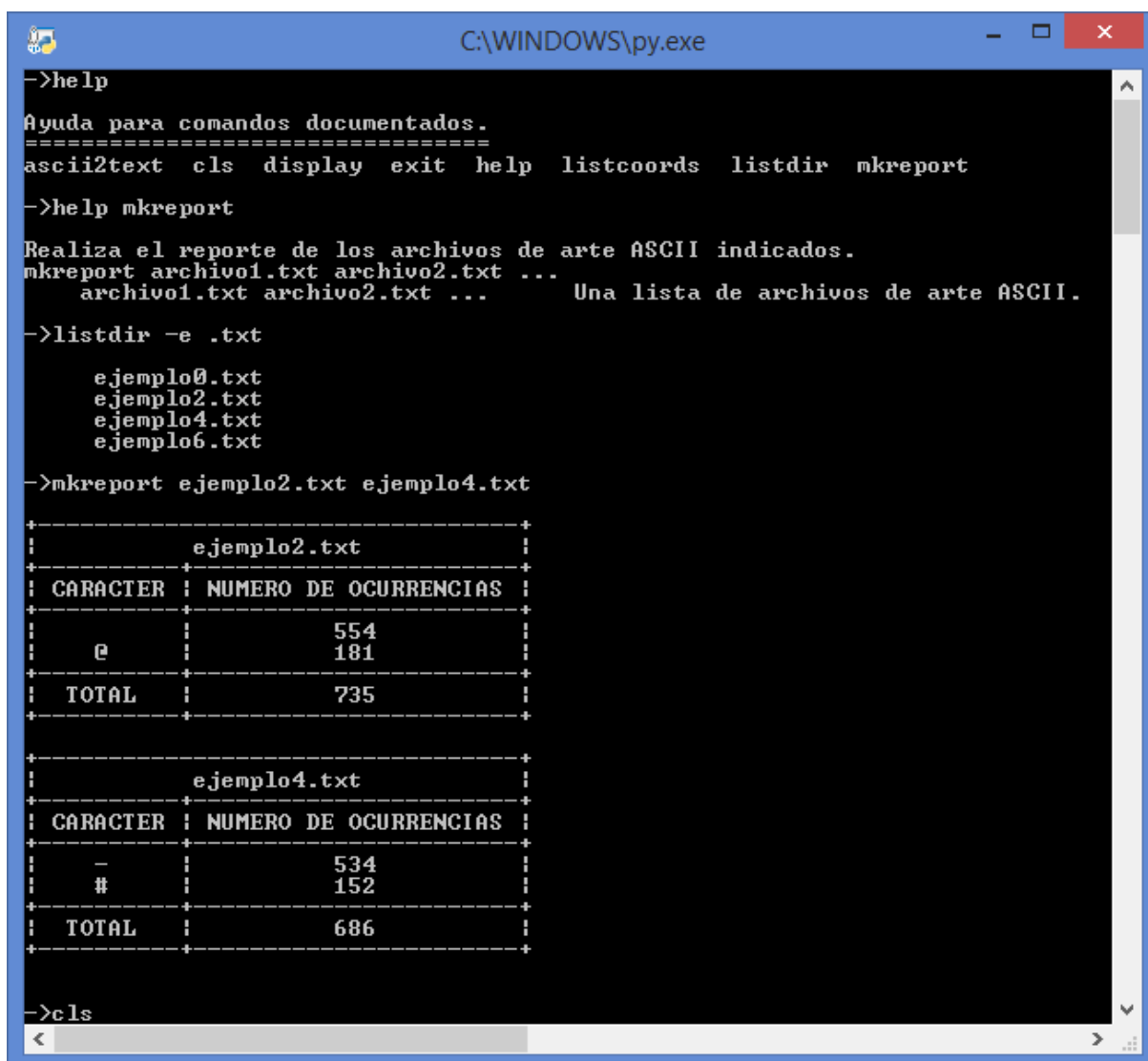


Figure 9: Diagrama de relaciones entre clases asociadas al servidor RPC.



```

C:\WINDOWS\py.exe
->help
Ayuda para comandos documentados.
=====
ascii2text  cls  display  exit  help  listcoords  listdir  mkreport
->help mkreport
Realiza el reporte de los archivos de arte ASCII indicados.
mkreport archivo1.txt archivo2.txt ...
    archivo1.txt archivo2.txt ...      Una lista de archivos de arte ASCII.
->listdir -e .txt
    ejemplo0.txt
    ejemplo2.txt
    ejemplo4.txt
    ejemplo6.txt
->mkreport ejemplo2.txt ejemplo4.txt
+-----+
