package com.twitter.product\_mixer.core.quality\_factor

/\*\*

\* [[QualityFactor]] is an abstract number that enables a feedback loop to control operation costs and ultimately

\* maintain the operation success rate. Abstractly, if operations/calls are too expensive (such as high

\* latencies), the quality factor should go down, which helps future calls to ease their demand/load (such as

\* reducing request width); if ops/calls are fast, the quality factor should go up, so we can incur more load.

\*

\* @note to avoid overhead the underlying state may sometimes not be synchronized.

\* If a part of an application is unhealthy, it will likely be unhealthy for all threads,

\* it will eventually result in a close-enough quality factor value for all thread's view of the state.

\*

\* In extremely low volume scenarios such as manual testing in a development environment,

\* it's possible that different threads will have vastly different views of the underling state,

\* but in practice, in production systems, they will be close-enough.

\*/

trait QualityFactor[Input] { self =>

/\*\* get the current [[QualityFactor]]'s value \*/

def currentValue: Double

def config: QualityFactorConfig

/\*\* update of the current `factor` value \*/

def update(input: Input): Unit

/\*\* a [[QualityFactorObserver]] for this [[QualityFactor]] \*/

def buildObserver(): QualityFactorObserver

override def toString: String = {

self.getClass.getSimpleName.stripSuffix("$")

}

}