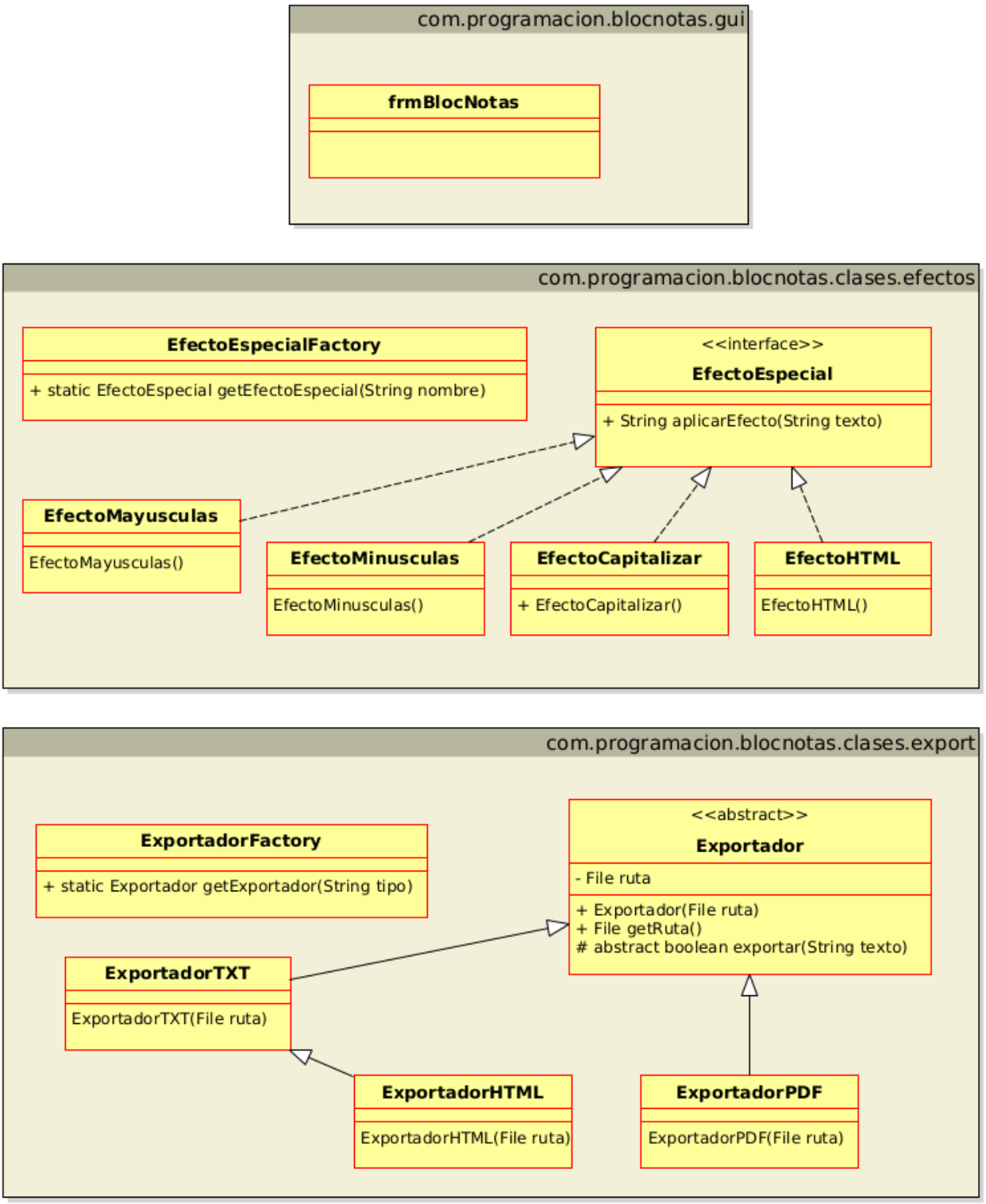
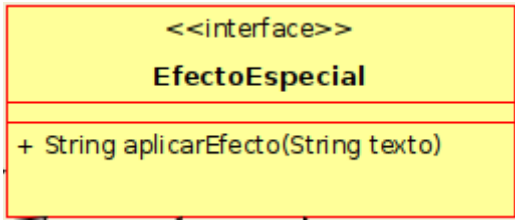
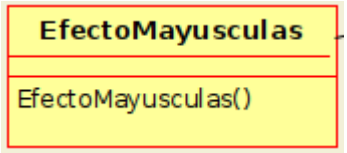
PROYECTO 4: EL BLOC DE NOTAS



***) Observa que hay clases con el <u>modificador de acceso por defecto</u>**



Esta interfaz representa un efecto especial que podemos aplicar a un String. Lo que hace el efecto estará determinado por las clases que implementen la interfaz.

