package com.twitter.product\_mixer.component\_library.selector.sorter

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

/\*\*

\* Makes a [[Sorter]] to run for the given input based on the

\* [[PipelineQuery]], the `remainingCandidates`, and the `result`.

\*

\* @note this should be used to choose between different [[Sorter]]s,

\* if you want to conditionally sort wrap your [[Sorter]] with

\* [[com.twitter.product\_mixer.component\_library.selector.SelectConditionally]] instead.

\*/

trait SorterProvider {

/\*\* Makes a [[Sorter]] for the given inputs \*/

def sorter(

query: PipelineQuery,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): Sorter

}

/\*\*

\* Sorts the candidates

\*

\* All [[Sorter]]s also implement [[SorterProvider]] to provide themselves for convenience.

\*/

trait Sorter { self: SorterProvider =>

/\*\* Sorts the `candidates` \*/

def sort[Candidate <: CandidateWithDetails](candidates: Seq[Candidate]): Seq[Candidate]

/\*\* Any [[Sorter]] can be used in place of a [[SorterProvider]] to provide itself \*/

override final def sorter(

query: PipelineQuery,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): Sorter = self

}