

Задание 1: Создание и объединение файлов

```
```bash
echo -e 'Собаки\nКошки\nХомяки' > Домашние_животные.txt
echo -e 'Лошади\nВерблюды\nОсли' > Выючные_животные.txt
```
```

- Первая команда создаёт файл `Домашние_животные.txt` и заполнит его названиями домашних животных.

- Вторая команда создаёт файл `Выючные_животные.txt` с названиями выючных животных.

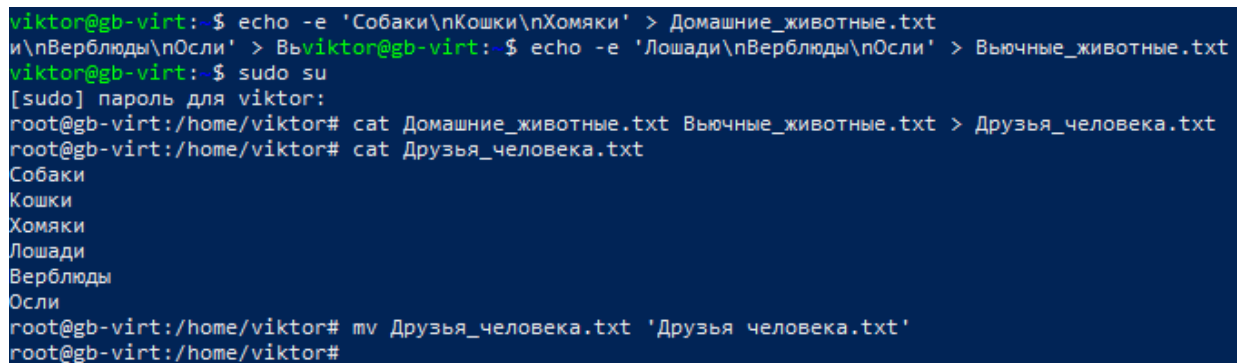
Объединить файлы в один:

```
```bash
cat Домашние_животные.txt Выючные_животные.txt > Друзья_человека.txt
```
```

- Эта команда создаст новый файл `Друзья_человека.txt`, который будет содержать все названия животных из обоих файлов.

Просмотра содержимого файла:

```
```bash
cat Друзья_человека.txt
```
```



```
viktor@gb-virt:~$ echo -e 'Собаки\nКошки\nХомяки' > Домашние_животные.txt
viktor@gb-virt:~$ echo -e 'Лошади\nВерблюды\nОсли' > Выючные_животные.txt
viktor@gb-virt:~$ sudo su
[sudo] пароль для viktor:
root@gb-virt:~# cat Домашние_животные.txt Выючные_животные.txt > Друзья_человека.txt
root@gb-virt:~# cat Друзья_человека.txt
Собаки
Кошки
Хомяки
Лошади
Верблюды
Осли
root@gb-virt:~# mv Друзья_человека.txt 'Друзья человека.txt'
root@gb-virt:~#
```

Переименование файла:

```
```bash
mv Друзья_человека.txt 'Друзья человека.txt'
```
```

- Теперь файл будет называться `Друзья человека.txt`.

Задание 2: Создание директории и перемещение файла

Создание директории:

```
```bash
mkdir Питомник
```
```

- Это создаст директорию с именем `Питомник`.

Перемещение файл `Друзья человека.txt` в созданную директорию:

```
```bash
```

```
mv 'Друзья человека.txt' Питомник/
````
```

Сохранение проекта на GitHub

1. ****Создание репозитория на GitHub****:

2. ****Клонирование репозитория****:

```
``bash
git clone https://github.com/WilhelMark/Kennel_Animal_Accounting.git
````
```

3. **\*\*Перемещение файлов в репозиторий\*\***:

```
``bash
mv ../Питомник . .
````
```

4. ****Добавление изменений****:

```
``bash
git add .
````
```

5. **\*\*Создание коммита\*\***:

```
``bash
git commit -m "Domestic and pack animal files added."
````
```

6. ****Отправка изменений на GitHub****:

```
``bash
git push origin main
````
```

P.S. Не все было так гладко, так как забыл настроить гит и выходили ошибки:

remote: Support for password authentication was removed on August 13, 2021. remote: Please see <https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls> for information on currently recommended modes of authentication. fatal: Authentication failed for 'https://github.com/WilhelMark/Kennel\_Animal\_Accounting.git/'

Ошибка, с которой мы столкнулись, связана с тем, что GitHub больше не поддерживает аутентификацию по паролю для операций с репозиториями. Вместо этого вам нужно использовать **\*\*персональные токены доступа\*\*** или **\*\*SSH-ключи\*\***. Вот как мы можем это сделать:

### ### Способ 1: Использование персонального токена доступа

1. **\*\*Создание персонального токена доступа\*\***:

- правом верхнем углу любой страницы щелкните фотографию своего профиля, затем нажмите Settings (Настройки).
- На левой боковой панели нажмите Developer settings (Настройки разработчика).
- На левой боковой панели щелкните Personal access tokens (Личные токены доступа).
- Щелкните Generate new token (Создать новый токен).

2. **\*\*Использование токена вместо пароля\*\***:

- При выполнении команд `git push`, `git clone` или `git pull`, когда Git запрашивает ваш пароль, введите созданный токен доступа.

### Способ 2: Использование SSH-ключей

1. **\*\*Генерация SSH-ключа\*\***:

- В терминале выполните следующую команду:

```
```bash
ssh-keygen -t rsa -b 4096 -C "vwilpilgrim@gmail.com"
```
```

- Нажать Enter для принятия стандартного имени файла и пути.

2. **\*\*Добавление SSH-ключа в ssh-agent\*\***:

- Запустить ssh-agent:

```
```bash
eval "$(ssh-agent -s)"
```
```

- Добавить SSH-ключ:

```
```bash
ssh-add ~/.ssh/id_rsa
```
```

3. **\*\*Добавление SSH-ключа в GitHub\*\***:

- Скопировать содержимое публичного ключа:

```
```bash
cat ~/.ssh/id_rsa.pub
```
```

- Перейти в настройки аккаунта на GitHub, затем в раздел **\*\*SSH and GPG keys\*\***.

- Нажмите на кнопку **\*\*"New SSH key"\*\***, вставьте ключ и дайте ему имя.

4. **\*\*Использование SSH для операций с репозиториями\*\***:

- Изменить URL удаленного репозитория на SSH:

```
```bash
git remote set-url origin git@github.com:WilhelMark/Kennel_Animal_Accounting.git
```
```

Теперь мы сможем выполнять команды `git push`, `git pull` и другие без запроса пароля, используя либо токен доступа, либо SSH-ключи.

### Задание 3: Подключение дополнительного репозитория MySQL и установка пакета

#### Добавление репозитория MySQL:

Для версий Ubuntu 20.04 и выше. Выполнить команду

```
```bash
sudo apt install mysql-server
```
```

. Это установит все пакеты, включая MySQL Client, MySQL Server и другие необходимые пакеты.

Для версий Ubuntu ниже 20.04. Сначала загрузить файл .deb репозитория MySQL с помощью к

```
```bash
wget https://dev.mysql.com/get/mysql-apt-config_0.8.26-1_all.deb
```
```

- Эта команда скачает конфигурационный пакет, который позволит нам добавить репозиторий MySQL в нашу систему.

### 3. \*\*Установить конфигурационный пакет\*\*:

После загрузки выполнить следующую команду:

```
```bash
sudo dpkg -i mysql-apt-config_0.8.26-1_all.deb
```
```

- При установке нам будет предложено выбрать версию MySQL и другие компоненты.

### 4. \*\*Обновить список пакетов\*\*:

После добавления репозитория обновить список доступных пакетов:

```
```bash
sudo apt update
```
```

#### Установка пакета из репозитория

### 5. \*\*Установите MySQL сервер и клиент\*\*:

Теперь вы можете установить MySQL, выполнив следующую команду:

```
```bash
sudo apt install mysql-server mysql-client
```
```

- Это установит сервер и клиент MySQL на наш компьютер.

### 6. \*\*Проверим установку\*\*:

Чтобы убедиться, что MySQL установлен правильно, выполним команду:

```
```bash
systemctl status mysql
```
```

- Мы должны увидеть сообщение о том, что служба MySQL активна и работает.

