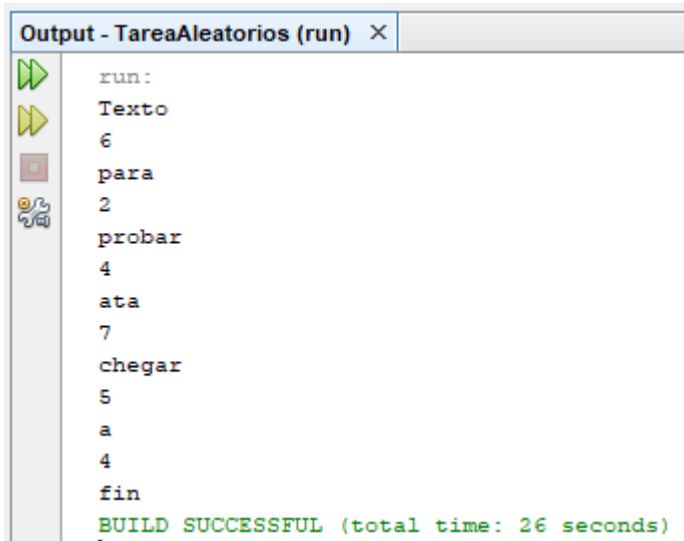


## Ejercicio 1: programa Aleatorios

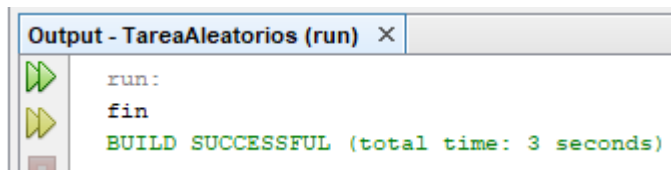
Este ejercicio consiste en un programa, **Aleatorios**, que espera un número por teclado y, como respuesta, devuelve un número aleatorio. Dicho número estará generado por un programa al que hemos denominado **AleatoriosHijo**.

El resultado esperado es:



```
run:
Texto
6
para
2
probar
4
ata
7
chegar
5
a
4
fin
BUILD SUCCESSFUL (total time: 26 seconds)
```

Tal y como figura en la captura del enunciado, al escribir **fin**, el proceso finaliza sin recibir respuesta:



```
run:
fin
BUILD SUCCESSFUL (total time: 3 seconds)
```

Como vemos en esta imagen, en la que únicamente se escribe la palabra de finalización, el programa finaliza correctamente.

Dicho comportamiento lo hemos comentado en el código:

```
// Bucle en el que leemos desde teclado hasta que coincida con la palabra de finalización
while (!texto.equals(STOP)) {
    writer.write(texto); //Enviamos el texto
    writer.newLine(); // Enviamos una nueva línea (para que sepa que puede devolver el número)
    writer.flush(); // Vaciamos el buffer

    System.out.println(reader.readLine()); // Leemos la "respuesta" del hijo
    texto = scan.nextLine(); // Esperamos por una nueva escritura en teclado
}

//Finalizamos el proceso hijo
p.destroy();

// Cerramos los dos buffers
writer.close();
reader.close();
```

Como se puede observar, la comprobación de si se ha introducido la palabra de finalización se realiza antes de comunicarnos con el proceso hijo.

En la parte del hijo, cada vez que recibimos texto, generamos un número aleatorio

```
// Bucle infinito en el que se genera un número aleatorio cada vez que se recibe una nueva línea.
do {
    br.readLine(); // Leemos lo recibido. No lo almacenamos ya que no es necesario.
    System.out.println(r.nextInt(11)); // Generamos un número aleatorio
} while (true);
```

Este proceso se repite infinitamente hasta que el proceso es finalizado. De ahí el `p.destroy()` del proceso padre.