use std::collections::HashMap;

#[derive(Debug)]

pub struct FeatureInfo {

pub tensor\_index: i8,

pub index\_within\_tensor: i64,

}

pub static NULL\_INFO: FeatureInfo = FeatureInfo {

tensor\_index: -1,

index\_within\_tensor: -1,

};

#[derive(Debug, Default)]

pub struct FeatureMapper {

map: HashMap<i64, FeatureInfo>,

}

impl FeatureMapper {

pub fn new() -> FeatureMapper {

FeatureMapper {

map: HashMap::new(),

}

}

}

pub trait MapWriter {

fn set(&mut self, feature\_id: i64, info: FeatureInfo);

}

pub trait MapReader {

fn get(&self, feature\_id: &i64) -> Option<&FeatureInfo>;

}

impl MapWriter for FeatureMapper {

fn set(&mut self, feature\_id: i64, info: FeatureInfo) {

self.map.insert(feature\_id, info);

}

}

impl MapReader for FeatureMapper {

fn get(&self, feature\_id: &i64) -> Option<&FeatureInfo> {

self.map.get(feature\_id)

}

}