package com.twitter.follow\_recommendations.common.rankers.weighted\_candidate\_source\_ranker

import com.twitter.follow\_recommendations.common.utils.RandomUtil

import scala.util.Random

sealed trait CandidateShuffler[T] {

def shuffle(seed: Option[Long])(input: Seq[T]): Seq[T]

}

class NoShuffle[T]() extends CandidateShuffler[T] {

def shuffle(seed: Option[Long])(input: Seq[T]): Seq[T] = input

}

class RandomShuffler[T]() extends CandidateShuffler[T] {

def shuffle(seed: Option[Long])(input: Seq[T]): Seq[T] = {

seed.map(new Random(\_)).getOrElse(Random).shuffle(input)

}

}

trait RankWeightedRandomShuffler[T] extends CandidateShuffler[T] {

def rankToWeight(rank: Int): Double

def shuffle(seed: Option[Long])(input: Seq[T]): Seq[T] = {

val candWeights = input.zipWithIndex.map {

case (candidate, rank) => (candidate, rankToWeight(rank))

}

RandomUtil.weightedRandomShuffle(candWeights, seed.map(new Random(\_))).unzip.\_1

}

}

class ExponentialShuffler[T]() extends RankWeightedRandomShuffler[T] {

def rankToWeight(rank: Int): Double = {

1 / math

.pow(rank.toDouble, 2.0) // this function was proved to be effective in previous DDGs

}

}