|          | ejemplo2.txt          |
+-----+
| CARACTER | NUMERO DE OCURRENCIAS |
+-----+
|    @     |          554          |
|          |          181          |
+-----+
|  TOTAL   |          735          |
+-----+
+-----+
|          | ejemplo4.txt          |
+-----+
| CARACTER | NUMERO DE OCURRENCIAS |
+-----+
|    -     |          534          |
|    #     |          152          |
+-----+
|  TOTAL   |          686          |
+-----+
->cls
  
```

4 Uso

4.1 Consigna 4

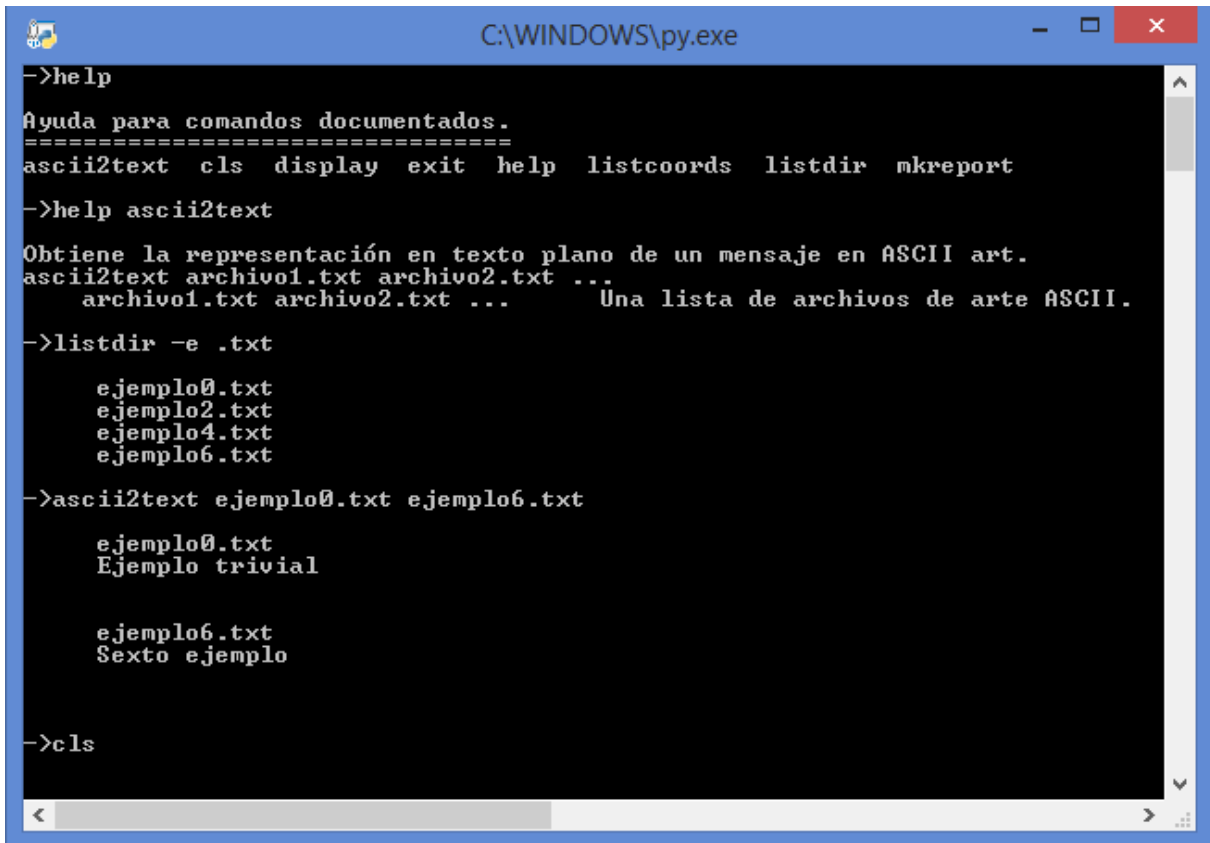
La aplicación se ejecuta abriendo el archivo "main.py".

