Simulacro de examen: editor de texto

Alumno: _____

Introducción

Debéis entrar en el campus virtual y descargar el fichero comprimido con el material necesario para el examen. En él encontraréis un proyecto que debéis importar a Eclipse, y que consiste en una implementación de un editor de texto básico.

Nuestro editor permite **abrir ficheros**, **insertar texto**, **borrar la última palabra** y **reemplazar texto**.

A continuación se muestra un ejemplo de funcionamiento:

```
> abre quijote.txt
En un lugar de la mancha de cuyo nombre

> inserta no quiero acordarme
En un lugar de la mancha de cuyo nombre no quiero acordarme

> borra
En un lugar de la mancha de cuyo nombre no quiero

> reemplaza a x
En un lugxr de lx mxnchx de cuyo nombre no quiero
```

Tareas a realizar

Se desea añadir la posibilidad de **grabar macros** de tal manera que se simplifique la tarea de tener que escribir las mismas acciones varias veces.

```
> graba m1
> borra
> inserta final
> reemplaza i y
> para
```

```
> abre quijote.txt
En un lugar de la mancha de cuyo nombre
> ejecuta m1
En un lugar de la mancha de cuyo fynal

> abre prueba.txt
hola a todos
> ejecuta m1
hola a fynal
```

Debéis modificar el código entregado para que realice la tarea descrita.

Aspectos a destacar

- Si durante la grabación se ejecuta otra acción de grabación de macro («graba») la grabación anterior se omite y empieza una nueva.
- Durante la grabación de una macro puede utilizarse la acción «ejecuta» para ejecutar una macro previamente grabada (ver prueba de funcionamiento más adelante).

Evaluación

Es requisito obligatorio para aprobar (pero no suficiente) que la práctica funcione correctamente (sin errores de compilación ni ejecución). Sin embargo lo que decidirá la nota es el diseño realizado por el alumno.

Se puede utilizar la prueba de funcionamiento incluida al final del documento para comprobar su correcto funcionamiento.

Es condición imprescindible señalar correctamente y sin lugar a equívocos qué patrón o patrones de diseño se han utilizado en su resolución, **indicando para cada uno los distintos participantes** (qué clase desempeña cada papel del patrón, según la estructura del mismo vista en teoría) **así como los métodos relevantes del patrón** (qué métodos de dichas clases se corresponden con los métodos de la estructura genérica del patrón de que se trate). *Para ello se incluirá un fichero de texto con un formato como el siguiente* (se ha tomado como ejemplo el caso de los sensores de seguridad explicado al hablar del patrón *Observer*):

```
Patrón utilizado: Observer
Participantes
-------------
1) Subject –> SecurityNotifier
   Métodos:
    - Attach(Observer) –> addObserver(SecurityObserver)
    - Detach(Observer) –> removeObserver(SecurityObserver)
    - Notify() –> updateObservers()
2) Observer –> SecurityObserver
   Métodos:
    - Update() –> notify(device: int, event: int)
3) ConcreteSubject –> SecurityMonitor, SecurityClient
   Métodos:
    - getState() –> getValue()
    - setState() –> -

Patrón utilizado: Adapter
Participantes
-------------
1) Target –> SecurityObserver
   Métodos:
    - Request() –> notify(device: int, event: int)
2) Adaptee –> SecurityMonitor
   Métodos:
    - SpecificRequest() –> showAlarm(event: int, source: Object)
3) Adapter –> SecurityAdapter
    - Request() –> notify(device: int, event: int)
```

Así mismo, se deberá entregar, en papel, junto con el enunciado del examen, un **diagrama de clases UML** representando el diseño final, donde figuren todas las clases junto con sus relaciones y métodos principales.

Prueba de funcionamiento

```
> graba m1
> reemplaza a x
> inserta aa
> para

> graba m2
> inserta bbb
> ejecuta m1
> para
```

```
> abre quijote.txt
> ejecuta m2
En un lugxr de lx mxnchx de cuyo nombre bbb aa
```