**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Новгородский государственный университет имени Ярослава Мудрого»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра ИТиС

«Поисковая система на основе фреймовой модели представления знаний»

Лабораторная работа №3

По дисциплине: Системы искусственного интеллекта

Отчет

Выполнил: студент гр. 8091

\_\_\_\_\_\_\_ \_\_\_\_Кудряшов И.С.

Проверил: Преподаватель

\_\_\_\_\_\_\_\_\_\_\_Михайлов Д.В.

Великий Новгород

2021

1. **Формулировка цели и задач**

Рассмотреть представление знаний фреймами и соответствующие способы управления выводом.

1. **Модель представления знаний**

Предметная область виды спорта. Их я разбил на оздоровительные и соревновательные.

Фреймом называется структура для описания понятия или ситуации, состоящая из характеристик этой ситуации и их значений.

Особенность фреймовых моделей представления знаний состоит в том, что все понятия, описываемые в каждом из узлов модели, определяются набором атрибутов и их значениями, которые содержатся в слотах фрейма.

Слот — это атрибут, который связывается с узлом в системе, основанной на фреймах. Он является основной составляющей фрейма. Он имеет имя слота и значение слота.

С каждым слотом может быть связана одна или несколько процедур, которые выполняются, когда изменяются значения слотов. Чаще всего со слотами связываются процедуры:

ЕСЛИ-ДОБАВЛЕНО (IF-ADDED) — выполняется в том случае, когда новая информация помещается в слот;

ЕСЛИ-УДАЛЕНО (IF-REMOVED) — выполняется при удалении информации из слота;

ЕСЛИ-НУЖНО (IF-NEEDED) — выполняется при запросе информация из слота в том случае, когда слот пуст.

В моей работе реализованы только процедуры IF-ADDED.

Эти процедуры следят за изменением информации, приписываемой данному узлу, и проверяют, что при изменении тех или иных значений производятся соответствующие действия.

Фреймовые системы позволяют использовать объектно-ориентированный подход к представлению знаний.

Таблица 1. Фрейм "Спорт"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| Название | U(Unique) | Text |  |  |
| Сезон | U(Unique) | Text | “Лето”, “Зима”, “Круглогодичные” |  |
| Необходимый возраст | U(Unique) | Integer |  | Проверка 2 <= возраст <=100 |

Таблица 2. Фрейм "Оздоровительный"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Спорт |  |
| Название | S(Same) | Text |  |  |
| Сезон | S(Same) | Text | “Лето”, “Зима”, “Круглогодичные” |  |
| Необходимый возраст | S(Same) | Integer |  | Проверка 2 <= возраст <=90 |
| Необходимо присутствие тренера | U(Unique) | Bool |  |  |
| Назначение | U(Unique) | Text |  |  |

Таблица 3. Фрейм "Соревновательный"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Спорт |  |
| Название | S(Same) | Text |  |  |
| Сезон | S(Same) | Text | “Лето”, “Зима”, “Круглогодичные” |  |
| Необходимый возраст | S(Same) | Integer |  | Проверка 2 <= возраст <= 90 |
| Количество участников | U(Unique) | Integer |  | Проверка 2 < участников <=30 |
| Наличие непосредственной борьбой и соприкосновением с противником | U(Unique) | Bool |  |  |
| Цель спорта | U(Unique) | Text |  |  |

Таблица 4. Фрейм "Плавание"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Оздоровительный |  |
| Название | U(Unique) | Text | Плавание |  |
| Сезон | U(Unique) | Text | “Круглогодичные” |  |
| Необходимый возраст | U(Unique) | Integer | 3 | Проверка 2 <= возраст <=90 |
| Необходимо присутствие тренера | U(Unique) | Bool | Нет |  |
| Назначение | U(Unique) | Text | Позвоночник и общеоздоровительное |  |

Таблица 5. Фрейм "Единоборства"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Соревновательный |  |
| Название | U(Unique) | Text | Единоборства |  |
| Сезон | U(Unique) | Text | “Круглогодичные” |  |
| Необходимый возраст | U(Unique) | Integer | 5 | Проверка 2 <= возраст <= 90 |
| Количество участников | U(Unique) | Integer | 2 | Проверка 2 < участников <=30 |
| Игра с непосредственной борьбой и соприкосновением с противником | U(Unique) | Bool | Да |  |
| Цель спорта | U(Unique) | Text | Признание поражения одной из сторон, либо нарушение правил одной из сторон |  |

1. **Механизм управления вводом**

Фреймовые системы позволяют использовать объектно-ориентированный подход к представлению знаний. Эта структура базируется на отношениях IS-А между фреймами, описывающими некоторое мобильное устройство.

