

**UNIVERSIDAD POLITÉCNICA DE MADRID**

**ESCUELA TÉCNICA SUPERIOR DE**

**INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN**

PROGRAMACIÓN II

Práctica 6 Interfaces y excepciones

***Curso 2020 – 2021***

**ÍNDICE**

# [OBJETIVOS Y ALCANCE DE LA PRÁCTICA . - 4 -](#_bookmark0)

[**PRERREQUISITOS. ................................................................................................................ - 4 -**](#_bookmark1)

# [DESCRIPCIÓN DE LA APLICACIÓN. ......................................................................................... - 4 -](#_bookmark2)

[PRIMERA PARTE: INTERFACES .................................................................................................................**- 4 -**](#_bookmark3)

[*SEGUNDA PARTE: USO DE EXCEPCIONES EN JAVA* .......................................................................................**- 6 -**](#_bookmark4)

# [DESARROLLO DE LA PRÁCTICA .............................................................................................. - 8 -](#_bookmark5)

[**MATERIAL A ENTREGAR Y PLAZOS ........................................................................................ - 8 -**](#_bookmark6)

# [CRITERIOS DE EVALUACIÓN .................................................................................................. - 8 -](#_bookmark7)

**Objetivos y alcance de la práctica**

Los objetivos que se pretenden cubrir con el desarrollo de esta práctica son los siguientes:

1. Trabajar el concepto de interface estudiado en teoría.
2. Familiarizarse con el uso de excepciones.

La duración de esta práctica es de 2 semanas.

**Prerrequisitos.**

Es imprescindible para la realización adecuada de esta práctica:

* Haber asistido a las clases de teoría y realizado todas las actividades de aprendizaje propuestas correspondientes a los apartados de interfaces y excepciones.
* Haber realizado las prácticas 1 a 5 propuestas en el laboratorio de la asignatura.

**Descripción de la aplicación.**

El objetivo de la práctica es generar efectos nuevos que puedan ser aplicados sobre una imagen y que permitan realizar cambios en su color, asignando a cada pixel de la imagen un nuevo valor basándose en el color original del mismo pixel. De entre la multitud de posibles efectos, se codificarán dos en esta práctica: trasformar una imagen a escala de grises y modificar el brillo de la imagen.

Además, en la programación de las nuevas clases se utilizará el mecanismo de excepciones para el control de los errores.

**Primera parte: interfaces**

Después de estudiar el problema, y debido a la cantidad de efectos - y clases - que puede ser necesario programar, se ha decidido utilizar una interfaz para simplificar la solución. Esta interfaz permitirá separar la trasformación realizada sobre el color de un solo píxel del efecto que se aplicará posteriormente sobre todos los píxeles de una imagen completa. Así, se deberá programar la siguiente interface:

public interface **TrasformarColor**

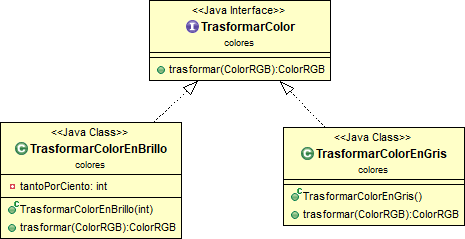
Esta interfaz permite la trasformación de un color RGB en otro color RGB. Su método trasformar genera un nuevo color RGB basándose únicamente en las componentes RGB del color original.

