МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Новгородский государственный университет имени Ярослава Мудрого (НовГУ) Кафедра «Электронных и информационных систем»

**Поисковая система на основе фреймовой модели представления знаний**

Лабораторная работа по дисциплине:

«Системы искусственного интеллекта»

Студент гр. 1095

Золотов Э. В.

(подпись)

«\_\_\_»\_ 2024 г.

Преподаватель

\_\_Михайлов Д.В

(подпись)

« » 2024 г.

г. Великий Новгород

-2024-

# ЦЕЛЬ РАБОТЫ

Рассмотреть представление знаний фреймами и соответствующие способы управления выводом.

# МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Предметная область профессии. Данные профессии разделены на группы: технические, гуманитарные.

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, IF-REMOVED.

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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| Название | U(Unique) | Text |  |  |
| Зарплата | U(Unique) | Text | “Высокая”, “Средняя”, “Низкая” |  |
| Опыт | U(Unique) | Integer |  | Проверка 2 <= опыт <= 60 |

***Таблица 1. Фрейм “Профессия”***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Профессия |  |
| Название | S(Same) | Text |  |  |
| Зарплата | S(Same) | Text | “Высокая”, “Средняя”, “Низкая” |  |
| Опыт | U(Unique) | Integer |  | Проверка 2 <= опыт <= 60 |
| Необходимо знание математики | U(Unique) | Bool |  |  |
| Роль | U(Unique) | Text |  |  |

***Таблица 2. Фрейм “Технический”***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Профессия |  |
| Название | S(Same) | Text |  |  |
| Зарплата | S(Same) | Text | “Высокая”, “Средняя”, “Низкая” |  |
| Опыт | U(Unique) | Integer |  | Проверка 2 <= опыт <= 60 |
| Необходимо умение рисовать | U(Unique) | Bool |  |  |
| Возраст | U(Unique) | Integer |  | Проверка 40 < возраст <= 60 |
| Роль | U(Unique) | Text |  |  |

***Таблица 3. Фрейм “Гуманитарный”***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Профессия |  |
| Название | S(Same) | Text |  |  |
| Зарплата | S(Same) | Text | “Высокая” |  |
| Опыт | U(Unique) | Integer | 3 | Проверка 2 <= опыт <= 60 |
| Необходимо знание математики | U(Unique) | Bool | Да |  |
| Роль | U(Unique) | Text | Аналитика данных |  |

***Таблица 4. Фрейм “Программист”***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя слота | Указатель наследования | Тип данных | Значение слота | Демон |
| IS-A | U(Unique) | FRAME | Профессия |  |
| Название | U(Unique) | Text | Историк |  |
| Зарплата | U(Unique) | Text | “Высокая” |  |
| Опыт | U(Unique) | Integer | 5 | Проверка 2 <= опыт <= 60 |
| Необходимо умение рисовать | U(Unique) | Bool | Нет |  |
| Возраст | U(Unique) | Integer | 55 | Проверка 40 < возраст <= 60 |
| Роль | U(Unique) | Text | Исследования в области истории Азии |  |

***Таблица 5. Фрейм “Историк”***

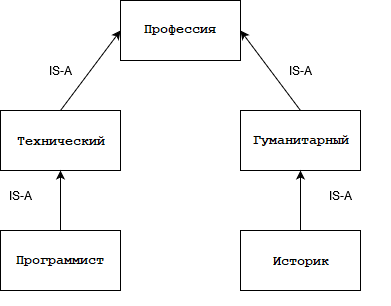
# МЕХАНИЗМ УПРАВЛЕНИЯ ВВОДОМ

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 "Профессия" 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 “Профессия”. В данном случае профессия разделяется на технический и гуманитарный профиль, 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.



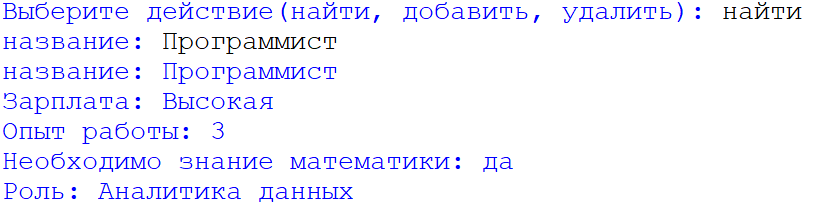
# ОПИСАНИЕ МЕХАНИЗМА ПОИСКА ПО ЗАПРОСУ В ПОИСКОВОЙ СИСТЕМЕ

В данной лабораторной работе предусмотрен поиск информации об определённой профессии, это происходит путем ввода названия профессии, и программа предоставляет основную информацию об этой профессии.

