1 модуль ontology.py

Здесь хранится классы для хранения и взаимодействия с онтологией

1.1 класс Ontology

этот класс хранит и обрабатывает онтологию.
Ограничения на онтологию:

- все классы имеют различные названия
- все отношения исходящие из 1 класса имеют одинаковое название
- все сущности из одного класса имеют различные названия
- фиксированное отношение между фиксированными сущностями может существовать только в единственном виде(без множественности)

1. def __init__(self, fileName=None)
   - fileName - файл с сохраненной онтологией с папке ./saved/

   конструктор, создающий пустую онтологию при fileName=None
2. def addClass(self,className)
   - className - название класса

   добавление нового класса, с генерацией ошибки при существовании добавляемого класса
3. def changeClass(self,oldClassName, newClassName)
   - oldClassName - старое название класса
   - newClassName - новое название класса

   изменение названия класса, с генерацией ошибки при существовании нового класса или отсутствии старого класса
4. def deleteClass(self,className,deep=False)
   - className - название класса
   - deep - метка для глубокого удаления

   удаление класса, с генерацией ошибки при несуществовании класса, при глубоком удалении со всеми связями на него и из него и с генерацией ошибки при наличии связей
5. def addRelationship(self, relationshipName, outClassName, inClassName)
   - relationshipName - название отношения
   - outClassName - название выходного класс
   - inClassName - название входного класс

   создание отношения, с генерацией ошибки при несуществовании одного из классов или уже существовании отношения

6. def changeRelationship(self, oldRelationshipName, newRelationshipName, outClassName)

   - oldRelationshipName - новое название отношения
   - newRelationshipName - старое название отношения
   - outClassName - название выходного класса

   изменение названия отношения, с генерацией ошибки при несуществовании старого отношения или существования нового отношения или несуществовании выходного класса

7. def deleteRelationship(self, relationshipName, outClassName, deep=False)

   - relationshipName - название отношения
   - outClassName - название выходного класса
   - deep - метка дял глубокого удаления

   удаление отношения с генерацией ошибки при несуществовании отношения или несуществовании выходного класса, с удалением свех связанных связей при глубоком удалении и генерацией ошибки при наличии связей при не глубоком удалении

8. def addEntity(self, entityName, className)

   - entityName - название сущности
   - className - название класса

   создание сущности класса, с генерацией ошибки при существовании сущности или несуществовании класса

9. def changeEntity(self, oldEntityName, newEntityName, className)

   - oldEntityName - название старой сущности
   - newEntityName - название новой сущности
   - className - название класса

   изменение названия сущности класса, с генерацией ошибки при не существовании старой сущности или существовании новой сущности или не существовании класса

10. def deleteEntity(self, entityName, className, deep=False)

    - entityName - название сущности
    - className - название класса
    - deep - метка для глубокого удаления

    удаление сущности класса, с генерацией ошибки при не существовании сущности или не существовании класса, с удалением всех связей при глубоком удалении и генерацией ошибки при наличии связей при не глубоком удалении

11. def addEntityRelationship(self, relationshipName, outClassName, outEntityName, inEntityName)

    - relationshipName - название отношения
    - outClassName - название выходного класса
    - outEntityName - название выходной сущности
    - inEntityName - название входной сущности

добавление связи между сущностями, с генерацией ошибки при несуществовании отношения класса или не существовании выходного класса или выходной сущности или входной сущности или существования отношения между сущностями

12. def deleteEntityRelationship(self, relationshipName, outClassName, outEntityName, inEntityName

    - relationshipName - название отношения
    - outClassName - название выходного класса
    - outEntityName - название выходной сущности
    - inEntityName - название входной сущности

    удаление связи между сущностями, с генерацией ошибки при несуществовании отношения класса или не существовании выходного класса или выходной сущности или входной сущности или не существования отношения между сущностями

13. def getAllClasses(self)

    - return - список с названиями классов

    получение списка классов

14. def getAllRelationshipsForOutClass(self, className)

    - className - название класса
    - return - список с названиями отношений

    получение все отношений класса, с генерацией ошибки при не существовании класса

15. def getAllEntitiesForClass(self, className)

    - className - название класса
    - return - список с названиями сущностей класса

    получение всех сущностей класса, с генерацией ошибки при не существовании класса

16. def getAllRelationshipedEntitiesForOutEntity(self, outEntityName, outClassName)

    - outEntityName - название сущности класса
    - outClassName - название выходного класса
    - return - список кортеджей вида (название класса, название сущности класса)

    получение всех сущностей класса связанных с заданной сущностью, с генерацией ошибки при не существования заданной сущности класса или несуществовании выходного класса

17. def getAllRelationshipedEntitiesForOutEntityForInClass(self, outEntityName, outClassName, inClassName)

    - outEntityName - название сущности класса
    - outClassName - название выходного класса
    - inClassName - название входного класса
    - return - список кортеджей вида (название класса, название сущности класса)

