**1)**  Crear un método **obtenerNombre**() en **ProcesosUtils** que se encargue de devolver un String con vuestro nombre. Mejor si ponéis vuestro nombre como una variable estática constante (como RUTA\_CLASS por ejemplo) y desde ese método la referenciáis.

Crear un método **obtenerHoraActual**() en ProcesosUtils con este codigo:

/\*\*\* Obtiene la hora actual de la zona horaria de Madrid (Europa)

\* @return

\*/

public static String obtenerHoraActual() {

       LocalTime horaActual = LocalDateTime.now(ZoneId.of("Europe/Madrid")).toLocalTime();

       return horaActual.toString();

}

**2)** Realizar una clase **Ejercicio1Multiple** (en otro paquete procesos.ejercicio1Multiple) y otra clase **HolaMundoMultiple**. Esta última que reciba un parámetro en los argumentos, que imprimirá por pantalla junto con vuestro nombre y la hora actual (ver método de utilidad entregado por el profesor).

El mensaje será algo como "**Mi amo/a es Diego. He recibido el número 6, y son las 12:10:30.811713500.** ". Utilizad String.format para representarlo.

El padre pedirá al usuario un número de entrada (“totalProcesos”) (usad la clase Scanner para solicitar un valor al usuario). Se creará un bucle de 1 a N para crear un subproceso que ejecute una instancia de HolaMundoMultiple, enviándole como parámetro el número actual del bucle (“cont”), para que cada subproceso muestre el número correspondiente recibido por argumentos. El padre establecerá que todos los procesos escriban su salida en el mismo fichero determinado, y lo mismo para la salida de error, pero en otro fichero distinto.

Ejecútalo y mira el fichero de salida:

**¿Coincide el orden de las invocaciones por número con la hora de ejecución?**

**2.1)** Ampliar el ejercicio anterior, estableciendo un "número prohibido". Si la clase **HolaMundoMultiple** recibe un número que sea par, elevará una excepción con un mensaje determinado ("**Proceso dice: ¡el número Y es un número prohibido!**"). Comprobar que se imprime ese mensaje en el fichero de error.

**2.2)**Ampliar el ejercicio anterior, estableciendo que se haga un waitFor() de cada proceso hijo.

**¿Coincide el orden de las invocaciones por número con la hora de ejecución?**

**2.3)**Que el subproceso duerma un número de segundos igual al número que se le está pasando por parámetros.

**2.4)**Eliminar el waitFor que pusimos en el ejercicio 2.2, y que el padre especifique que se coja la salida estándar de los procesos para imprimir su salida por pantalla.

**¿Qué pasa con el orden de ejecución de los procesos?**

**2.5)**Coged lo mismo del ejercicio anterior (2.4) y en vez de llamar al método imprimirSalidaProcesoHijo, después de construir el objeto ProcessBuilder, invocad su método **.redirectOutput(Redirect.INHERIT)** y **.redirectError(Redirect.INHERIT)**

**¿Qué pasa con el orden?**

**2.6) Coged lo mismo del ejercicio anterior pero en vez de las llamadas .redirectOutput(Redirect.INHERIT) y .redirectError(Redirect.INHERIT), invocad al método .inheritIO() de vuestro objeto ProcessBuilder. ¿Se comporta igual la salida que en el 2.5?**

**3)** Ahora vamos a rizar el rizo (cuidao). Lo que queremos es que el **Ejercicio1Multiple**, que hasta ahora recibía el input del número de procesos a ejecutar desde teclado (entrada estándar) a mano, ahora se ejecute como un subproceso y reciba el nº de procesos desde la entrada estándar que será alimentada por otro proceso padre. Haremos una clase llamada **Ejercicio1MultipleInvocador**, que haga básicamente lo que venimos haciendo hasta ahora:

·         Declare la ruta del “out” del proyecto donde están los .class

·         Cree el ProcessBuilder con la invocación a la clase. En este caso queremos invocar a la clase Ejercicio1Multiple

·         Redirigiremos la salida hacia el padre, con el método que vimos en el 2.5

Y ahora haremos algo nuevo: crearemos un canuto desde el padre para que podamos volcar un número hacia el hijo. Hacer este código en un método estático privado de la clase, que reciba un parámetro de tipo Process. Por ejemplo la firma de ese método puede ser **private** **static** **void** mandarDatosAHijoWriter(Process hijo). La lógica que hará es:

1.       Del proceso hijo, obtenemos su outputStream, y creamos un nuevo objeto OutputStreamWriter con él.

2.       De esa variable de tipo OutputStreamWriter, invocamos el método write para escribir un número constante como String (por ej: “20”) dentro de ese stream de datos que viajará al hijo.

3.       Decimos al writer que haga “flush”

Ejecutamos y … magic!

**4)**Ahora el nivel DIOS: vamos a coger lo anterior, y en vez de pasarle el número a fuego, vamos a pedir al usuario ese número, y se lo vamos a pasar por el stream al hijo, utilizando el mismo método que antes