La aplicación asociada a esta consigna permite el tratamiento general de archivos con arte ASCII. Presenta una interfaz de línea de comandos que permite al usuario llevar a cabo distintas operaciones mediante comandos.

Las operaciones que se pueden realizar con archivos de arte ASCII están documentadas. Puedes acceder a esta documentación de ayuda para cada comando utilizando el comando "help" sin parámetros, que lista todos los comandos disponibles, o utilizando el mismo comando seguido del nombre de un comando específico para acceder a ayuda específica para ese comando ("help comando").

A continuación, se indican las operaciones que se pueden llevar a cabo:

- **listdir**: Muestra los nombres de los archivos en el directorio de la aplicación. También puedes listar aquellos con una extensión específica, por ejemplo, .txt para



```

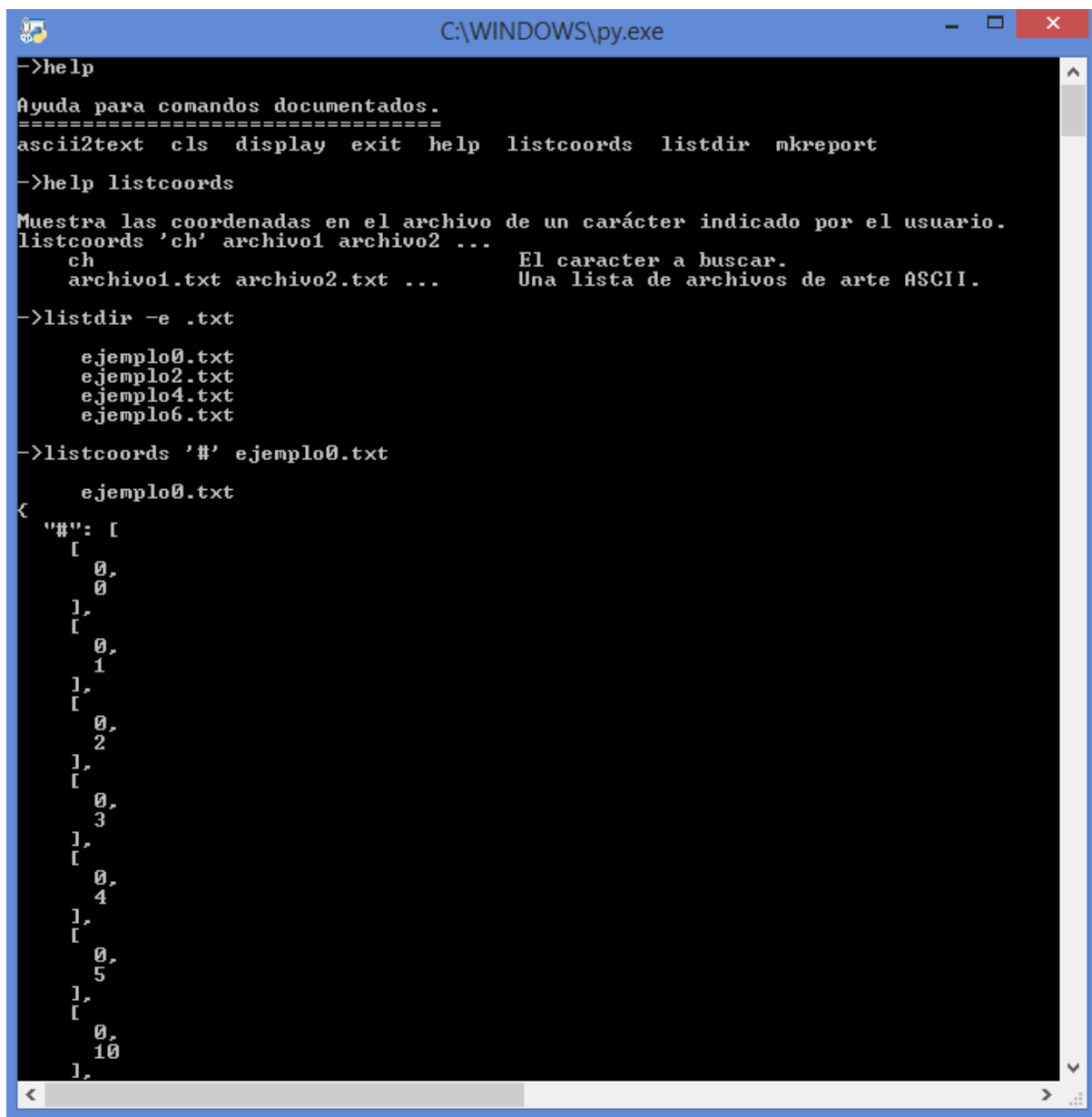
C:\WINDOWS\py.exe
->help
Ayuda para comandos documentados.
=====
ascii2text  cls  display  exit  help  listcoords  listdir  mkreport
->help ascii2text
Obtiene la representación en texto plano de un mensaje en ASCII art.
ascii2text archivo1.txt archivo2.txt ...      Una lista de archivos de arte ASCII.
    archivo1.txt archivo2.txt ...
->listdir -e .txt
    ejemplo0.txt
    ejemplo2.txt
    ejemplo4.txt
    ejemplo6.txt
->ascii2text ejemplo0.txt ejemplo6.txt
    ejemplo0.txt
    Ejemplo trivial

    ejemplo6.txt
    Sexto ejemplo
->cls
  
```

arte ASCII.

- **display**: Muestra por pantalla el contenido de una lista de archivos de arte ASCII.
- **mkreport**: Genera un reporte del contenido del archivo de arte ASCII, incluyendo los caracteres utilizados y el número de ocurrencias.
- **listcoords**: Muestra las coordenadas en las que se presenta un carácter específico dentro de un archivo de arte ASCII.
- **ascii2text**: Convierte el contenido de un archivo de arte ASCII en un mensaje en texto plano.
- **cls**: Limpia la pantalla.
- **exit**: Termina la ejecución.

