PSP - U2 Proyectos

Descargar estos apuntes

Índice

1. Clase psp.proyectos.U2P1_LanzadorProgramas

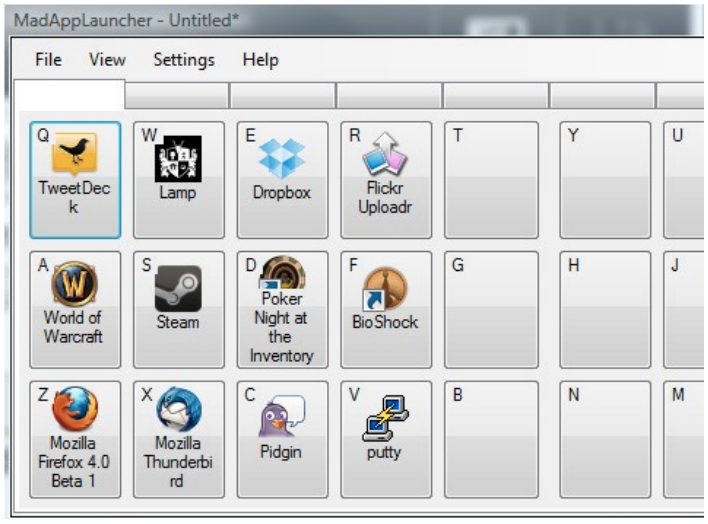
Realiza un programa en Java que, usando la librería gráfica **Swing**, nos muestre un panel de control para ejecutar distintas aplicaciones.

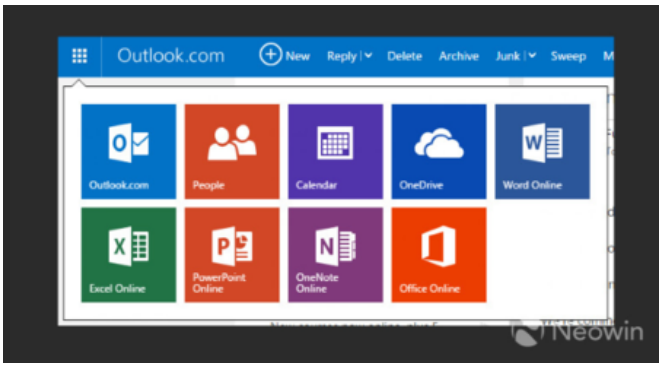
Videotutorial creación proyecto Swing en Netbeans

Guía de programación de proyecto Swing en Netbeans(1)(2)
(1) Es una guía muy completa y breve. Sólo debéis tener en cuenta que el primer paso que hace realmente debería estar entre los pasos 4 y 5, tal y como se explica en el videotutorial.
(2) Es muy interesante el proceso que explica antes de la Figura 8 para cambiar el nombre de los campos sin que nos de problemas.

Puedes crear todas las clases auxiliares que consideres necesarias. Todas ellas deben estar dentro del paquete psp.proyectos.

Este lanzador tendrá un aspecto similar a la pantalla de un móvil.





No es necesario que nuestro Lanzador incorpore menús y el diseño os lo dejo a vuestra elección, aunque no será un apartado calificable en la actividad, por lo tanto, no dediquéis mucho tiempo a esto.

El tamaño de la ventana será fijo. Así que el JFrame que usemos no será “Resizable”. Cada aplicación estará representada por un JButton que contendrá una imagen de la aplicación y, opcionalmente un texto (Ver imágenes de ejemplo).

Podemos lanzar tanto aplicaciones gráficas como comandos del sistema. Para mostrar la posible salida que generen las aplicaciones que creemos nuestro lanzador tendrá un JTextArea en la parte inferior, al estilo de la consola que aparece en Netbeans, donde mostraremos la salida y los errores que produzcan las aplicaciones.

Debéis incluir al menos tres aplicaciones gráficas y un botón para ejecutar comandos de consola.

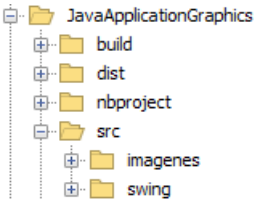
Para indicar el comando de consola que queremos ejecutar, incluiremos en el diseño un JTextField donde introduciremos el comando, sus argumentos y sus parámetros. La salida del comando se visualizará en el JTextArea de la parte inferior.

> ⚠ **Control de procesos activos**

> Para evitar que se lance más de una instancia de cada aplicación, cada vez que se pulse un botón, este se deshabilitará.
>
> Para volver a habilitar todos los botones deshabilitados, añadiremos otro botón más, “Actualizar”, que comprobará si el proceso lanzado por cada uno de los botones sigue activo (seguirá dejando el botón deshabilitado) o si por el contrario el proceso ha terminado (volverá a habilitar el botón).
>
> Este control se debe hacer de forma dinámica. No se valorará el uso de variables por botón, ya que en cualquier momento se pueden añadir nuevas aplicaciones al lanzador y esto no debe afectar al código ya escrito.

