Práctica 1 - Procesado de imágenes

El objetivo de esta práctica es aplicar los conocimientos adquiridos sobre las clases `ProcessBuilder`, `Process`, y `Streams` y sus derivados, escribiendo un programa capaz de convertir a escala de grises un conjunto de imágenes mediante programación concurrente.

Imagen original

Imagen en escala de grises

escala de grises

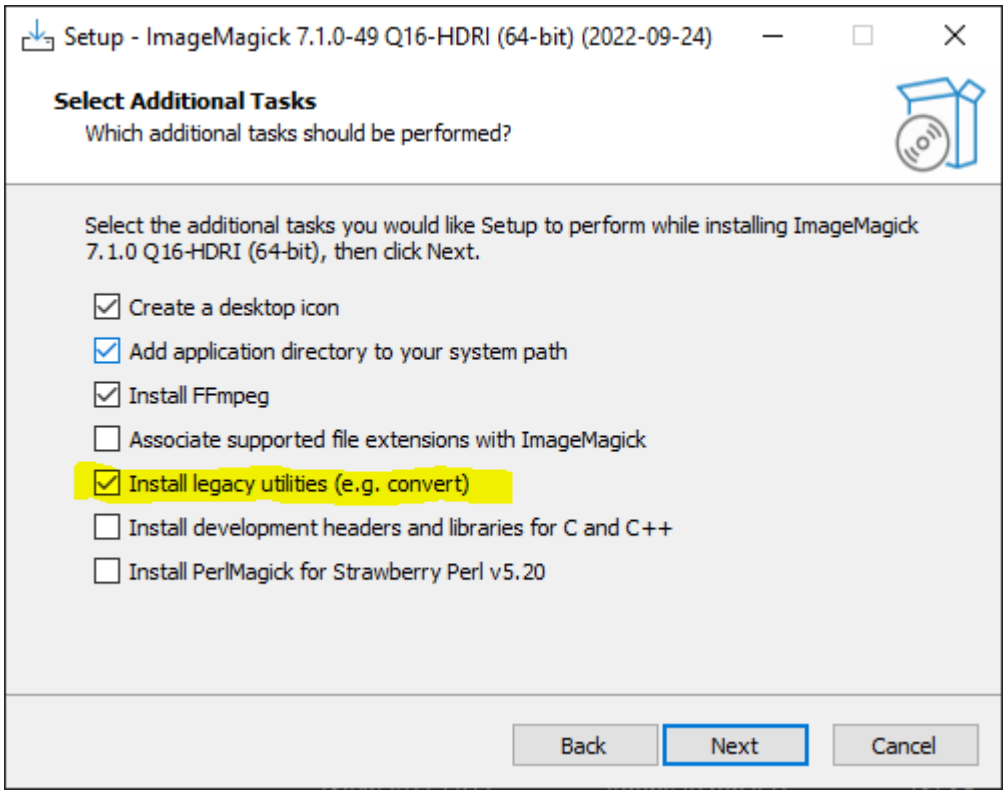
En la práctica deberás entregar un proyecto de IntelliJ con dos módulos, uno padre y otro hijo.

Módulo hijo

El módulo hijo implementará la funcionalidad de convertir a escala de grises con la ayuda de la biblioteca ImageMagick. La implementación recibirá la ruta a la imagen origen a través de los argumentos del método `main`, creará una ruta para la imagen destino siguiendo el formato nombre-fuente.gray.jpg. De esta forma, el fichero `101-367x267.jpg` se convertiría en `101-367x267.gray.jpg`). Seguidamente realizará la conversión a escala de grises utilizando el código de ejemplo proporcionado y devolverá a través de la salida estándar la ruta al fichero generado.