Para que la aplicación funcione correctamente, es importante que todos los archivos de código se encuentren en el mismo directorio tal como se entregan. Además, el módulo auxiliar en C++ utilizado para el tratamiento de archivos de arte ASCII, el cual se usa para la traducción del contenido de arte ASCII a un mensaje en texto plano, debe estar contenido en la carpeta especificada dentro del directorio de la solución, junto con el archivo "font.txt". En resumen, la disposición de archivos en directorios debe ser la misma que se proporciona con la aplicación.



```

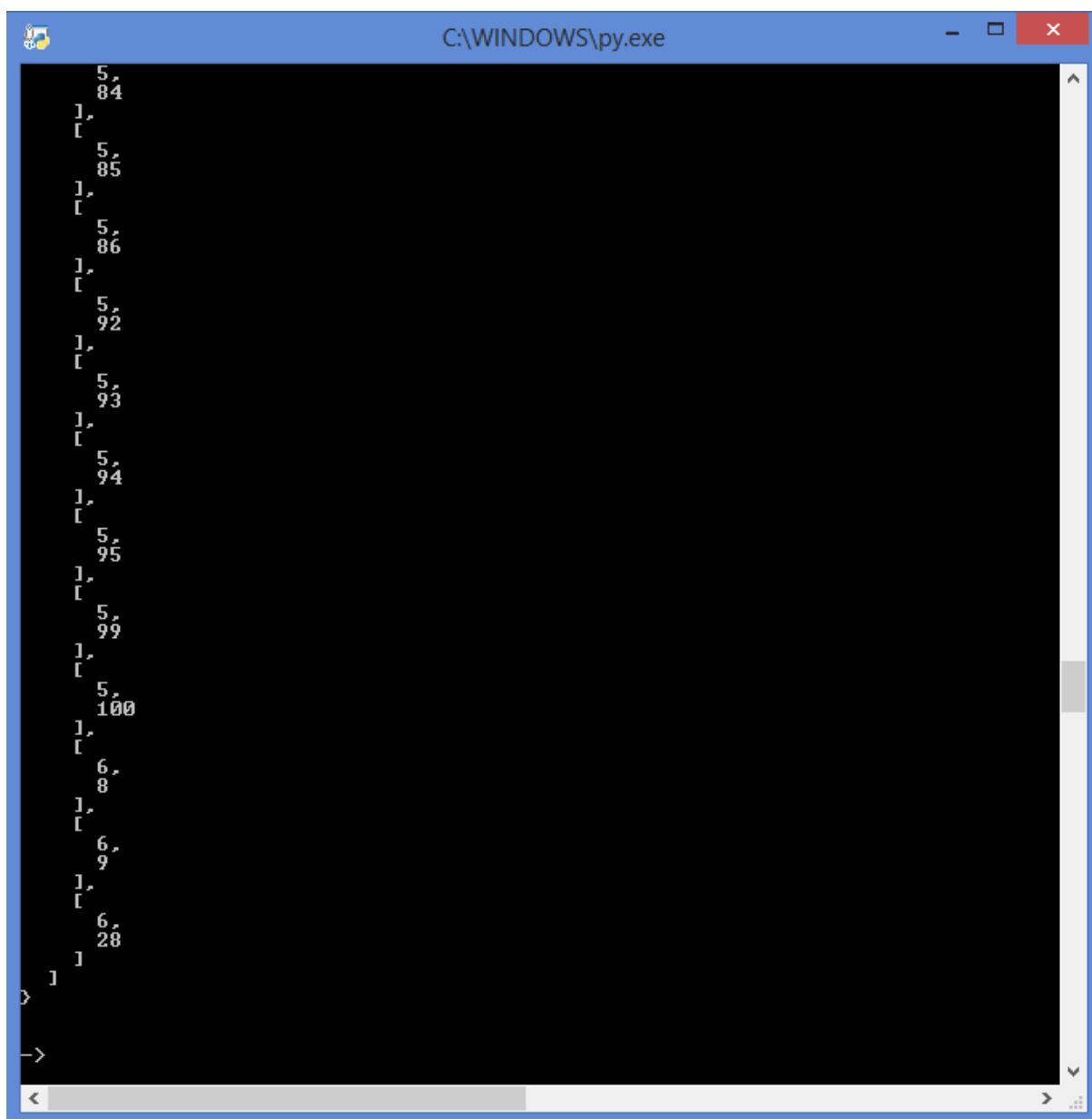
C:\WINDOWS\py.exe
->help
Ayuda para comandos documentados.
=====
ascii2text cls display exit help listcoords listdir mkreport

->help listcoords
Muestra las coordenadas en el archivo de un carácter indicado por el usuario.
listcoords 'ch' archivo1 archivo2 ...
    ch                               El caracter a buscar.
    archivo1.txt archivo2.txt ...     Una lista de archivos de arte ASCII.

->listdir -e .txt
    ejemplo0.txt
    ejemplo2.txt
    ejemplo4.txt
    ejemplo6.txt

->listcoords '#' ejemplo0.txt
    ejemplo0.txt
{
  "#": [
    [
      0,
      0
    ],
    [
      0,
      1
    ],
    [
      0,
      2
    ],
    [
      0,
      3
    ],
    [
      0,
      4
    ],
    [
      0,
      5
    ],
    [
      0,
      10
    ],
  ]
}
  
