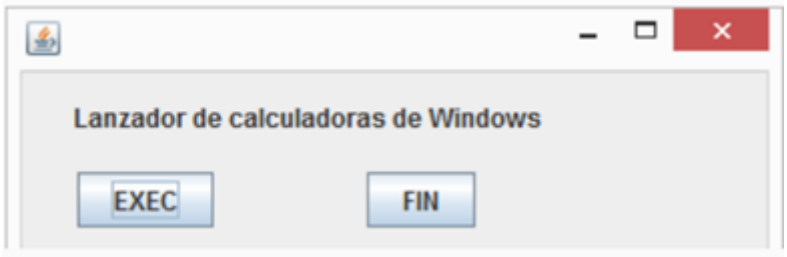
EJERCICIO 1: Programa que lanza varios procesos

Aplicación Java que lanza varias calculadoras a la vez, lo que implica crear tantos procesos de la calculadora como queramos usar.

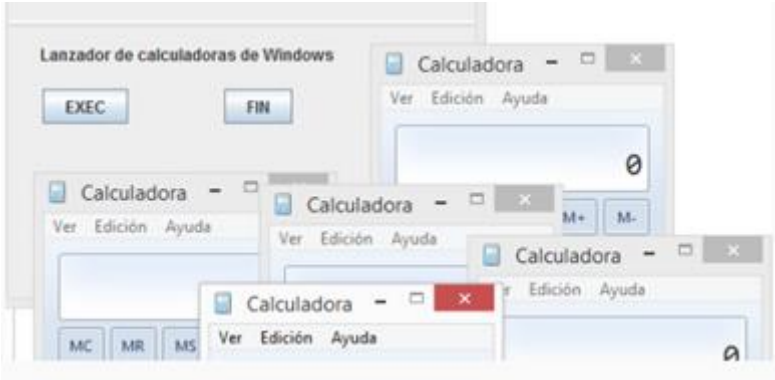
Pasos:

1. Crea un proyecto NetBeans que ejecuta la calculadora lanzando un proceso que ejecuta el comando Calc. Genera la aplicación “calculadora.jar”.
2. Crea un proyecto Netbeans "lanzador" que permita ejecutar varios procesos de "calculadora.jar". Para ello, nuestra aplicación "lanzador" tendrá estas características:

- Será una ventana con dos botones:
  - EXEC: que lanzará tres procesos de “calculadora.jar” cada vez que se pulse.
  - FIN: finalizará la aplicación



3. Genera la aplicación "lanzador.jar" y prueba su ejecución desde la línea de comandos. Haz una captura de pantalla de la ejecución del comando.



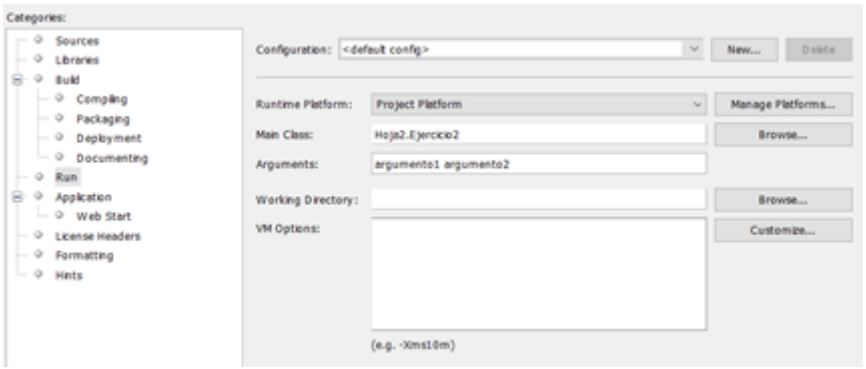
EJERCICIO 2: El comando ping desde Java

Realiza un programa Java que ejecute en Windows el comando de MS-DOS “CMD /C PING *ip*” que nos dice si un recurso de la red está accesible o no. El programa tendrá que hacer ping a un rango de IPs consecutivas.

Tendrás que hacer los siguientes pasos:

1. El programa *MiPing* recibirá tres argumentos: la primera ***ip*** a hacer ping del rango, el ***número de Ips*** del rango y un nombre de ***fichero*** donde guardar la salida de los pings.

- Para pasar argumentos a un programa desde NetBeans deberás ir al menú: Run -> Set Project Configuration ->Customize…
- En el apartado run, en arguments deberás pasar los argumentos separados por un espacio.



Muy importante ejecutar el proyecto pinchando en el botón de Play verde de NetBeans para que coja los argumentos.

2. Deberás lanzar tantos comandos pings como IPs haya en el rango. Por ejemplo, si la ip comienza por la 142.250.201.67 y hay que hacer cinco pings, habrá que lanzar un proceso nuevo para los pings:

ping 142.250.201.67

ping 142.250.201.68

ping 142.250.201.69

ping 142.250.201.70

ping 142.250.201.71

3. Por cada proceso lanzado en el programa principal (main), se leerá lo que devuelve el proceso línea a línea (**readLine()** de la clase BufferedReader) para imprimirlo por pantalla y en un fichero de texto.

- Para obtener un flujo de entrada conectado con la salida del proceso hijo, deberás utilizar el método **getInputStream()** de la clase Process.

4. El proceso principal (main) deberá esperar deberá esperar a que termine la ejecución de cada proceso hijo y obtener el valor de finalización del proceso y mostrarlo por pantalla.

- Para que el proceso principal (main) espere a los procesos hijos tendrás que utilizar el método **waitFor()** de la clase Process.