Diagrama de clases

a) Para la implementación de la práctica 4 se ha elaborado el diseño de clases mostrado a continuación. Este parte de las interfaces dadas para generar el resto de clases.

Las clases `SiLengthMetricSystem`, `SiTimeMetricSystem` e `ImperialLengthMetricSystem` heredan de *`MetricSystem`* (que implementa la interfaz `IMetricSystem`), y utilizan el patrón de diseño Singleton para tener instancias únicas de cada sistema métrico. Para esto es necesario que sus constructores sean privados. Las unidades de estos sistemas son instanciadas como variables de clase que no pueden ser modificadas para evitar posibles duplicados.

En cuanto a las excepciones, se ha optado por hacer que `UnknownUnitException` herede de `QuantityException`, de forma que se simplifique su gestión.

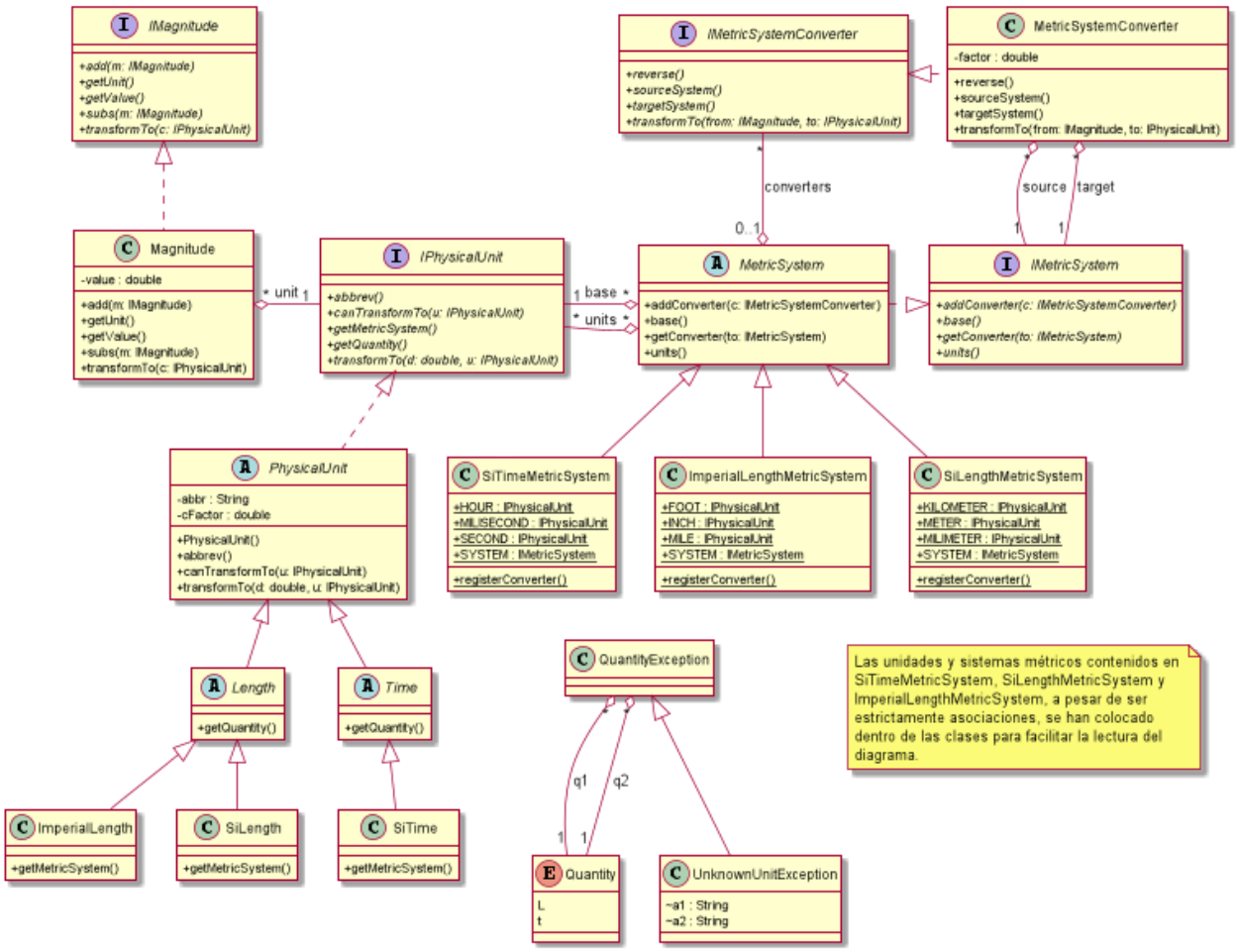

Figura 1: Diagrama de clases de la práctica 4

b) El diseño de las clases permite que este se pueda extender en un futuro sin modificar la estructura actual.

Si se quisiera ampliar un sistema métrico con nuevas unidades, tan solo habría que instanciarlas como variables estáticas de la misma forma que las demás.

Para añadir nuevos tipos de cantidad a la biblioteca hay que añadirlos a la enumeración y crear una clase abstracta que herede de *`PhysicalUnit`* y tenga un funcionamiento análogo al de

Length y *Time* (podría ser *Mass* en este caso). También habría que añadir una clase heredera de esta última por cada sistema métrico de masa que se quisiera añadir para poder instanciar las unidades posteriormente; en este caso, `SiMass`. Finalmente, han de implementarse los sistemas mérticos de masa asociados a las unidades de masa concretas creadas, esto es, clases que heredan de `MetricSystem`; en este caso, `SiMassMetricSystem`, a la que habría que añadir unidades constantes de la clase `SiMass` (`KILOGRAM`, `GRAM`, `MILLIGRAM`...).

c) Un posible punto débil de esta implementación es que, al usarse el patrón Singleton en los sistemas métricos y heredar estos de *MetricSystem*, no es posible poner como privados los constructores de las unidades físicas, ya que no es posible la herencia múltiple en Java (por ejemplo, puede haber dos objetos `METER` con los mismos atributos). Sin embargo, al utilizarse las unidades como constantes de clase, no habrá lugar a confusiones al referirse a ellas ni pueden ser alteradas posteriormente, además de garantizarse la unicidad de unidades en un sistema métrico (siguiendo el ejemplo, `SiLengthMetricSystem.METER` es único).