```
EIF400-II-2025
Paradigmas de Programación.
Escuela de Informática
UNA
```

SPEC de Expresso: Un minilenguaje muy concentrado

Anexo I Alcances esperados Sprint inicial

1. Un programa en Java (no `.bat`, no `.sh` ni similares) que permite desde un `cmd` o `terminal` teclear

   a. `expressor transpile --out output HelloWorld.Expresso`: Procesa adecuadamente las opciones (usando alguna herramienta como la sugerida en el SPEC) para obtener `transpile` `--out` `output` y `HelloWorld.Expresso` u otra opción. Dado que en este caso es un `transpile`:

      i. Lee de disco el archivo `HelloWorld.expresso` que no está vacío (su contenido no es relevante). Mensaje de error sino existe ó está vacío.

      ii. Lee de disco de una carpeta `resources/template` (relativo a raíz del proyecto) un archivo `HelloWorld.java` que pueda compilarse y correrse (eso lo hacen a mano. Que tenga una lógica mínima, por ejemplo saludar 10 veces en consola).

      iii. Salva textualmente `HelloWorld.java` en la carpeta `output` ese archivo. Si esa carpeta no existiera la crea.

      iv. Puede recibir otra opción `--verbose` con lo cual indica qué pasos está haciendo. Ejemplo: leyendo, transpilando

      - Si `--out nombre_de_carpeta` es omitido es lo mismo `--out .` (es decir, salva en la misma carpeta donde se ejecuta el comando expressor

   b. `expressor build --out output HelloWorld.expresso` En este caso análogo que en a., pero en vez `expressor` compila dejando el `.class` en `output`.

   c. `expressor run --out output HelloWorld.Expresso` En este caso análogo que antes en b. pero además ejecuta el `.class` generado.

2. Un proyecto bien organizado que se pueda compilar desde un `cmd` o `terminal` sin tener que usar un IDE (puede ser un proyecto `maven` or `gradle` o simplemente un `.bat` (`.sh`) si no conocen esas herramientas). El proyecto cumple con los estándares de github. El `README` describe el proyecto, indica como hacer el build y ejecutar y cualquier indicación pertinente al profesor/usuario. Recuerde todo fuente (incluyendo el `README`) debe tener un comentario que identifica proyecto, al curso, autores, código de grupo de trabajo. Incluye referencias usadas. Considere además que se puede requerir recompilar durante la revisión.

3. Estar en capacidad cada estudiante de defender el proyecto.

4. Subir al drive asignado al grupo (le llegará invitación al coordinador a más tardar 26/08), quien es el/la responsable de subir el proyecto, así:

   a. Es un zip (o equivalente estándar). Nombre exacto `EIF400-II-2025_Expresso_Initial_GG-HH_NNNN` donde `GG-HH` es el código de grupo asignado (por ejemplo `GG=01` , `HH=1pm` es el grupo `01-1pm`) y `NNNN` es el nombre y apellido del coordinador.

   b. El contenido del zip es una única carpeta `expresso` y ahí adentro el proyecto.

   c. Fecha y Hora máximo (puntual al minuto): `31/08 12md` . Luego cuenta como no entregado, no se revisa. Pueden subir antes pero solo se puede subir una vez. Lo contrario anula el proyecto.

5. Evaluación la indicada en el SPEC. El día `Lunes 1/9` en la hora de cada grupo se hace una demo usando el entregable subido al drive. No se puede usar otro. El profesor publicará el fin de semana una `Guía de Revisión` que cada grupo llena e imprime y presenta el `1/9` en los primeros 5 minutos de cada clase para poner el profesor las observaciones y la nota.

6. El coordinador mantiene la privacidad del drive asignado. Si le asignara derechos a terceros sin autorización del profesor, se anula el proyecto por copia/plagio.

7. El profesor recompilará cada proyecto en Windows desde un cmd (tomar en cuenta eso). Pueden asumir que tengo JDK requerido.

8. Cualquier incumplimiento mínimo en tiempo y forma anula el proyecto. Cualquier dificultad ajena al profesor que impida o haga muy extenso el tiempo para revisar el proyecto durante la demo o en la revisión offline anula el proyecto.