***Method Summary***

|  |  |  |
| --- | --- | --- |
| **Type** | **Method** | **Description** |
| ColorRGB **trasformar**(ColorRGB colorOriginal) Genera un nuevo color basado en las componentes RGB del  color original. | | |

De esta interfaz se realizarán dos implementaciones en las clases TrasformarColorEnGris y

TrasformarColorEnBrillo, tal como se muestra en el siguiente diagrama UML:



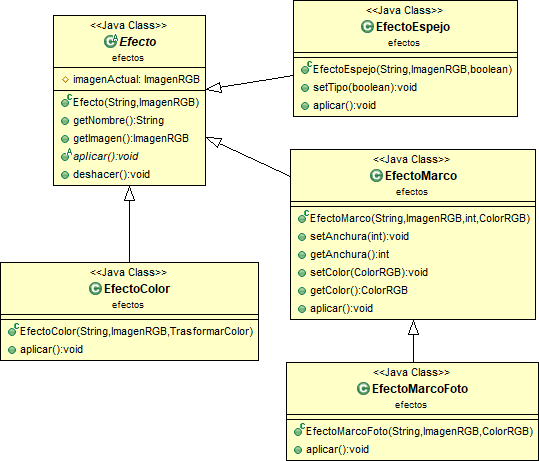
**Ilustración 1. Diagrama UML del paquete colores**

Los elementos del diagrama anterior se añadirán en un nuevo paquete llamado colores. Cada una de estas clases tendrá su propia codificación del método trasformar:

* La trasformación a grises se realiza de forma simple sumando las componentes RGB y dividiendo por 3: (R+G+B)/3. El valor resultante se aplica a todas las componentes del nuevo color.
* El cambio de brillo se realizará sumando a cada componente RGB un tanto por ciento de su valor (el mismo tanto por ciento para todas las componentes), que puede ser positivo (aumentar brillo) o negativo (disminuir brillo).

Para aplicar la transformación de color sobre todos los píxeles de una imagen se añadirá un nuevo efecto, llamado

EfectoColor, al paquete efectos de la práctica anterior, tal como se muestra en el siguiente diagrama UML:



**Ilustración 2 Diagrama UML del paquete efectos**

La principal característica de este nuevo efecto es que **recibe** en su constructor, como parámetro, **un objeto** (TransformarColorEnBrillo o TransformarColorEnGris) con el método de transformación de color a aplicar a cada píxel de la imagen, utilizando como tipo la interfaz TrasformarColor. Su método *aplicar* se encargará de realizar la trasformación indicada con todos los píxeles de la imagen.

*Segunda parte: Uso de excepciones en Java*

En esta parte se procederá a generar y tratar excepciones en Java. Se generará una excepción cuando se detecte alguna anomalía en determinados argumentos de algunos métodos, o cuando el resultado de un método sea incorrecto. Tenga en cuenta, en lo sucesivo, que **un método** elige terminar con “return” (si ha realizado su función conforme a su especificación y, en su caso, devuelve el valor oportuno) o con “throw” de una excepción (si se ha producido alguna situación excepcional que no le ha permitido realizar su función u obtener el resultado). Pero nunca debe terminar con “return” cuando se ha producido un error que no le ha permitido realizar adecuadamente su función.

En esta actividad se deberá modificar la aplicación para que:

* Las clases que se indiquen modifiquen sus constructores y métodos para que avisen a quien los invoca (sólo avisar: nunca mostrar por su cuenta mensajes en salida estándar, etc.) de situaciones excepcionales por medio de excepciones. **Todas las excepciones que se lancen deben transportar un mensaje** (un objeto String) que detalle la naturaleza del error.
* El programa principal capture y trate las excepciones anteriores. Este tratamiento puede ser tan sencillo como:
  1. Imprimir en la salida estándar un mensaje de error constituido por una parte propia (qué estaba intentando hacer el programa) y el mensaje con el detalle del error que transporta la excepción.
  2. Terminar la ejecución.