    получение всех сущностей класса связанных с заданной сущностью из заданного входного класса, с генерацией ошибки при не существования заданной сущности класса или несуществовании выходного класса или не существовании входного класса

18. getAllRelationshipedEntitiesForOutEntityForRelationship(self, outEntityName, outClassName, relationshipName)

    - outEntityName - название сущности класса
    - outClassName - название выходного класса
    - relationshipName - название отношения
    - return - список кортеджей вида (название класса, название сущности класса)

    получение всех сущностей класса связанных с заданной сущностью заданным отношением, с генерацией ошибки при не существования заданной сущности класса или несуществовании выходного класса или не существовании отношения

19. def saveToFile(self, fileName)

    - fileName - название файла

    сохранение онтологии в файл с папке /saved/

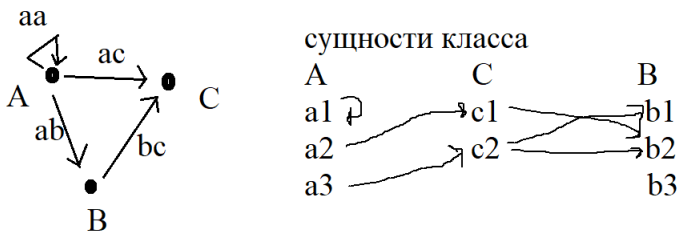
20. def loadFromFile(self, fileName)

    - fileName - название fileName

    загрузка и обновление текущей онтологии из файла с папке /saved/

21. def clear(self) отчистка онтологии (удаление всех данных)

Пример использования на примере онтологии, изображенной на рисунке




```
from ontology import *

ont=Ontology()
ont.addClass('A')
ont.addClass('B')
ont.addClass('CC')
ont.addClass('D')
ont.changeClass('CC','C')
ont.deleteClass('D', True)
print(ont.getAllClasses())# [A, B, C]

ont.addRelationship('aa','A','A')
ont.addRelationship('ab','A','B')
ont.addRelationship('ac','A','C')
```

```
ont.addRelationship('cb_','C','B')
ont.addRelationship('bb','B','B')
ont.changeRelationship('cb_','cb','C')
ont.deleteRelationship('bb','B', False)

ont.addEntity('a1','A')
ont.addEntity('a2','A')
ont.addEntity('b1','B')
ont.addEntity('b2','B')
ont.addEntity('b3','B')
ont.addEntity('c1','C')
ont.addEntity('c2_','C')
ont.addEntity('c3','C')
ont.changeEntity('c2_','c2','C')
ont.deleteEntity('c3','C')

ont.addEntityRelationship('aa','A','a1','a1')
ont.addEntityRelationship('ac','A','a2','c1')
ont.addEntityRelationship('ac','A','a2','c2')
ont.addEntityRelationship('cb','C','c1','b2')
ont.addEntityRelationship('cb','C','c2','b2')
ont.addEntityRelationship('cb','C','c2','b1')
ont.addEntityRelationship('cb','C','c2','b3')
ont.deleteEntityRelationship('cb','C','c2','b3')

print(ont.getAllClasses())# [A, B, C]
print(ont.getAllRelationshipsForOutClass('A'))# [aa, ab, ac]
print(ont.getAllRelationshipsForOutClass('B'))# []
print(ont.getAllRelationshipsForOutClass('C'))# [cb]
print(ont.getAllRelationshipsForOutClassToInClass('A','B'))# [ab]
print(ont.getAllRelationshipsForOutClassToInClass('A','A'))# [aa]
print(ont.getAllRelationshipsForOutClassToInClass('C','A'))# []
print(ont.getAllEntitiesForClass('A'))# [a1, a2]
print(ont.getAllEntitiesForClass('B'))# [b1, b2, b3]
print(ont.getAllEntitiesForClass('C'))# [c1, c2]
print(ont.getAllRelationshipedEntitiesForOutEntity('a1','A'))# [(A,a1)]
print(ont.getAllRelationshipedEntitiesForOutEntity('a2','A'))# [(C,c1), (C,c2)]
print(ont.getAllRelationshipedEntitiesForOutEntity('b1','B'))# []
print(ont.getAllRelationshipedEntitiesForOutEntityForInClass('a2','A','C'))# [(C,c1), (C,c2)]
print(ont.getAllRelationshipedEntitiesForOutEntityForInClass('a2','A','B'))# []
print(ont.getAllRelationshipedEntitiesForOutEntityForRelationship('a2','A','ac'))# [(C,c1), (C,c2)]
print(ont.getAllRelationshipedEntitiesForOutEntityForRelationship('a2','A','ab'))# []
ont.saveToFile('ont1.txt')

ont.loadFromFile('ont1.txt')
#...
ont.crear()
ont=Ontology('ont1.txt')
#...
```