# ТЕСТОВЫЕ НАБОРЫ ДЛЯ ВСЕХ ФУНКЦИИ ПРЕОБРАЗОВАНИЯ НАД ЗНАНИЯМИ В СИСТЕМЕ

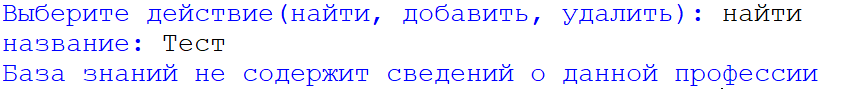
# *Поиск по названию профессии:*

На вход подается название профессии. Вводится “Программист”. Ответ системы:



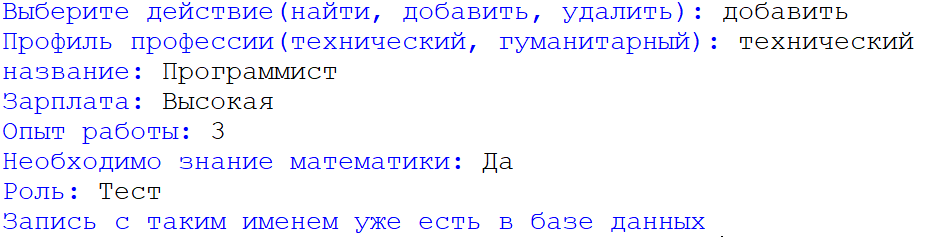
*Неудачный набор данных:*

Ввод названия, которого нет в базе:

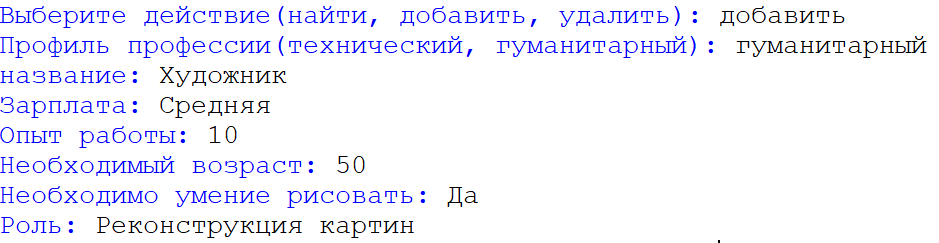


*Добавление профессии, которая уже существует в базе:*

Ввод “Программист”:



*Демонстрация работы демона-процедуры IF-ADDED:*



*Демонстрация работы демона-процедуры IF-REMOVED*

# 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 A

# Код программы

**import** **inspect**

**from** **abc** **import** ABC

**def** **dbool**(bool\_str: str):

**return** bool\_str == 'True'

**class** **Prof**(ABC):

**def** **\_\_init\_\_**(self, name: str, salary: str, exp: int):

self.\_\_name = name

self.salary = salary

self.exp = int(exp)

**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**] == 'Tech':

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

**else**:

**return** Humanitarian(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** **salary**(self):

**return** self.\_\_salary

**@salary**.setter

**def** **salary**(self, salary: str):

salary\_LIST = ['Высокая', 'Средняя', 'Низкая']

**if** salary **in** salary\_LIST:

self.\_\_salary = salary

**else**:

**raise** **ValueError**('Доступно:'+' '.join(salary\_LIST))

**@property**

**def** **exp**(self):

**return** self.\_\_exp

**@exp**.setter

**def** **exp**(self, exp: int):

**if** exp <= **2**:

**raise** **ValueError**("Опыт работы не может быть <= 2")

**if** exp > **60**:

**raise** **ValueError**("Опыт работы не может быть > 60")

self.\_\_exp = exp

**class** **Tech**(Prof):

**def** **\_\_init\_\_**(self, name: str, salary: str, exp: int, math\_required: bool, role: str):

super().\_\_init\_\_(name, salary, exp)

self.\_\_math\_required = math\_required

self.\_\_role = role

**def** **get\_type**(self):

**return** "Tech"

**@property**

**def** **math\_required**(self):

**return** self.\_\_math\_required

**@property**

**def** **role**(self):

**return** self.\_\_role

**class** **Humanitarian**(Prof):

**def** **\_\_init\_\_**(self, name: str, salary: str, exp: int, age: int, art: bool, role: str):

super().\_\_init\_\_(name, salary, exp)

self.age = int(age)

self.\_\_art = art

self.\_\_role = role

**def** **get\_type**(self):

**return** "Humanitarian"

**@property**

**def** **age**(self):

**return** self.\_\_age

**@age**.setter

**def** **age**(self, age: int):

**if** age < **40**:

**raise** **ValueError**("Возраст не может быть меньше 40")

**if** age > **60**:

**raise** **ValueError**(

"Возраст не может быть больше 60")

self.\_\_age = age

**@property**

**def** **art**(self):

**return** self.\_\_art

@ property

**def** **role**(self):

**return** self.\_\_role

**def** **parse\_db**():

db = []

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

**for** line **in** db\_file.readlines():

**if** len(line) > **0**:

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

**return** db

**def** **save\_db**():

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

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

**def** **add\_entry**(e: Prof):

**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",

"salary": "Зарплата",

"age": "Необходимый возраст",

"exp": "Опыт работы",

"math\_required": "Необходимо знание математики",

"art": "Необходимо умение рисовать",

"role": "Роль",

}

**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 данной профессии")

**elif** command == 'добавить':

**try**:

dclass = **None**

**while** dclass **not** **in** ['технический', 'гуманитарный']:

dclass = input(

"Профиль профессии(технический, гуманитарный): ").lower()

**if** dclass == "технический":

values = []

**for** key **in** list(inspect.signature(Tech.\_\_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(Tech(\*values))

**else**:

values = []

**for** key **in** list(inspect.signature(Humanitarian.\_\_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(Humanitarian(\*values))

**except** **ValueError** **as** e:

print(e)

**else**:

**try**:

remove\_entry(input('название: '))

**except** **ValueError** **as** e:

print(e)

# Файл dbase.txt

Tech,Программист,Высокая,**3**,**True**,Аналитика данных

Humanitarian,Историк,Высокая,**5**,**55**,**False**,Исследования в области истории Азии