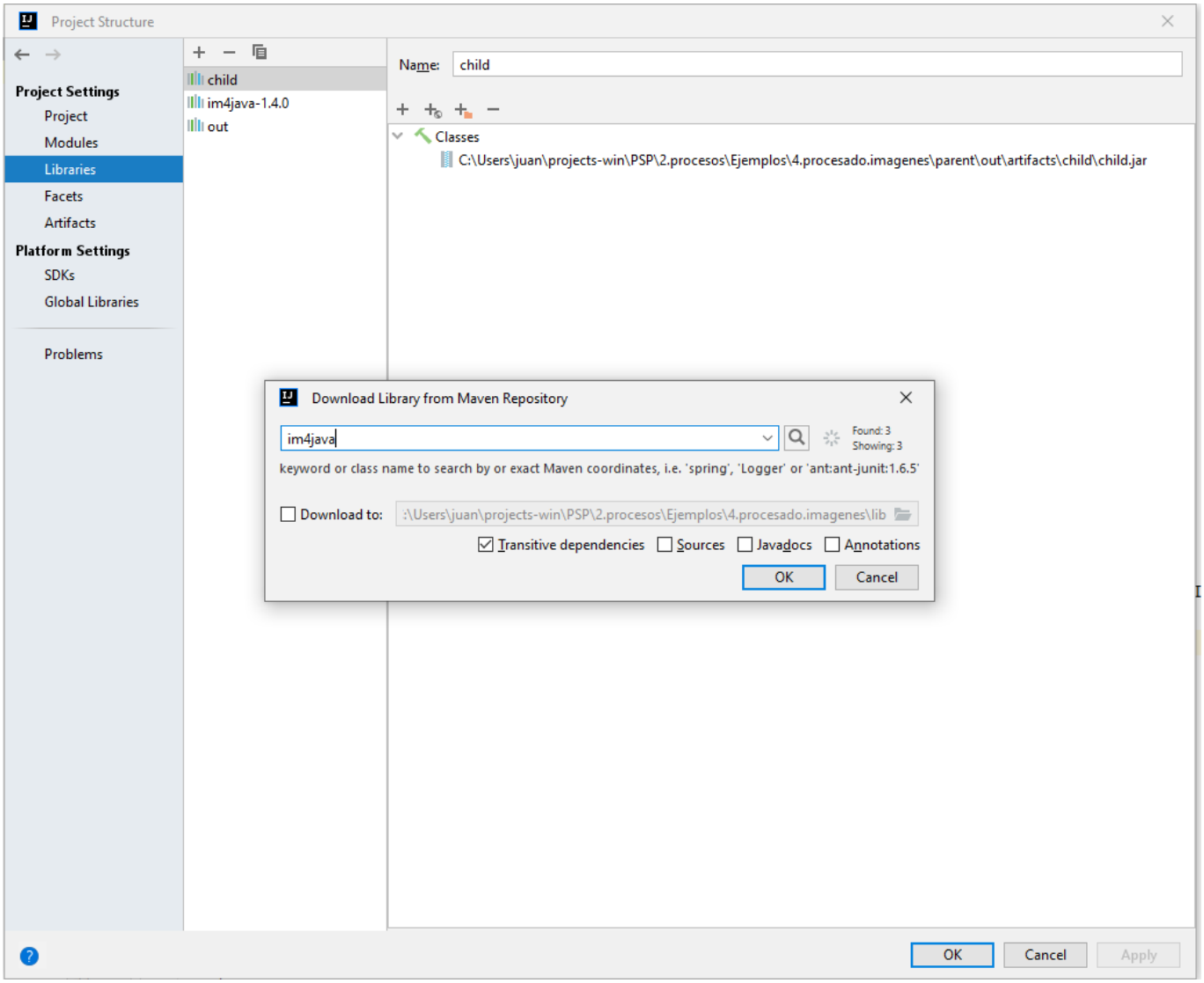
Instalación de ImageMagick

1. Descarga la versión ImageMagick para Windows desde su página oficial o desde este enlace
2. Instala la aplicación prestando especial atención a marcar la opción de instalar las `legacy utilities` tal y como se muestra en la figura:



Importa la biblioteca im4java

En el diálogo de `estructura de proyecto` selecciona el módulo hijo y añade una dependencia con la biblioteca im4java descargada desde Maven, como se muestra en la figura



