Programación Orientada a Objetos

Práctica 3 - El proyecto *marketplace*. Google Test.

Google Test

- Una de las principales estrategias de desarrollo es el desarrollo guiado por tests (*test driven development* – TDD).
- Se escriben primero las pruebas que *verifican* si nuestros requisitos se están cumpliendo, y después el código que las satisface.
  - Si el código pasa la prueba, cumple con nuestros requisitos.
- Con frecuencia pruebas a nivel de *unidad* (*Unit testing*).
- Uno de los frameworks más utilizados para realizar tests unitarios es el que proporciona Google.

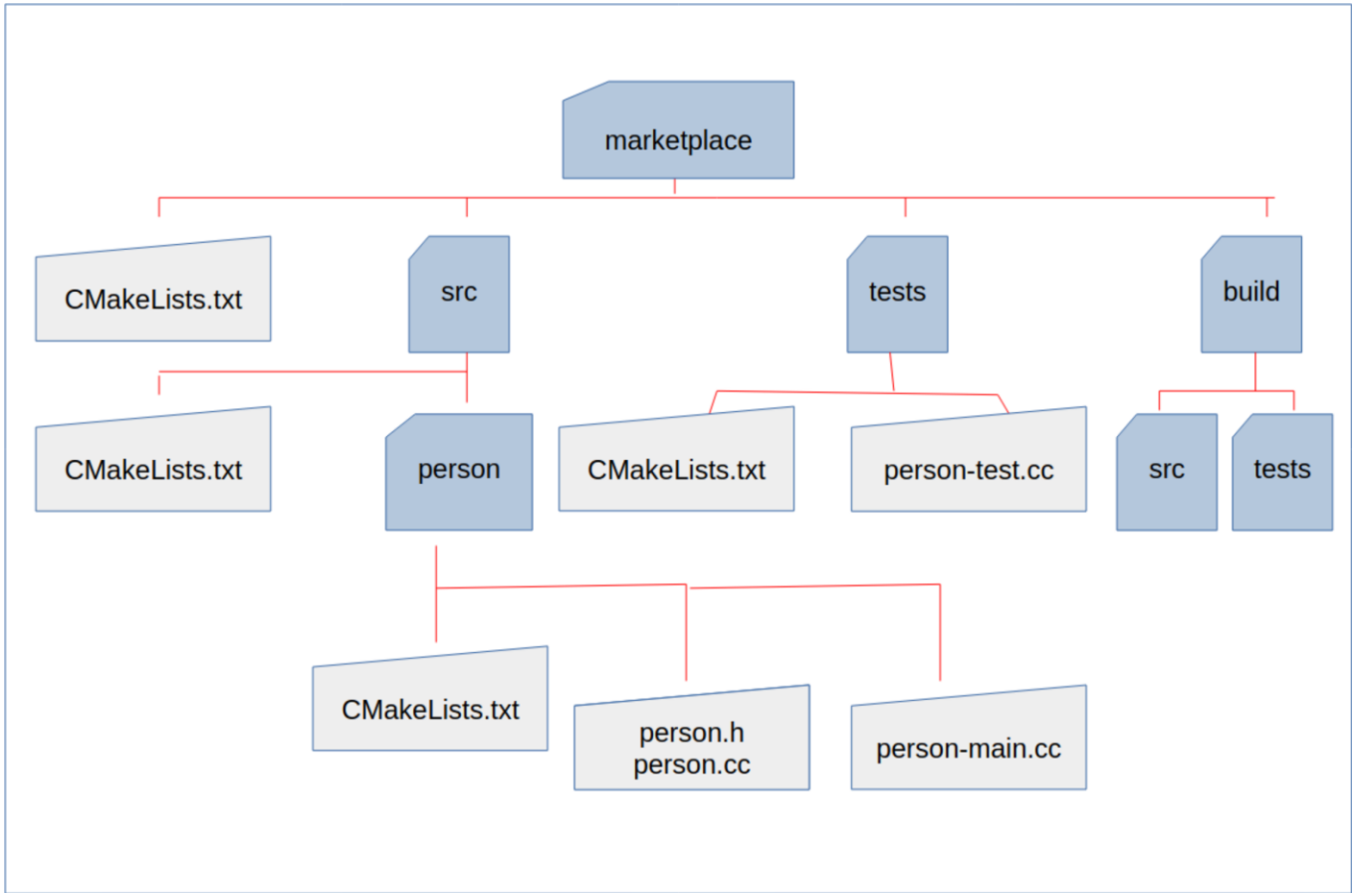
Google Test

- Para definir tests usaremos las siguientes instrucciones de CMake:
  - enable_testing(): Habilita el uso de tests.
  - add_test(<nombre> <comando/target_ejecutable>): Añade un nuevo test unitario llamado nombre, que ejecuta el programa indicado por el segundo parámetro.
- Por su parte, para hacer uso de los test de Google:
  - include(FetchContent): Permite descargar contenido de internet.
  - FetchContent_Declare(
    googletest
    GIT_REPOSITORY https://github.com/google/googletest.git
    GIT_TAG v1.14.0
    ): Declara las dependencias de Google Test.
  - FetchContent_MakeAvailable(googletest): "Puebla" las dependencias a partir de su declaración.

El proyecto *marketplace*

- Continuamos a partir del código de la Práctica 2, pero esta vez tendremos 3 directorios:
  - marketplace/build: Directorio de destino para los ejecutables.
    - También los test compilados.
  - marketplace/src: Directorio con el código fuente.
  - marketplace/tests: Directorio que contendrá el código de test.
- El código de test necesario para esta práctica está disponible en Moodle (person-test.cc).
  - Deberéis ubicarlo correctamente en la estructura de ficheros.

El proyecto marketplace



La clase Person

- Con el objetivo de probar el funcionamiento de los tests, se ampliará la funcionalidad de la clase Person.
  - Nuevos campos: id_, town_, province_, country_, entry_year_
    - Se ampliará el constructor, con valores por defecto y una verificación en el campo entry_year.
  - Setters y getters para todos los campos.
  - SetAge(), SetEntryYear(): Devolverán *true* si se permite el cambio, *false* si el cambio no tiene sentido (no se modificará entonces).
- GetDataStr() deberá seguir un formato predefinido.

Pasando los test

- Para pasar los test de forma manual, podéis ejecutarlos desde el directorio build/tests.
  - Tras compilar!

- El comando ctest lanzará todos los test configurados.
  - Si hemos habilitado los tests con enable_testing() y añadido los test usando add_test()!

- Prueba las dos opciones y analiza su salida.

Ejercicio

- Modificar el proyecto para incluir los test disponibles en Moodle.
  - Añadir los subdirectorios y CMakeLists pertinentes.
- Modificar la clase Person para incluir los nuevos campos y métodos.
- Compilar y ejecutar los tests; observar la salida.
- Limpiar y optimizar el código; mejorar la presentación, autodocumentación, etc.
  - Modificar el código (no los tests) para corregir cualquier error y pasar los tests.
- Estudiar todo el material adicional disponible en Moodle.