При этом наши представления и знания об объекте "Спорт" детализируем рядом понятий, которые определим через набор атрибутов и возможными типы их значений. Все атрибуты разместим в слотах фрейма.

Таким образом, на самом верхнем уровне иерархии определен фрейм “Спорт”. В данном случае спорт разделяется на оздоровительный и соревновательный, которые определены в качестве соответствующих дочерних фреймов.

Фреймы, описывающие различные объекты называются шаблонами, а фреймы верхнего уровня, использующиеся для представления этих шаблонов, называются фреймами класса.

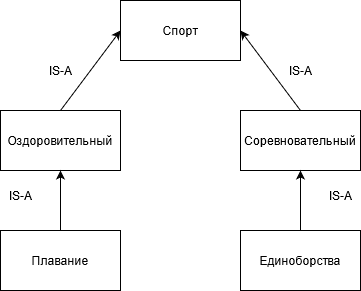


Рисунок 1 Структура фреймов

1. **Описание механизма поиска по запросу в поисковой системе**

В этой лабораторной работе предусмотрен поиск информации об определённом спорте, это происходит путем ввода названия спорта, и программа предоставляет основную информацию об этом спорте.

1. **Тестовые наборы для всех функций преобразования над знаниями в системе**

**На вход подается название вида спорта:**

Вводим: Плавание

Ответ:

название: Плавание

Сезон: Круглогодичные

Необходимый возраст: 2

Необходимо присутствие тренера: нет

Назначение: Позвоночник и общеоздоровительное

**Неудачный набор данных:**

Является ситуация при которой мы введем название спорта, которого нет в базе:

Вводим: Велосипед

Ответ: База знаний не содержит сведений о данном виде спорта

**Также при добавлении новой модели если она уже есть в базе:**

Вводим: Плавание

Ответ: Запись с таким именем уже есть в базе данных

**Демонстрация работы демона-процедуры IF-ADDED:**

Выберите действие(найти, добавить, удалить): добавить

Вид спорта(оздоровительный, соревновательный): оздоровительный

название: Плавание

Сезон: 123

Необходимый возраст: 2

Необходимо присутствие тренера: нет

Назначение: Позвоночник и общеоздоровительное

Допустимы сезоны Лето Зима Круглогодичные

**Вывод**

Были приобретены умения и навыки реализации пополняемой динамической базы знаний, основанной на фреймовой системе представления знаний. И также был разработан редактор фреймов, с помощью которого можно добавлять и удалять фреймы.

**Приложение:**

**Код программы**

import inspect

from abc import ABC

def dbool(bool\_str: str):

    return bool\_str == 'True'

class Sport(ABC):

    def \_\_init\_\_(self, name: str, season: str, age: int):

        self.\_\_name = name

        self.season = season

        self.age = int(age)

    def get\_type(self):

        raise NotImplementedError(

            "Пожалуйста не используйте экземпляры базового класса")

    def \_\_str\_\_(self):

        return ','.join([self.get\_type(), \*map(str, self.\_\_dict\_\_.values())])

    def deserialize(device\_str: str):

        str\_parts = device\_str.split(',')

        if str\_parts[0] == 'Healfull':

            return Healfull(str\_parts[1], str\_parts[2], int(str\_parts[3]), dbool(str\_parts[4]), str\_parts[5])

        else:

            return Competitive(str\_parts[1], str\_parts[2], int(str\_parts[3]), int(str\_parts[4]), dbool(str\_parts[5]), str\_parts[6])

    @property

    def name(self):

        return self.\_\_name

    @property

    def season(self):

        return self.\_\_season

    @season.setter

    def season(self, season: str):

        season\_LIST = ['Лето', 'Зима', 'Круглогодичные']

        if season in season\_LIST:

            self.\_\_season = season

        else:

            raise ValueError('Допустимы сезоны '+' '.join(season\_LIST))

    @property

    def age(self):

        return self.\_\_age

    @age.setter

    def age(self, age: int):

        if age <= 2:

            raise ValueError("Возраст не может быть <= 2")

        if age > 90:

            raise ValueError(

                "Возраст спротсмена не может быть > 90")

        self.\_\_age = age

class Healfull(Sport):

    def \_\_init\_\_(self, name: str, season: str, age: int, trainer\_required: bool, Purpose: str):

        super().\_\_init\_\_(name, season, age)

        self.\_\_trainer\_required = trainer\_required

        self.\_\_Purpose = Purpose

    def get\_type(self):

        return "Healfull"

