

Actividad 8

1. A partir del ejemplo visto en clase con el ping 8.8.8.8, probarlo y hacer:

- Probarlo en windows donde el padre espere a que el hijo muera (4 ping)
- Probarlo en linux, donde el padre muestre 10 ping del hijo y luego lo mate)
- ¿Qué sucede si en la ejecución del hijo, el padre no lo mata?

Probaremos el programa en linux.

Creamos la clase *Tarea8*

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Tarea8 {
    public static void main(String[] args) {
        Runtime runtime = Runtime.getRuntime();
        Process process = null;
```

```

try {
    process = runtime.exec("ping " + args[0]);
    BufferedReader in = new BufferedReader(new
    for (int i = 0; i < 10; i++) {
        System.out.println("Saludo desde PSP 2
    }
} catch (IOException e) {
    System.out.println("No pudimos correr el p
    System.exit(-1);
}
if (process != null) {
    process.destroy();
    System.out.println("Me he cargado el ping.
}
try {
    System.out.println("Ahora esperaré a que a
    process.waitFor();
    System.out.println("Ya no existe mi proces
} catch (InterruptedException e) {
    System.out.println("No pudimos esperar por
    System.exit(-1);
}
System.out.println("Estado de término: " + pro
System.exit(0);

}

}

```

Uso

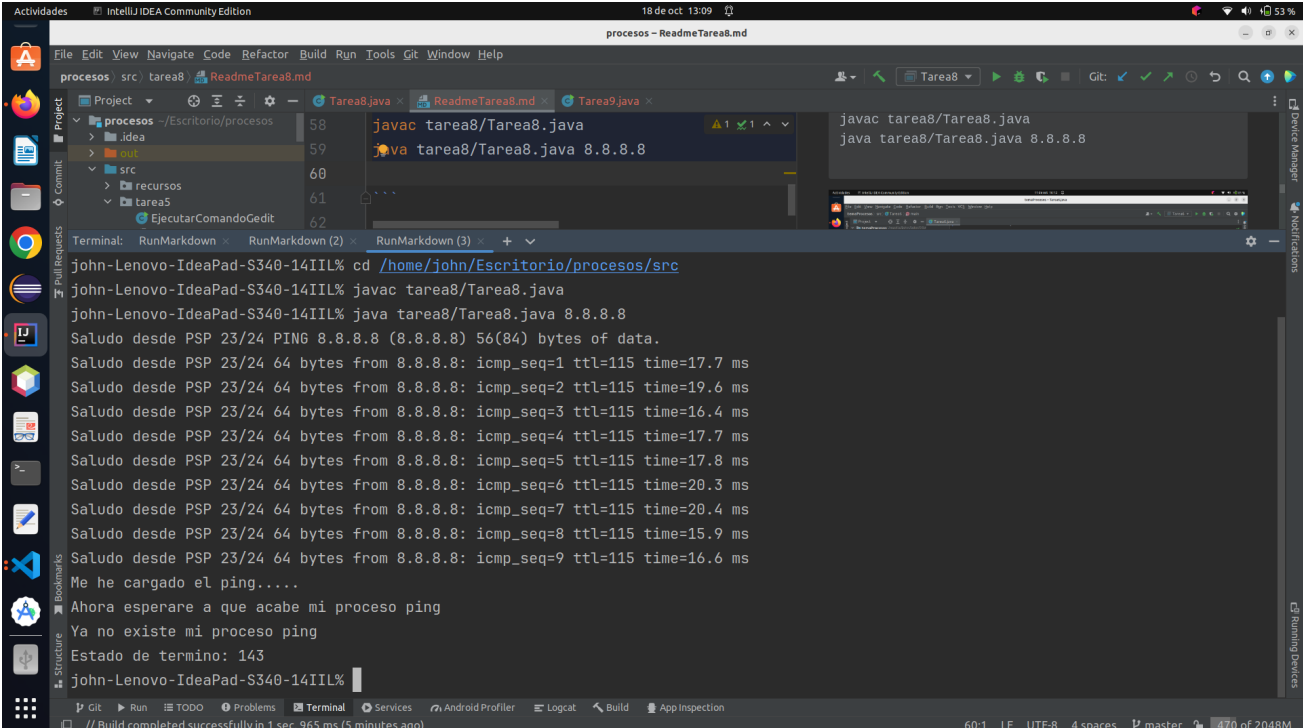
El programa se ejecuta desde la línea de comandos y requiere un argumento que debe ser la dirección IP o el nombre de host al que se le realizará el ping.

Ejecutamos nuestro programa: ▶ Tarea8

Ejecutamos nuestro programa desde la consola con los siguientes comandos en mi caso son estos porque mi proyecto se encuentra en el escritorio y mi programa se encuentra en el package tarea8, le pasemos un argumento en este caso:

8.8.8.8