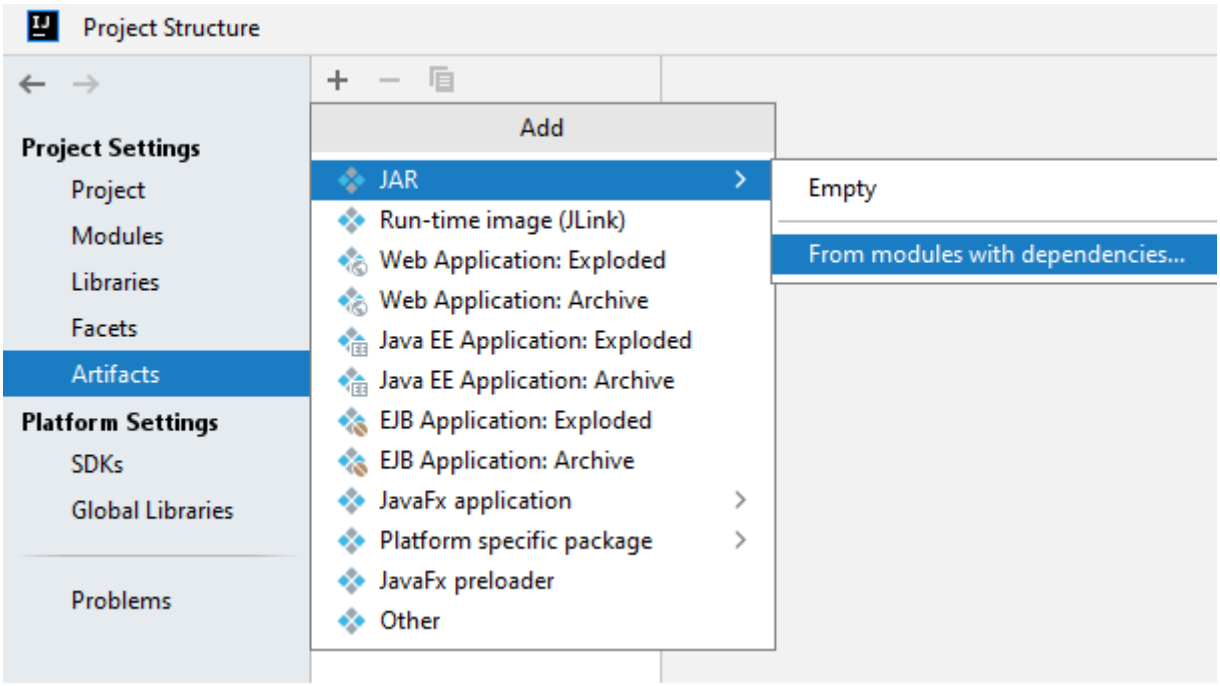
Utiliza la biblioteca im4java

Para convertir una imagen a escala de grises puedes utilizar el siguiente fragmento de código

