Звичайний текстовий опис розрідженої матриці обмежений, тому на порятунок приходить графічна репрезентація матриць.

Раніше ми розглядали, візуалізували їх структуру, пригадаємо.

Roassal – це візуалізаційний програмний засіб, написаний на smalltalk, для візуалізації та взаємодії з довільними даними. Відображення задаємо об’єктами і зв’язками між ними.

*Приклад композиції на 5*

Smalltalk – рефлексивна мова програмування. Це означає, що програми здатні «відобразити» своє власне виконання та структуру. З програмної точки зору це означає, що метаоб’єкти системи виконання можна змінювати як звичайні об’єкти, які можна зчитувати і перевіряти. Це – інтроспекція, вона підтримується багатьма сучасними мовами програмування.

І навпаки, у Smalltalk можна модифікувати метаоб’єкти та відобразити ці зміни назад у систему виконання. Це також називається заступництвом (або посередництвом) і підтримується головним чином мовами динамічного програмування і лише в дуже обмеженій мірі статичними мовами.

Reflectivity - це пакет, який розширює звичайні можливості рефлексії Pharo. Він містить класи, за допомогою яких можна «прив’язатися» до будь-якого вузла абстрактного синтаксичного дерева. Іншими словами – він дозволяє виконувати довільний код при виконанні певного елемента коду: методу, змінної, змінної класу тощо.

Клас MetaLink пакету Reflectivity – це одна з основних складових цілої системи, яка розширює можливості рефлексії в Pharo. За допомогою цього класу та інших допоміжних компонент пакету можна додавати анотації до вузлів абстрактного синтаксичного дерева. Анотований вузол розкривається, компілюється і виконується на льоту.

Для певних вузлів метапосилання можна встановити до, після чи замість виконання цього вузла. Також існує можливість прив’язки до успішного чи помилкового виконання вузла, але не для всіх вузлів доступні такі анотації. Наприклад, для літералів не можна додати метапосилання на успішне чи неуспішне виконання.

Рядки 2-4 створюють і налаштовують нове метапосилання наступним чином: об’єкту, який ми передали в metaObject: (звичайний вивід дружнього повідомлення в Transcript) буде надіслано повідомлення передане в selector: після виконання вузла, до якого буде додано це посилання. Рядок 6 створює новий об’єкт instance класу ReflectivityExamples2. Рядки 7-10 використовують метод link:toClassVariableNamed:option: для прив’язки нашого метапосилання до запису в змінну класу classVar. Рядки 12-15 демонструють можливість додавання анотації лише до вузлів певного екземпляра класу ReflectivityExamples2.

Інакше кажучи, при першому прикладі встановлення анотації вивід в Transcript буде відбуватися кожного разу, коли хтось буде записувати нове значення в змінну класу classVar. У другому прикладі – вивід відбудеться лише при умові, що запис у змінну буде здійснюватися екземпляром класу instance.