    @ property

    def trainer\_required(self):

        return self.\_\_trainer\_required

    @ property

    def Purpose(self):

        return self.\_\_Purpose

class Competitive(Sport):

    def \_\_init\_\_(self, name: str, season: str, age: int, participants: int, conflict: bool, taget: str):

        super().\_\_init\_\_(name, season, age)

        self.participants = int(participants)

        self.\_\_conflict = conflict

        self.\_\_taget = taget

    def get\_type(self):

        return "Competitive"

    @ property

    def participants(self):

        return self.\_\_participants

    @participants.setter

    def participants(self, participants: int):

        if participants < 2:

            raise ValueError("Число участников не может быть меньше 2")

        if participants > 30:

            raise ValueError(

                "Число участников не может быть больше 30")

        self.\_\_participants = participants

    @ property

    def conflict(self):

        return self.\_\_conflict

    @ property

    def taget(self):

        return self.\_\_taget

def parse\_db():

    db = []

    with open('db.txt', 'r', encoding="utf-8") as db\_file:

        for line in db\_file.readlines():

            if len(line) > 0:

                db.append(Sport.deserialize(line.strip("\r\n ")))

    return db

def save\_db():

    with open('db.txt', 'w', encoding="utf-8") as db\_file:

        db\_file.write('\n'.join(map(str, db)))

def add\_entry(e: Sport):

    if find\_entry(e.name) is None:

        db.append(e)

        save\_db()

    else:

        raise ValueError(f"Запись с таким именем уже есть в базе данных")

def find\_entry(name: str):

    for e in db:

        if e.name == name:

            return e

    return None

def remove\_entry(name: str):

    e = find\_entry(name)

    if e is not None:

        db.remove(e)

        save\_db()

    else:

        raise ValueError(f"Запись с таким именем не найдена в базе данных")

db = parse\_db()

def bool\_to\_ru(e):

    if type(e) == bool:

        return 'да' if e else 'нет'

    return e

while True:

    command = None

    en\_to\_ru = {

        "name": "название",

        "season": "Сезон",

        "age": "Необходимый возраст",

        "trainer\_required": "Необходимо присутствие тренера",

        "Purpose": "Назначение",

        "participants": "Количество участников",

        "conflict": "Наличие непосредственной борьбой и соприкосновением с противником",

        "taget": "Цель спорта"

    }

    while command not in ['найти', 'добавить', 'удалить']:

        command = input("Выберите действие(найти, добавить, удалить): ")

    if command == 'найти':

        e = find\_entry(input('название: '))

        if e is not None:

            for key in e.\_\_dict\_\_:

                print(

                    f"{en\_to\_ru[key.split('\_\_')[1]]}: {bool\_to\_ru(e.\_\_dict\_\_[key])}")

        else:

            print(f"База знаний не содержит сведений о данном виде спорта")

    elif command == 'добавить':

        try:

            dclass = None

            while dclass not in ['оздоровительный', 'соревновательный']:

                dclass = input(

                    "Вид спорта(оздоровительный, соревновательный): ").lower()

            if dclass == "оздоровительный":

                values = []

                for key in list(inspect.signature(Healfull.\_\_init\_\_).parameters)[1:]:

                    values.append(input(f"{en\_to\_ru[key]}: "))

                for i in range(len(values)):

                    if values[i].lower() == 'да':

                        values[i] = True

                    elif values[i].lower() == 'нет':

                        values[i] = False

                add\_entry(Healfull(\*values))

            else:

                values = []

                for key in list(inspect.signature(Competitive.\_\_init\_\_).parameters)[1:]:

                    values.append(input(f"{en\_to\_ru[key]}: "))

                for i in range(len(values)):

                    if values[i].lower() == 'да':

                        values[i] = True

                    elif values[i].lower() == 'нет':

                        values[i] = False

                add\_entry(Competitive(\*values))

        except ValueError as e:

            print(e)

    else:

        try:

            remove\_entry(input('название: '))

        except ValueError as e:

            print(e)

**Содержимое db.txt**

Healfull,Плавание,Круглогодичные,3,False,Позвоночник и общеоздоровительное

Competitive,Единоборства,Круглогодичные,5,2,True,Признание поражения одной из сторон, либо нарушение правил одной из сторон