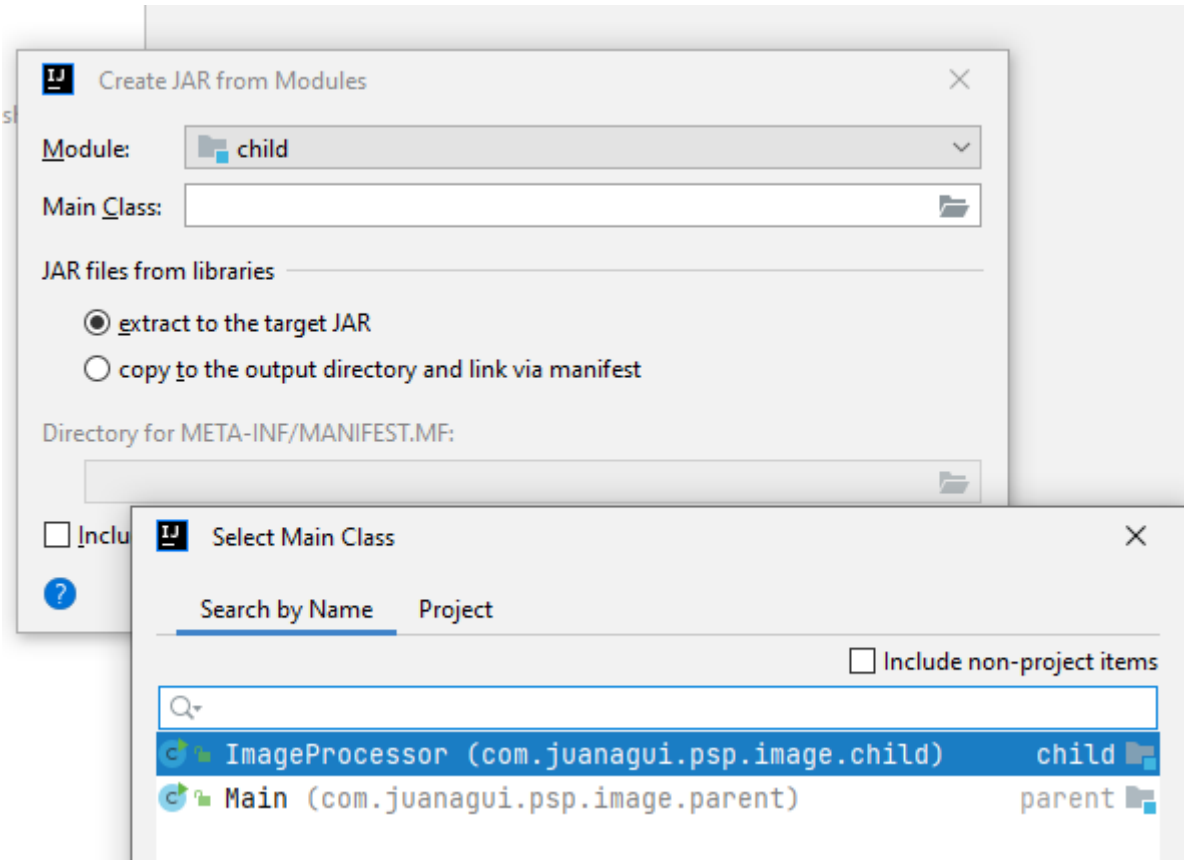
```
// Define la ruta a donde se encuentran la instalación de ImageMagick.
// El fichero convert.exe debe estar en este directorio
public static final String IMAGE_MAGICK = "C:\\Program Files\\ImageMagick-7.1.0-
Q16-HDRI";

...

// Con este código puedes convertir la imagen que se encuentre en path a una
imagen en escala de grises que se almacena en dstImageFilePath
ProcessStarter.setGlobalSearchPath(IMAGE_MAGICK);
ConvertCmd cmd = new ConvertCmd();
IMOperation op = new IMOperation();
op.addImage(path.toString());
op.colorspace("Gray");
op.addImage(dstImageFilePath);
cmd.run(op);
```