Vamos a dar la opción de que la salida, en vez de mostrarse en el JTextArea, se guarde en un archivo. Para indicar esta posibilidad vamos a añadir un `JCheckBox` a nuestra aplicación de forma que si no está marcado la salida se muestra en la ventana y si está marcado, la salida (estándar y de error) se irá guardando en el archivo `salida.txt`, de forma incremental, en el directorio raíz del proyecto.

La forma de añadir imágenes en nuestra aplicación Swing es incluyéndolas en una carpeta dentro del directorio src



y después las añadimos a los controles y a la ventana usando el siguiente código. Las propiedades que controlan el posicionamiento dentro del botón de la imagen y el texto son:

- horizontalAlignment/verticalAlignment: Alineación horizontal y vertical del icono y el texto dentro del botón
- horizontalTextPosition/verticalTextPosition: Posición horizontal/vertical del texto en relación con la imagen

Para añadir el icono de la aplicación (al JFrame)

```
URL url1 = MainFrame.class.getResource("/imagenes/play.png");
ImageIcon image1 = new ImageIcon(url1);
this.setIconImage(image1.getImage());
```

Para añadir el icono a un botón

```
URL url2 = MainFrame.class.getResource("/imagenes/open.png");
ImageIcon image2 = new ImageIcon(url2);
jButton1.setIcon(image2);
//Versión con redimensionado de imagen, por ejemplo a 32x32
//jButton1.setIcon(new ImageIcon(image2.getImage().getScaledInstance(32, 32, Image.SCALE_DEFAULT)));
```

> **MainFrame** debe sustituirse por el nombre de nuestra clase que hereda de JFrame.

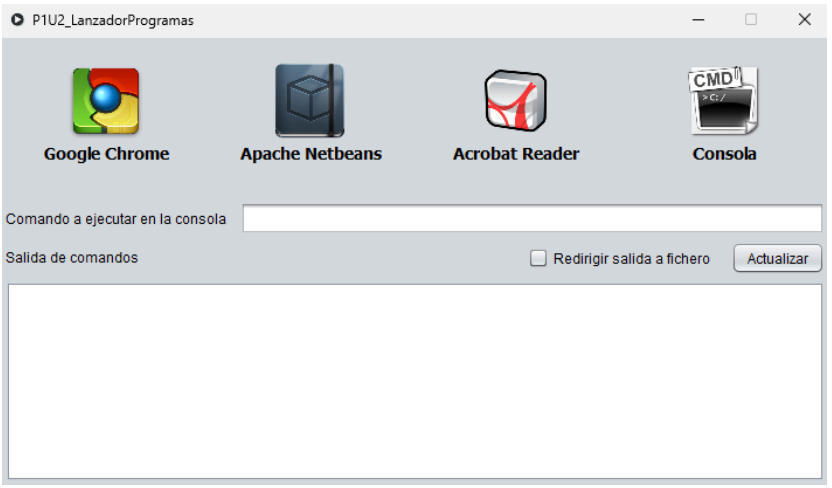
Todo el código de carga de imágenes debe hacerse en el constructor de nuestro JFrame, justo después de haber inicializado los componentes, es decir, podemos llamar a un método creado por nosotros justo después de la llamada a initComponents().

```
public class MainFrame extends javax.swing.JFrame
public MainFrame() {
    initComponents();
    addComponentImages(); // Método que añade las imágenes a los componentes
}
```

Control de eventos

¿Podríamos hacer un waitFor() en el evento de pulsación del botón? Explica qué ocurriría y por qué.
¿Cómo podrías solucionarlo?

Un ejemplo de como podría ser el aspecto de nuestro lanzador es el siguiente:



2. Clase psp.proyectos.U2P2_ApiRestTester

Realiza un programa en Java que, usando la librería gráfica **Swing**, nos muestre un entorno gráfico para poder ejecutar el comando `curl` con todas las opciones que hay en el Anexo II del tema 2.

En la lista de APIs para probar puedes incluir las que se ven en los apuntes o cualquier otra de las que se proponen en este listado de API REST “diferentes”. Si quieres usar la API de traducción de Azure, escríbeme un correo para pedirme la secret key de acceso. Debes incluir un mínimo de 2 API REST en el desplegable para probar.

En la documentación de las API podrás encontrar las rutas que puedes usar para tus pruebas. Las rutas al ser dinámicas, se escribirán de forma manual.

Tienes que seleccionar el método de envío (GET / POST) que se utilizará en tu petición

En los TextArea de Header y Data se añadirá una línea por cada encabezado o pareja clave-valor de datos que se quiera enviar en la solicitud.

Si el envío es información en formato JSON, habrá que indicarlo marcando el checkbox, de forma que todo el contenido de TextArea Data se enviará como información JSON. Además, al marcar el checkbox debes generar un evento que añada la cabecera de content-type JSON al TextArea Header.

Cuando se pulse el botón “Enviar”, se recogerá toda la información de la pantalla y se preparará el comando `curl` a ejecutar. El comando completo se mostrará en el TextField y se ejecutará.

La salida (estándar y error) que produzca el comando curl se recogerá y se mostrará en el TextArea de la parte inferior.

> Cuando entregues la actividad, debes indicarme qué rutas puedo probar para el API que hayas seleccionado.