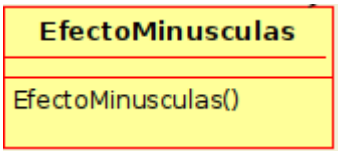


Esta clase es un efecto especial que pasa a mayúsculas un texto.

- Constructor: no hace nada
- aplicarEfecto: devuelve el texto pasado como parámetro escrito en mayúsculas

Test

1. Comprueba que si aplicas el efecto a “hola, ¿cómo estás?” se devuelve “HOLA, ¿CÓMO ESTÁS?”

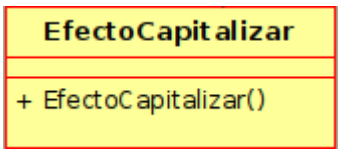


Esta clase es un efecto especial que pasa a minúsculas un texto.

- Constructor: no hace nada
- aplicarEfecto: devuelve el texto pasado como parámetro escrito en minúsculas

Test

1. Comprueba que si aplicas el efecto a “HOLA, ¿CÓMO ESTÁS?” se devuelve “hola, ¿cómo estás?”

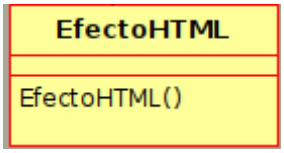


Esta clase es un efecto especial que “capitaliza” todas las palabras del texto que haya tras un punto.

- Constructor: no hace nada
- aplicarEfecto: devuelve el texto pasado como parámetro capitalizado. Esto significa que:
  - La primera letra del texto deberá estar escrita en mayúsculas.
  - Tras cada punto, la siguiente letra deberá estar escrita en mayúsculas.

Test

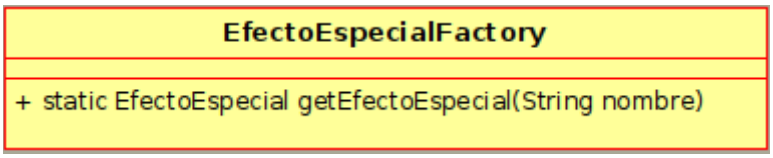
1. Comprueba que si aplicas el efecto a “esto es un texto de prueba” se devuelve “Esto es un texto de prueba”
2. Comprueba que si aplicas el efecto a “esto es un texto de prueba. aquí tenemos otro texto de prueba. Esto es el último texto de prueba” se devuelve “Esto es un texto de prueba. Aquí tenemos otro texto de prueba. Esto es el último texto de prueba”
3. Comprueba que si aplicas el efecto a “esto es un texto de prueba.aquí tenemos otro texto de prueba” (observa que no hay espacio entre el punto) se devuelve “esto es un texto de prueba.Aquí tenemos otro texto de prueba”
4. Comprueba que si aplicas el efecto a “esto es un texto de prueba. aquí tenemos otro texto de prueba” (observa que hay 6 espacios tras el punto) se devuelve “esto es un texto de prueba. Aquí tenemos otro texto de prueba”;
5. Comprueba que si aplicas el efecto a “esto es un texto de prueba.” (observa que termina en punto) se devuelve “Esto es un texto de prueba.”



Esta clase es un efecto especial que añade una cabecera HTML al texto y reemplaza los saltos de línea por etiquetas

- Constructor: no hace nada
- aplicarEfecto: Crea un String que tenga todo este contenido:
  - El String comienza como si fuese una página web: doctype, head, etc.
  - Tras escribir la etiqueta <body>, se escribe el String del texto, sustituyendo cada salto de línea (“\n”) por una etiqueta
  - Por último, se añade la etiqueta de cierre </body> y </html>

No se harán test de la clase. El funcionamiento correcto se comprobará usando el programa cuando esté lo suficientemente avanzado.