Generar un JAR

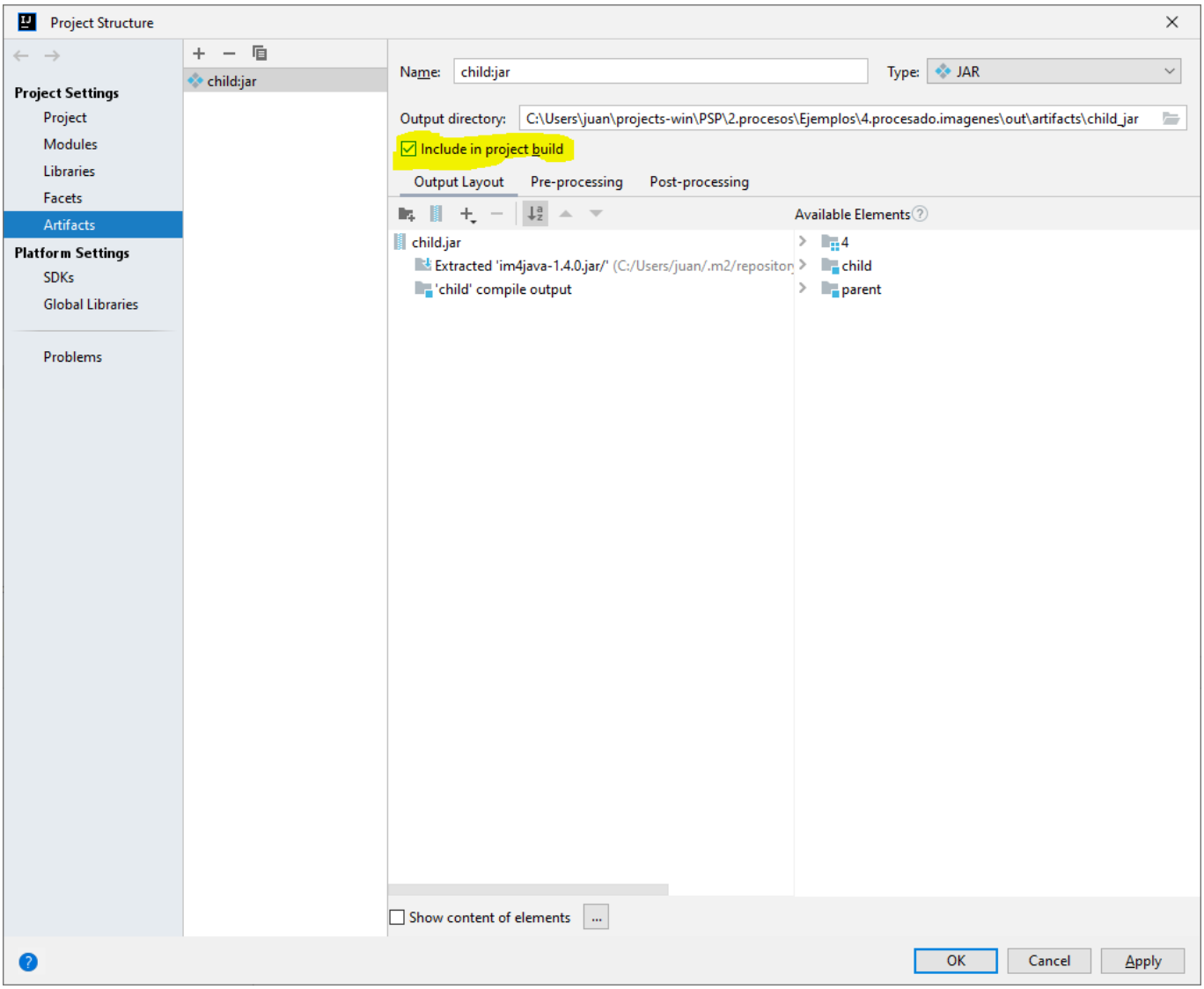
Para generar un JAR a partir de un módulo, vamos al diálogo `estructura de proyecto` y en artefactos, añadimos un JAR a partir de un módulo con dependencias, como se observa en la figura:



Seguidamente seleccionamos el módulo y la clase que contiene el `main`



El resultado final, en el que marcamos la casilla para que se genere el JAR al construir el proyecto, se observa en la siguiente figura:



Módulo padre

El módulo padre utilizará la clase `Files.newDirectoryStream` para leer los ficheros contenidos dentro del directorio `lorem` en el que se encuentran las imágenes de ejemplo obtenidas del sitio web Lorem Picsum

Para cada uno de los ficheros creará y lanzará un proceso hijo con la siguiente línea de comandos:

```
java -jar ruta-absoluta-al-fichero.jar
```

> ⚠ **asegúrate de que los procesos de ejecutan concurrentemente**. Bloquear el proceso padre hasta que el hijo termine de procesar una imagen es un grave error de concepto

Una vez hayas lanzado todos los procesos, asegúrate de que **esperas** a que terminen y escribe por la salida estándar el nombre del fichero convertido que cada proceso devuelve por su **salida**. Para ello deberás mantener una lista de los procesos que has creado.

Mejoras

Lanzar un proceso por cada fichero que queramos convertir es ineficiente y potencialmente peligroso para la estabilidad del sistema si en el directorio se encontrasen cientos de imágenes.

Una alternativa mejor diseñada, utilizaría una serie de procesos hijos creados de antemano, que esperarían en un bucle infinito a recibir la ruta al fichero a convertir a través de la entrada estándar. En este caso el proceso padre tiene que encargarse de ir enviando los ficheros a los procesos en bloques, en lugar de crear un proceso por cada fichero.

Si te sientes aventurero, crea una versión de la práctica con el diseño ineficiente que funcione y entrégala. Después añade a la entrega una versión con este diseño más eficiente.