```
root@gb-virt:/home/viktor/Kennel_Animal_Accounting# systemctl status mysql
● mysql.service - MySQL Community Server
 Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
 Active: active (running) since Sun 2024-10-06 19:45:16 MSK; 9min ago
 Process: 14090 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
 Main PID: 14099 (mysqld)
 Status: "Server is operational"
 Tasks: 37 (limit: 4615)
 Memory: 365.2M (peak: 379.5M)
 CPU: 6.651s
 CGroup: /system.slice/mysql.service
 └─14099 /usr/sbin/mysqld

окт 06 19:45:15 gb-virt systemd[1]: Starting mysql.service - MySQL Community Server...
окт 06 19:45:16 gb-virt systemd[1]: Started mysql.service - MySQL Community Server.
```

### ### Задание 4: Установка и удаление deb-пакета с помощью dpkg

#### #### Установка deb-пакета:

1. **\*\*Скачаем любой deb-пакет\*\***:

Для примера загрузим пакет `curl`:

```
``bash
wget http://archive.ubuntu.com/ubuntu/pool/main/c/curl/curl_7.68.0-1ubuntu2_amd64.deb
``
```

2. **\*\*Установим deb-пакет с помощью dpkg\*\***:

После загрузки выполним команду:

```
``bash
sudo dpkg -i curl_7.68.0-1ubuntu2_amd64.deb
``
```

3. **\*\*Проверим установку\*\***:

Чтобы убедиться, что `curl` установлен, выполним команду:

```
``bash
curl --version
``
```

```
root@gb-virt:/home/viktor/Kennel_Animal_Accounting# curl --version
curl 7.68.0 (x86_64-pc-linux-gnu) libcurl/8.5.0 OpenSSL/3.0.13 zlib/1.3 brotli/1.1.0 zstd/1.5.5 libidn2/2.3.7 libpsl/0.21.2 (+libidn2/2.3.7) libssh/0.10.6/openssl/zlib nghttp2/1.59.0 librtmp/2.3 OpenLDAP/2.6.7
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtmpe rtmps rtmpt rtmpte rtmpts rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli ESNI GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM PSL SPNEGO SSL TLS-SRP UnixSockets
```

#### #### Удаление deb-пакета

4. **\*\*Удалим установленный пакет\*\***:

Если вам нужно удалить пакет, используйте следующую команду:

```
``bash
sudo dpkg -r curl
``
```

5. **\*\*Проверим удаление\*\***:

```
``bash
dpkg -l | grep curl
``
```

### ### Задание 5: Выложить историю команд в терминале Ubuntu

Просмотр истории команд:

```
``bash
history
``
```

- Эта команда выведет список всех команд, которые вы вводили в текущем сеансе терминала.

Каждая команда будет иметь свой номер.

```
root@gb-virt:/home/viktor/Kennel_Animal_Accounting# history
```

```
1 sudo apt update
```

```

2 sudo apt upgrade
3 sudo apt install apt-transport-https ca-certificates curl software-properties-common
4 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
5 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -
cs) stable"
6 sudo apt update
7 sudo apt install docker-ce
8 mkdir -p ~/myproject
9 cd ~/myproject
10 version: '3'
11 services:
12 echo -e 'version: "3"\n\nservices:\n web:\n image: nginx:latest\n ports:\n - "80:80"\n volumes:\n
n - ./html:/usr/share/nginx/html\n\n db:\n image: mysql:latest\n environment:\n
MYSQL_ROOT_PASSWORD: root\n MYSQL_DATABASE: mydb' > docker-compose.yml
13 mkdir html
14 pwd
15 mkdir html
16 ls -l
17 echo '<h1>Hello, World!</h1>' > html/index.html
18 ls -l html
19 docker-compose up -d
20 sudo apt install docker-compose
21 docker-compose up -d
22 sudo apt-get install python3-distutils
23 sudo apt install python3-setuptools
24 sudo apt-get remove docker-compose
25 sudo apt-get install docker-compose
26 python3 --version
27 docker-compose up -d
28 sudo apt-get install python3-pip
29 sudo pip3 install docker-compose
30 docker-compose --version
31 sudo apt install python3-venv
32 docker-compose up -d
33 sudo apt-get remove docker-compose
34 sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -
s)-$(uname -m)" -o /usr/local/bin/docker-compose
35 sudo chmod +x /usr/local/bin/docker-compose
36 docker --version
37 docker-compose --version
38 sudo apt install docker-compose
39 docker-compose --version
40 docker --version
41 docker system prune -a
42 sudo systemctl restart docker
43 docker-compose up -d
44 sudo pip3 install --upgrade requests urllib3
45 sudo apt install
46 sudo apt install python3-venv
47 python3 -m venv venv
48 source venv/bin/activate
49 pip install requests urllib3
50 sudo apt install pipx
51 sudo apt install python3-requests python3-urllib3
52 pip list
53 dpkg -l | grep python3-requests
54 sudo apt-get remove --purge docker docker-engine docker.io containerd runc

```

