**COMP 3002 Winter 2020 Assignment #4**

**Building the class Relation**

**Due: Tuesday Midnight Feb 9.**

**Basic goals**: To implement the class Relation described in the notes.

**What’s not obvious…**

Even though the class is known to contain triples, it’s not obvious how to store the triples. You could store them in an ordered collection or maybe in a dictionary. Just be prepared to change your mind as you write the code. Nothing wrong with changing your mind.

One nice way to work is to create example class methods that can be used to show that the class works. For example,

 example1

 "Relation example1"

 | collection relation |

 "First, build a relation."

 collection := #((2 < 3) (1 = 1) (3 > 1) (2 < 4)(1 < 5) (5 < 6) (2 < 5)).

 relation := Relation new.

 collection do: [:triple | relation addTriple: triple].

 "Second, show that the 3-parameter do: works…"
 Transcript cr; << 'Let relation = '; << collection.
 Transcript cr; << 'The created relation is '.

 relation do: [:a :b :c |Transcript ' <'; << a; space; << b; space; << c; <'>']]

 example2

 "Relation example2"

 | collection relation |

 "First, build a relation."

 collection := #((2 < 3) (1 = 1) (3 > 1) (2 < 4)(1 < 5) (5 < 6) (2 < 5)).

 relation := Relation new.

 collection do: [:triple | relation addTriple: triple].

 "Second, show that the 1-parameer do: and from:do: works…"
 Transcript cr; << 'Let relation = '; << collection.
 Transcript cr; << 'Starting from {1 2 3},'.

 relation from: #(1 2 3) do: [:relationship :subrelation |
 Transcript cr; << 'There is a relationship '; << relationship;
 << ' with subrelation'; cr; tab.
 subrelation do: [:triple |Transcript space; << triple]]

Hint: Would separate: or partitionsUsing: be of any use!

Also implement the class methods involving performItemStar and performRelationStar, etc. Use the example in the notes to prove that it works.

Addendum

I found 2 mistakes in my notes… The first one is that “items” is a parameter, not a message on the last line.



So it has to be

using: relation performItemStar: items
 | newRelation |
 newRelation := self using: relation performRelationStar: items.

 ^newRelation allTo addAllIfAbsent: items

The second one has been revised to

**using: relation performRelationOneStep: items**

 **"Builds new triples by applying 'relation' ONCE to each item in items. Returns the triples in a totally new relation."**

 **| newRelation tos |
 newRelation := Relation new. tos = OrderedCollection new.**

 **relation from: items do: [:relationship :subrelation |
 subrelation do: [:triple | tos addifAbsent: triple third]].**

 **relation from: tos do: [:relationship :subrelation |
 subrelation do: [:triple | newRelation addTriple: triple]].**

 **^newRelation**