Esta clase sirve para proporcionar al programa efectos especiales. Observa que el programa no puede crear directamente objectos de los efectos especiales porque sus constructores tienen el modificador de acceso por defecto y están en un paquete diferente.

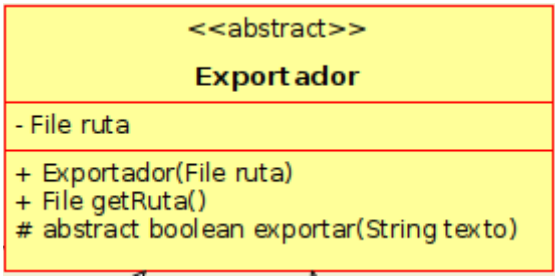
- getEfectoEspecial: Devuelve un objeto del tipo que indica la tabla según el nombre que se pasa como parámetro:

Nombre pasado como parámetro	Objecto que hay que crear y devolver
mayusculas	EfectoMayusculas
minusculas	EfectoMinusculas
capitalizar	EfectoCapitalizar
html	EfectoHTML

Si el nombre pasado como parámetro no está en la tabla anterior, se lanzará una IllegalArgumentException con el mensaje “El tipo (añadir aquí el nombre pasado como parámetro) no está disponible”.

Test

1. Usa la clase EfectoEspecialFactory para obtener un efecto de tipo “mayusculas”. Aplica ese efecto a la frase “hola, estamos trabajando” y comprueba que se obtiene “HOLA, ESTAMOS TRABAJANDO”.
2. Usa la clase EfectoEspecialFactory para obtener un efecto de tipo “pepe” y comprueba que se lanza una IllegalArgumentException.



Esta clase representa un proceso genérico que sirve para exportar un texto a un formato determinado, como txt, pdf, etc. Cada clase hija servirá para exportar un texto a un formato concreto.

Propiedades

- ruta: Es la ruta del archivo donde se guardará el texto exportado al formato correspondiente de la clase hija.

Métodos

- constructor: Inicializa la propiedad ruta
- getRuta: Devuelve la ruta
- exportar: Será programado en las clases hijas. Allí cada clase hija hará lo necesario para guardar el texto pasado como parámetro en el formato correspondiente a la clase hija.

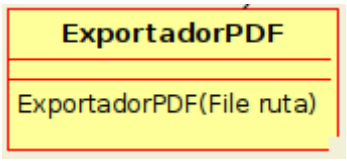
No son necesarios test

Esta clase sirve para exportar un texto a formato txt.

- Constructor: llama al constructor del padre para inicializar la ruta
- exportar: Realiza estos pasos:
  - Crea un PrintWriter para escribir en el archivo cuya ruta se hereda de la clase padre.
  - Escribe el texto pasado como parámetro
  - Cierra el archivo
  - Si en cualquier momento se produce un error, el método devolverá false. En caso contrario, devolverá true

Test

1. Usa un ExportadorTXT para guardar en un archivo una frase. Comprueba que el archivo se ha creado y que la frase está correctamente guardada en él.



Esta clase sirve para exportar un texto a formato pdf.

- Constructor: llama al constructor del padre para inicializar la ruta
- exportar: Realiza estos pasos:
  - Usa la librería gnujpdf para crear un documento pdf que tenga escrito el texto pasado como parámetro.
  - No olvides cerrar el archivo de texto
  - Si en cualquier momento se produce un error, el método devolverá false. En caso contrario, devolverá true

  Mejora voluntaria: Por defecto, la librería gnujpdf no tiene en cuenta los saltos de línea y el texto se saldrá de la hoja si es muy largo. Para solucionarlo, deberás hacer esto:
  - Usa el método split del texto para "partirlo" en frases. Deberás obtener una lista String[] con todas las frases del String.
  - Crea una variable "altura" que indicará la altura de la página donde comenzarás a escribir las líneas. Inicialmente pon la altura=12.
  - Recorre la lista de frases y escribe cada una en la hoja usando como altura la variable anterior. Tras escribir cada frase, incrementa 12 píxeles la altura.

Test

1. Usa un ExportadorPDF para guardar en un archivo una frase. Comprueba que el archivo se ha creado y que la frase está correctamente guardada en él.

Esta clase sirve para exportar un texto a formato html.