El alumno debe modificar el código desarrollado en los paquetes efectos y colores para usar dos clases de excepciones:

1. Se deberá lanzar la excepción IllegalArgumentException en los constructores y métodos de todas las clases cuando los parámetros de entrada que no sean referencias a objetos tengan valores no válidos, concretamente:
   1. En el constructor de la clase TrasformarColorEnBrillo cuando el parámetro correspondiente al tanto por ciento no esté comprendido entre -100 y 100,
   2. En el constructor de la clase EfectoMarco y en su método setAnchura cuándo el parámetro ancho sea menor que 0, o mayor que la mitad del menor valor entre el ancho y el alto de la imagen.
2. Definir una excepción de usuario de clase EfectoException en el paquete efectos. Esta excepción se lanzará en el método deshacer de la clase Efecto cuando se llame a este método y no se haya aplicado previamente el efecto alguna vez desde la creación del efecto, o desde la última llamada a deshacer; O sea, cuando no se puede deshacer nada.

Nota: Para controlar si algún efecto ha sido aplicado se hace necesario incluir un nuevo atributo en la clase Efecto que registre este hecho. Puede codificar el método aplicar de la clase Efecto si le simplifica la programación (con lo que no será ya abstracto) o declarar el nuevo atributo como *protected* para poder realizar un acceso directo al mismo desde las subclases.

En el programa principal debe completar el código de la plantilla proporcionada para ejecutar las operaciones que se le indican, entre ellas se encuentra un cambio a escala de grises y varios aumentos de brillo, realizados con la clase EfectoColor. Además, en la segunda parte, se capturarán las posibles excepciones EfectoException e IllegalArgumentException, dentro del método *main*, utilizando dos bloques catch separados: en ambos casos se mostrará el mensaje descriptivo del error que incluye la excepción y el programa terminará en ese punto. En el método operarEfecto no se capturará ninguna excepción.

**IMPORTANTE:** para aprovechar sus sesiones de trabajo es imprescindible que realice, previamente a cualquier codificación, la lectura detallada del enunciado y del *javadoc* de la práctica.

**Desarrollo de la práctica**

Se recomienda seguir la siguiente secuencia de pasos:

1. Copiar el proyecto eclipse ***practica5*** y pegarlo renombrándolo a ***practica6.***
2. Codificar la interfaz TransformarColor y las dos clases que la implementan.
3. Codificar la subclase EfectoColor, que debe heredar de Efecto.
4. Cumplimentar la plantilla de la clase principal *P6Aplicacion* para comprobar el correcto funcionamiento de la estructura implementada en la primera parte de la práctica.
5. Codificar la nueva excepción EfectoExcepcion y añadir la gestión de excepciones en las clases correspondientes de los paquetes efectos y colores.
6. Modificar la clase *P6Aplicacion* añadiendo la gestión de excepciones.

**Material a entregar y plazos**

El alumno deberá subir a Moodle (en la tarea de entrega habilitada a tal efecto) todos los archivos fuente de las actividades de la práctica. Para ello creará un fichero comprimido (de tipo zip) con los archivos fuente (.java) estructurados en paquetes. Dichos archivos deberán tener los nombres especificados en el enunciado.

El nombre del archivo ZIP debe tener el siguiente formato:

*grupolaboratorio\_P6\_apellidos\_nombre.zip*

Por ejemplo, un alumno que se llamara Rafa Nadal y que estuviera en el grupo X03X04 subiría a la plataforma Moodle el fichero *X03X04\_P6\_Nadal\_Rafa.zip.*

El plazo límite de entrega en Moodle de los archivos fuente correspondientes con las actividades de la práctica será el **10 de mayo de 2021 a las 23:59**.

**Criterios de evaluación**

Esta práctica es de **entrega obligatoria**. Los Requisitos de Entrega son los siguientes:

* Se ha subido a Moodle en el plazo indicado.
* Se entregan archivos fuente desarrollados de todas las clases solicitadas.
* Los archivos fuente pueden ser objeto de compilación y de chequeo respecto a que el comportamiento de la aplicación sea compatible con el solicitado en el enunciado.

El incumplimiento de cualquiera de estos requisitos tendrá su reflejo en la nota final, de acuerdo a lo indicado en la Guía de Aprendizaje de la asignatura.

La evaluación de los conocimientos y habilidades obtenidas con esta práctica se realizará en el segundo examen parcial.