```

El manejo de errores se realiza principalmente en los métodos de comandos de la clase C4. En los demás módulos, se lanzan excepciones en casos específicos. Aprovechamos el sistema para la validación de archivos, por lo que no realizamos explícitamente una verificación de la existencia de un archivo. En su lugar, al intentar abrir un archivo que no existe, capturamos la excepción generada por el sistema.



4.2 Consigna 5

5 Conclusiones

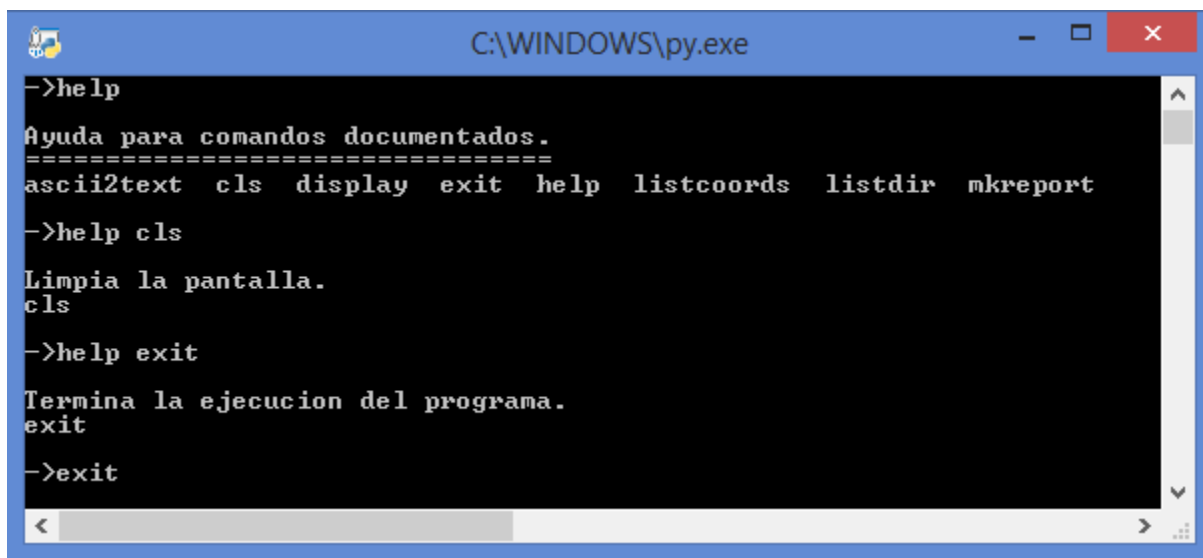
5.1 Cambios Realizados/Particularidades

5.1.1 Consigna 4

La mayoría de los cambios realizados en el código se encuentran indicados en el código fuente de la aplicación como comentarios de línea. Otros de los cambios realizados son los siguientes.

En C4Text:

Atributos:



```
->help
Ayuda para comandos documentados.
=====
ascii2text  cls  display  exit  help  listcoords  listdir  mkreport
->help cls
Limpia la pantalla.
cls
->help exit
Termina la ejecucion del programa.
exit
->exit
```

- **fileName**: El nombre del archivo de arte ASCII al que está asociado.
- **elements**: Una matriz (lista de listas) de elementos C4Elements.

