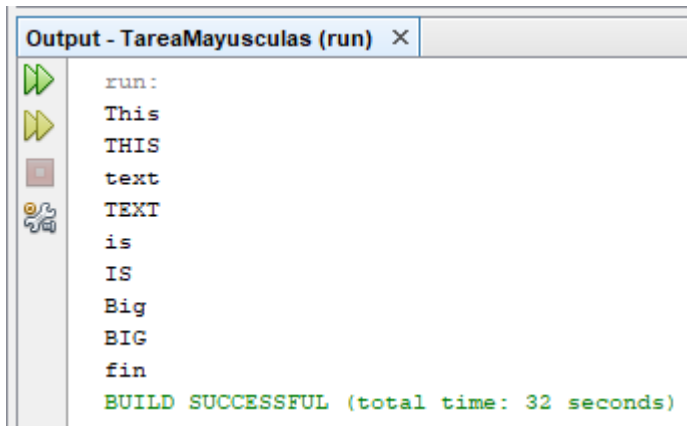


Ejercicio 2: programa Mayusculas

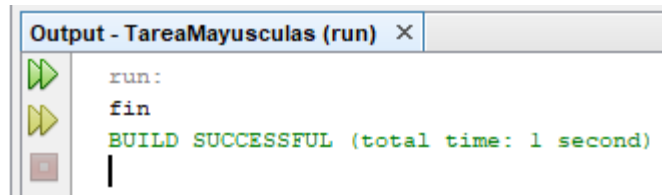
Este ejercicio consiste en un programa, **Mayusculas**, que espera un texto por teclado y, como respuesta, devuelve el mismo texto en mayúsculas. Dicho texto estará generado por un programa al que hemos denominado **MayusculasHijo**.

El resultado esperado es:



```
Output - TareaMayusculas (run) X
run:
This
THIS
text
TEXT
is
IS
Big
BIG
fin
BUILD SUCCESSFUL (total time: 32 seconds)
```

Hemos incorporado la palabra **fin** como palabra clave para finalizar el proceso.



```
Output - TareaMayusculas (run) X
run:
fin
BUILD SUCCESSFUL (total time: 1 second)
```

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 has
while (!texto.equals(STOP)) {
    writer.write(texto); //Enviamos el text
    writer.newLine(); // Enviamos una nueva
    writer.flush(); // Vaciamos el buffer

    System.out.println(reader.readLine());
    texto = scan.nextLine(); // Esperamos p
}
```

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, lo devolvemos en mayúsculas.

```
// Bucle infinito en el que se genera un número
do {
    texto = br.readLine(); // Leemos lo recibido
    System.out.println(texto.toUpperCase()); //
} while (true);
```

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

Como observación, los programas padre `Mayusculas` y `Aleatorios` se podrían adaptar/unificar en uno mismo. Por ejemplo, se le podría dar la opción al usuario de que eligiese si quiere un número aleatorio o el texto en mayúsculas. Por nuestra parte, únicamente tendríamos que cambiar la variable asignada al directorio de ejecución del proceso hijo así como el nombre del proceso `.java`.