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

/\*\*

\* Provides a way to apply inclusive min/max bounds to a given value.

\*/

case class Bounds[T](minInclusive: T, maxInclusive: T)(implicit ordering: Ordering[T]) {

def apply(value: T): T = ordering.min(maxInclusive, ordering.max(minInclusive, value))

def isWithin(value: T): Boolean =

ordering.gteq(value, minInclusive) && ordering.lteq(value, maxInclusive)

def throwIfOutOfBounds(value: T, messagePrefix: String): Unit =

require(isWithin(value), s"$messagePrefix: value must be within $toString")

override def toString: String = s"[$minInclusive, $maxInclusive]"

}

object BoundsWithDefault {

def apply[T](

minInclusive: T,

maxInclusive: T,

default: T

)(

implicit ordering: Ordering[T]

): BoundsWithDefault[T] = BoundsWithDefault(Bounds(minInclusive, maxInclusive), default)

}

case class BoundsWithDefault[T](bounds: Bounds[T], default: T)(implicit ordering: Ordering[T]) {

bounds.throwIfOutOfBounds(default, "default")

def apply(valueOpt: Option[T]): T = valueOpt.map(bounds.apply).getOrElse(default)

}