Métodos:

- **getChars**: Devuelve una lista con los caracteres en el archivo en lugar de un string.
- **getCoord**: Devuelve un diccionario en el que la clave es el carácter indicado y el valor es una lista con las coordenadas (tuplas) de ocurrencia del carácter.
- **C4Text**: El constructor para objetos de la clase recibe el nombre de un archivo de arte ASCII. En el constructor se obtiene la lista de elementos asociados al objeto.
- **str**: Permite la obtener la representación del objeto en una cadena de caracteres.

En C4:

- No se dispone de comandos específicos para seleccionar archivos o conjuntos de archivos en esta clase. En su lugar, los comandos pueden aplicarse en cualquier momento a cualquiera de los archivos de arte ASCII que se encuentran en el directorio. Esto se hizo para hacer más cómodo para el usuario el uso de la aplicación, en lugar de requerir una selección previa de archivos para operar.
- El manejo de errores durante la ejecución se aplica en los métodos de esta clase, aunque se limita a mostrar mensajes de error en relación a las excepciones sin realizar, por ejemplo, operaciones de seguimiento de errores.
- Para reducir la cantidad de veces que se abre un archivo de arte ASCII y se crea el objeto C4Text asociado, se incluye como atributo de esta clase una colección (lista) de objetos C4Text creados durante las llamadas a comandos. De esta manera, cuando un comando requiere la creación de un objeto C4Text para un archivo que se abrió recientemente, en lugar de crear un nuevo objeto asociado a ese archivo, se accede directamente a él desde la lista si está presente o se crea si no lo está. La

lista tiene un tamaño máximo de 5 elementos. Cuando se llena, se agrega un nuevo elemento al final de la lista y se elimina el primero. Para identificar la asociación de objetos C4Text con archivos de arte ASCII y poder recuperarlos de la lista, se incluyó como atributo adicional el nombre del archivo al que están asociados en los objetos de la clase C4Text.

- Algunos comandos, como el de limpieza de pantalla y el de visualización de archivos, hacen uso de comandos específicos del sistema. Dado que la aplicación puede ejecutarse en diferentes sistemas operativos, se verifica la plataforma antes de ejecutar esos comandos. En principio, la funcionalidad debería estar disponible en Windows, Linux y macOS.

5.1.2 Consigna 5

En C3Report:

Atributos:

- **fileroute:** La ruta de la carpeta que

En C3File:

Atributos:

- **file:** El archivo en sí o un manejador

Métodos:

- Los últimos tres métodos que se indican en el diagrama lógico para la consigna 3 se utilizan para obtener las líneas de borde superior, inferior y las que separan los campos de la tabla de las filas con datos.

5.2 Dificultades Encontradas

5.3 Comentario/Extensiones

En la **consigna 4** se podría agregar un comando más que permita la operación de generación de un archivo de arte ASCII a partir de un mensaje dado por el usuario, lo que se lograría llamando a otro método de la aplicación auxiliar.

En cuanto al manejo de excepciones, este se limita a mostrar en pantalla mensajes de error asociados a las excepciones. Se podría realizar un proceso de traceback de las excepciones con el objeto de obtener información más relevante o más completa. Se vio que tener esta información es especialmente útil durante el proceso de desarrollo. En principio, no se sabe si se ha logrado un adecuado manejo de las excepciones.

5.4 Valoración Personal/Observaciones

Considero que he logrado cumplir con los requisitos establecidos en las consignas aplicando los conceptos de Programación Orientada a Objetos. Entre los aspectos que destacaría:

- En ambas soluciones, se podría mejorar la gestión de las excepciones que puedan surgir durante la ejecución del programa.

- Es importante profundizar en la comprensión de las relaciones entre clases a través de la práctica.
- Se requiere un mejor entendimiento de los estereotipos de clases, como control, interfaz y entidad, así como su implementación.

[7], [12], [9], [11], [4], [5], [2], [3], [6], [8], [10], [14], [13], [1]