```
» cd /home/john/Escritorio/procesos/src
javac tarea8/Tarea8.java
java tarea8/Tarea8.java 8.8.8.8
```



```
IntelliJ IDEA Community Edition
procesos - ReadmeTarea8.md

Tarea8.java x ReadmeTarea8.md x Tarea9.java x
javac tarea8/Tarea8.java
java tarea8/Tarea8.java 8.8.8.8

Terminal: RunMarkdown x RunMarkdown (2) x RunMarkdown (3) x +
john-Lenovo-IdeaPad-S340-14IIL% cd /home/john/Escritorio/procesos/src
john-Lenovo-IdeaPad-S340-14IIL% javac tarea8/Tarea8.java
john-Lenovo-IdeaPad-S340-14IIL% java tarea8/Tarea8.java 8.8.8.8
Saludo desde PSP 23/24 PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=17.7 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=19.6 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=16.4 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=4 ttl=115 time=17.7 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=5 ttl=115 time=17.8 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=6 ttl=115 time=20.3 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=7 ttl=115 time=20.4 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=8 ttl=115 time=15.9 ms
Saludo desde PSP 23/24 64 bytes from 8.8.8.8: icmp_seq=9 ttl=115 time=16.6 ms
Me he cargado el ping....
Ahora esperare a que acabe mi proceso ping
Ya no existe mi proceso ping
Estado de termino: 143
john-Lenovo-IdeaPad-S340-14IIL%

// Build completed successfully in 1 sec, 965 ms (5 minutes ago)
60:1 LF UTF-8 4 spaces master 470 of 2048M
```

Este programa ejecuta el comando "ping" en un sistema operativo, en este caso linux, tomando como argumento la dirección IP o el nombre de host proporcionado en la línea de comandos.

Descripción

El programa utiliza la clase `Runtime` para interactuar con el sistema operativo y ejecutar el comando "ping" en una nueva instancia de proceso. Luego, captura y muestra la salida del comando "ping" en la consola. Después de 10 líneas de salida del "ping", se destruye el proceso "ping".

Detalles del Programa

1. Se crea una instancia de la clase `Runtime` y un objeto `Process` para gestionar el proceso de ping.
2. El programa intenta ejecutar el comando "ping" con la dirección IP o nombre de hosts proporcionados en la línea de comandos. En caso de error, se muestra un mensaje y se sale del programa.
3. Se crea un objeto `BufferedReader` para leer la salida del proceso "ping".
4. El programa entra en un bucle que recorre 10 líneas de la salida del "ping". En cada iteración del bucle, muestra la línea de salida en la consola con un mensaje de saludo.

5. Después de recopilar 10 líneas de salida del "ping", el proceso "ping" se destruye.
6. El programa espera a que el proceso "ping" finalice.
7. Una vez que el proceso ha finalizado, se muestra el estado de terminación del proceso.
8. El programa sale con un código de salida apropiado.

Observaciones

- El programa utiliza el comando "ping" para realizar un ping a la dirección IP o nombre de hosts proporcionados, y muestra las respuestas en la consola.
- Si el proceso "ping" no puede iniciarse, se mostrará un mensaje de error y el programa se cerrará con un código de error.
- Después de capturar 10 líneas de salida del "ping", el proceso "ping" se destruirá para finalizarlo.
- Finalmente, el programa espera a que el proceso "ping" finalice y muestra el estado de terminación del proceso.

¿Qué sucede si en la ejecución del hijo, el padre no lo mata?

Si el padre no mata al proceso hijo y lo deja en ejecución, el programa se quedará en un estado de espera indefinido, ya que la llamada `process.waitFor()` bloqueará la ejecución del padre hasta que el proceso hijo termine.

En otras palabras, si el proceso hijo no se cierra o finaliza por sí solo, el programa padre se mantendrá en espera y no continuará ejecutando ninguna operación adicional. En este escenario, el programa quedará "atascado" indefinidamente hasta que el proceso hijo se cierre de alguna manera, ya sea por la finalización natural de la operación de ping o por intervención manual del usuario para finalizar el proceso.

Por lo tanto, es importante asegurarse de que el proceso hijo pueda finalizar correctamente o, en caso de ser necesario, implementar una lógica adicional para finalizar el proceso hijo desde el padre en función de ciertas condiciones o límites de tiempo, para evitar que el programa quede bloqueado en un estado de espera indefinido.

Dejo el enlace al repositorio

<https://github.com/johnlopez0505/procesos.git>