- Constructor: llama al constructor del padre para inicializar la ruta
- exportar: Realiza estos pasos:
  - Obtén un efecto especial que permita obtener la versión html del texto
  - Llama al método heredado exportar de la clase ExportadorTXT para exportar a un archivo la versión html que has obtenido anteriormente.
  - Si en cualquier momento se produce un error, el método devolverá false. En caso contrario, devolverá true

Test

1. Usa un ExportadorHTML para guardar en un archivo una frase. Comprueba que el archivo se ha creado y que la frase está correctamente guardada en él.

Esta clase sirve para obtener un exportador que sepa guardar datos en el archivo pasado como parámetro según sea su extensión. Por ejemplo, si el archivo tiene extensión .txt, obtendrá un ExportadorTXT y si el archivo tiene extensión .pdf, devolverá un ExportadorPDF

- getExportador: Primero obtiene la extensión del archivo pasado como parámetro (la extensión es el texto que hay en el nombre del archivo después de su último punto). A continuación devuelve un objeto del tipo que indica la tabla según la extensión obtenida

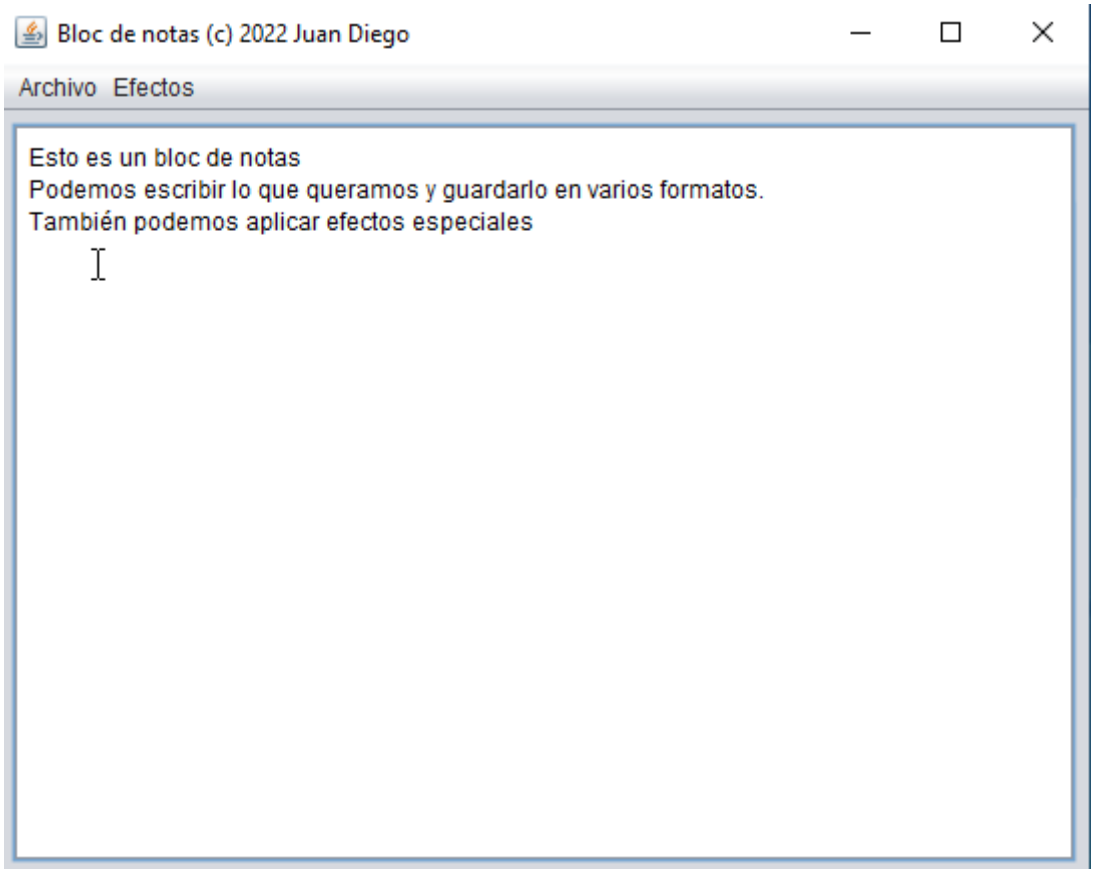
Extensión obtenida	Objecto que hay que crear y devolver
txt	ExportadorTXT
pdf	ExportadorPDF
html	ExportadorHTML

Si la extensión del archivo no está en la tabla anterior, se lanzará una IllegalArgumentException con el mensaje “La extensión (añadir aquí el nombre pasado como parámetro) no está disponible”.