```
55 sudo apt-get install docker.io
56 sudo apt install docker.io
57 sudo apt-get remove containerd
58 sudo apt-get update
59 sudo apt-get clean
60 sudo apt-get install -f
61 sudo apt-get remove $(dpkg --get-selections | grep hold | sed 's/\t.*//')
62 sudo apt-get install docker.io
63 sudo apt-get remove --purge containerd containerd.io
64 sudo apt-get clean
65 sudo apt-get update
66 sudo apt-get install -f
67 sudo apt-get install docker.io
68 sudo apt-get remove docker docker-engine docker.io containerd runc
69 sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
70 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
71 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -
cs) stable"
72 sudo apt-get update
73 sudo apt-get install docker-ce
74 sudo systemctl status docker
75 journalctl -xe -u docker.service
76 sudo systemctl restart docker
77 sudo systemctl status docker
78 sudo systemctl stop docker
79 docker --version
80 docker-compose --version
81 sudo pip3 install --upgrade docker-compose
82 sudo apt-get update
83 sudo apt-get install docker-compose-plugin
84 docker compose version
85 docker --version
86 docker-compose up -d
87 mkdir -p ~/myproject/dev ~/myproject/prod ~/myproject/lab
88 docker ps
89 mysql -h localhost -P 3306 -u root -p
90 sudo apt install mysql-client-core-8.0
91 sudo apt install mariadb-client-core
92 mysql -h localhost -P 3306 -u root -p
93 sudo systemctl status mysql
94 sudo systemctl start mysql
95 bind-address = 127.0.0.1
96 bind-address = 0.0.0.0
97 sudo systemctl restart mysql
98 sudo apt-get install mysql-server
99 sudo systemctl status mariadb
100 dpkg -l | grep mariadb
101 sudo apt-get update
102 ls /lib/systemd/system/mariadb.service
103 sudo apt-get remove --purge mariadb-server
104 sudo systemctl status mysql
105 sudo systemctl status mariadb
106 sudo systemctl start mysql
107 journalctl -xe -u mysql
108 docker ps > ~/myproject/logs/docker_ps_output.txt
109 mkdir -p ~/myproject/logs
110 docker ps > ~/myproject/logs/docker_ps_output.txt
```

```
111 cat ~/myproject/logs/docker_ps_output.txt
112 pwd
113 nano Dockerfile.web
114 nano Dockerfile.db
115 ls -l Dockerfile.*
116 cd ~/myproject/dev
117 docker build -t my-nginx-image -f Dockerfile .
118 ls -l ~/myproject/dev
119 cd ~/myproject/dev
120 nano Dockerfile
121 docker build -t my-nginx-image -f Dockerfile .
122 mkdir ~/myproject/dev/html
123 nano ~/myproject/dev/html/index.html
124 ls -l ~/myproject/dev
125 ls -l
126 cd ~/myproject/dev
127 docker build -t my-nginx-image -f Dockerfile .
128 cd ~/myproject/lab
129 docker build -t my-mysql-image -f Dockerfile .
130 nano Dockerfile
131 ls -l Dockerfile
132 docker build -t my-mysql-image -f Dockerfile .
133 docker ps
134 docker logs my-mysql-container
135 docker build -t my-mysql-image -f Dockerfile
136 cd.
137 cd ..
138 docker run -d --name dev-web -p 8080:80 my-nginx-image
139 docker run -d --name dev-db my-mysql-image
140 docker ps
141 docker ps > ~/myproject/logs/docker_ps_output_lab_dev.txt
142 docker logs --tail 100 dev-web
143 docker logs -f dev-db
144 cat Домашние_животные.txt Вьючные_животные.txt > Друзья_человека.txt
145 cat Друзья_человека.txt
146 mv Друзья_человека.txt 'Друзья человека.txt'
147 mkdir Питомник
148 mv 'Друзья человека.txt' Питомник/
149 git clone https://github.com/WilhelMark/Kennel_Animal_Accounting.git
150 cd Kennel_Animal_Accounting
151 git add .
152 ls
153 mv ../Питомник .
154 ls
155 git add Питомник
156 git commit -m "Domestic and pack animal files added."
157 git push origin main
158 WilhelMark
159 git push origin main
160 pwd
161 git status
162 git branch -a
163 ls
164 cd Питомник
165 ls
166 cd ..
167 git add .
```



