Explicación

Usamos un estilo de programación basado en *flujos* para procesar los ficheros y convertirlos al resultado final. Además, en aquellos casos donde no precisamos de **herencia**, usamos `record` para reducir líneas de código innecesarias.

Nota: Debido al uso de patrones en expresiones `switch`, es posible que el programa requiera ser ejecutado con el flag `--enable-preview`.

Entrada

Cuando el usuario inicia el programa, se le pedirá que seleccione el directorio donde se encuentran los archivos. Estos deben estar nombrados como `Fichero Participantes.txt` y `Fichero Etapa X.txt`. Si el usuario cancela la entrada, el programa terminará.

Posteriormente se le pide la etapa a generar. El programa generará tanto la etapa indicada como las anteriores.

Procesamiento

Participantes

Para leer los participantes, leemos las líneas y se le aplica el método `Participants::parse`. Este método se encarga de convertir la línea proporcionada en una estructura que después podremos manipular con mayor facilidad. Por último, acumulamos los participantes a un `Map` para poder acceder a los participantes según su código de forma directa y no tener que iterar una lista.

Etapa

Como cada archivo contiene una etapa, aplicamos el método `Stage::parse` a la lista de líneas obtenidas de leer el archivo. Procesamos la cabecera de la etapa con `StageHeader::parse` y, a continuación, agrupamos el resto de líneas de forma que cada grupo contenga la cabecera de la carrera seguida con los tiempos de cada participante.

Del primer grupo obtenemos los tiempos de salida de cada participante. Al resto le aplicamos `Race::parse`.

Carreras

`Race` es una clase abstracta que instanciaremos en `Climb` (puerto), `BonusSprint` (meta volante) o `LastRace` (meta).

El método `Race::parse` se encarga de generar la cabecera correcta junto al top y la lista de tiempos. Usamos la expresión `switch` para instanciar la subclase apropiada y devolverla en un único `return`.

Rankings

`Rankings` contiene ambas clasificaciones de puntos (de puerto y meta volante).

`Rankings::generateNext` se encarga de generar la clasificación siguiente en base a la clasificación y carrera anteriores.

Proceso Principal

El proceso principal se encuentra en `App::main`. Aquí, además de procesar los datos como se ha indicado anteriormente, acumulamos los tiempos de la etapa actual con la anterior, y generamos el archivo. Tras generarlo, eliminamos los participantes que han abandonado y guardamos los participantes que continuan corriendo junto la clasificación actual.

Pruebas

Se han escrito pruebas unitarias para los métodos `_::parse` y para `Time`. El contenido del archivo generado se ha revisado de forma manual, junto al funcionamiento de la entrada de datos por parte del usuario (directorio y etapa).