Test

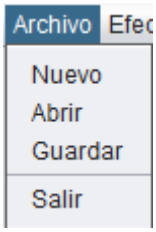
1. Usa la clase Exportador para obtener un exportador que sepa trabajar con el archivo “prueba.pdf”. Usa instanceof para comprobar que el objeto obtenido es de tipo ExportadorPDF
2. Usa la clase EfectoEspecialFactory para obtener un exportador que sepa trabajar con el archivo “prueba.mp3” y comprueba que se lanza una IllegalArgumentException

Esta clase define la ventana principal del programa:



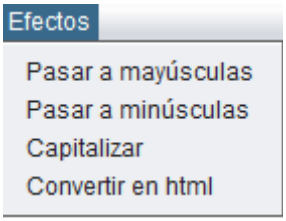
<u>Instrucciones para diseñar la ventana:</u>

1. Crea la ventana frmBlocNotas
2. Añade un JMenuBar y renombra las opciones “File” y “Edit” por “Archivo” y “Efectos”
3. Pulsa el botón derecho en la opción “Archivo” y luego “Add from palette” > “Menu Item” para ir añadiendo estas opciones:



Usa estos nombres de variable para las opciones: opcNuevo, opcAbrir, opcGuardar y opcSalir

4. Haz lo mismo con “Efectos” para añadir estas opciones:



   Usa estos nombres de variables para cada opción: opcMayusculas, opcMinusculas, opcCapitalizar y opcHtml

5. Añade un JTextArea y haz que ocupe toda la pantalla. Configúralo de esta forma:
   - Cambia su nombre de variable a txtTexto
   - Activa la casilla de su propiedad lineWrap (esto es para que el texto baje de renglón al llegar al final y no salgan barras de desplazamiento hacia la derecha)
   - Activa la casilla de su propiedad wrapStyleWord (esto es para que las palabras no se partan cuando se llega al final del reglón y bajen enteras de renglón)

Instrucciones para programar los eventos

- Objeto opcNuevo, evento ActionPerformed:
  1. Borra el contenido del text area.

- Objeto opcAbrir, evento ActionPerformed:
  1. Crea y muestra un JFileChooser con su método showOpenDialog
  2. Si el usuario selecciona un archivo y pulsa el botón de aceptar, crea un Scanner que permita leer dicho archivo
  3. Lee todas las líneas del archivo y añádelas al text area.

- Objeto opcGuardar, eventoActionPerformed:
  1. Crea y muestra un JFileChooser con su método showSaveDialog
  2. Si el usuario selecciona un archivo y pulsa el botón de aceptar, obtén un objeto File con el archivo seleccionado.
  3. Usa la clase ExportadorFactory para obtener el objeto Exportador que sabe guardar el texto en el formato apropiado para el archivo seleccionado.
  4. Usa el Exportador obtenido anterior para guardar el archivo.
  5. Si se produce un error, muestra una ventana emergente con el mensaje “Error al guardar el archivo”

- Objeto opcSalir, eventoActionPerformed:
  1. Llama al método heredado “dispose” para que se cierre la ventana que estás programando

- Objeto opcMayusculas, evento ActionPerformed:
  1. Obtén un efecto especial para pasar a mayúsculas y úsalo para convertir el texto del text area en mayúsculas. Reemplaza el texto del text area por su versión en mayúsculas.

- Objeto opcMinusculas, evento ActionPerformed:
  1. Obtén un efecto especial para pasar a minúsculas y úsalo para convertir el texto del text area en minúsculas. Reemplaza el texto del text area por su versión en minúsculas.

- Objeto opcCapitalizar, evento ActionPerformed:
  1. Obtén un efecto especial para capitalizar y úsalo para capitalizar el texto del text area. Reemplaza el texto del text area por su versión capitalizada.

- Objeto opcHtml, evento ActionPerformed:
  1. Obtén un efecto especial para convertir en html el texto y úsalo para convertir el texto del text area en html. Reemplaza el texto del text area por su versión en html.