```
168 ls
169 git commit -m "Domestic and pack animal files added."
170 git commit -m "Domestic and pack animal files added."
171 git config --global user.email "vwpiligrim@gmail.com"
172 git config --global user.name "WilhelMark"
173 git commit -m "Domestic and pack animal files added."
174 git push origin main
175 git remote -v
176 git remote set-url origin git@github.com:WilhelMark/Kennel_Animal_Accounting.git
177 git config --global user.name
178 git config --global user.email
179 git push origin main
180 ls ~/.ssh
181 ssh-keygen -t rsa -b 4096 -C "vwpiligrim@gmail.com"
182 ssh-add ~/.ssh/id_rsa
183 cat ~/.ssh/id_rsa.pub
184 git push origin main
185 wget https://dev.mysql.com/get/mysql-apt-config_0.8.26-1_all.deb
186 sudo dpkg -i mysql-apt-config_0.8.26-1_all.deb
187 sudo apt-get update
188 sudo apt-get upgrade
189 sudo apt-get purge mysql-server mysql-client mysql-common mysql-server-core-* mysql-client-core-*
190 sudo rm -rf /etc/mysql /var/lib/mysql
191 sudo apt-get autoremove
192 sudo apt-get autoclean
193 sudo dpkg -i /PATH/version-specific-package-name.deb
194 sudo apt install mysql-server
195 sudo apt update
196 sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys B7B3B788A8D3785C
197 sudo apt update
198 sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys B7B3B788A8D3785C
199 apt-key list
200 sudo add-apt-repository ppa:webupd8team/y-ppa-manager; sudo apt install y-ppa-manager
201 sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys B7B3B788A8D3785C
202 sudo apt update
203 apt-get install debian-keyring
204 gpg --keyserver pgp.mit.edu --recv-keys 1F41B907
205 sudo apt update
206 wget -qO - https://dev.mysql.com/doc/refman/8.0/en/checking-gpg-signature.html | sudo apt-key add -
207 sudo apt update
208 sudo add-apt-repository --remove ppa:webupd8team/y-ppa-manager
209 sudo apt update
210 sudo apt install mysql-server mysql-client
211 systemctl status mysql
212 wget http://archive.ubuntu.com/ubuntu/pool/main/c/curl/curl_7.68.0-1ubuntu2_amd64.deb
213 sudo dpkg -i curl_7.68.0-1ubuntu2_amd64.deb
214 curl --version
215 sudo dpkg -r curl
216 dpkg -l | grep curl
217 sudo dpkg -r curl
218 sudo dpkg -r libcurl
219 dpkg -l | grep curl
220 history
```

Установка формат времени для истории:

```
```bash
```

```
export HISTTIMEFORMAT='%F %T '
```

...

Эта команда изменяет формат вывода команды history, добавляя дату и время к каждой команде.

3. ****Сохранить историю в файл****:

- Если вы хотите сохранить вывод команды в текстовый файл, используйте следующую команду:

```
```bash
history > history_commands.txt
```
```

- Это создаст файл `history_commands.txt` в текущей директории с вашей историей команд.

Задание 6: Нарисовать диаграмму классов

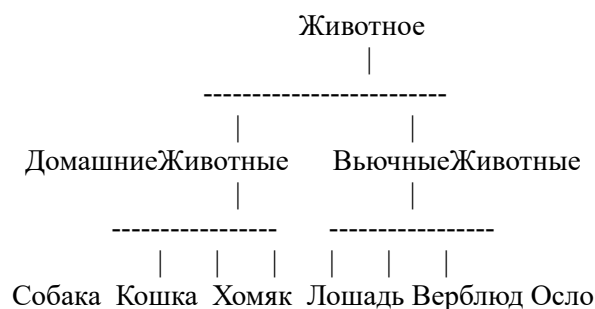
Для задания 6 нам нужно создать диаграмму классов, которая будет представлять иерархию классов домашних и выючных животных.

Структура классов

1. ****Родительский класс****: `Животное`
 - Атрибуты:
 - `имя`
 - `возраст`
 - `порода`
2. ****Класс-наследник****: `ДомашниеЖивотные` (наследует от `Животное`)
 - Классы-наследники:
 - `Собака` (наследует от `ДомашниеЖивотные`)
 - `Кошка` (наследует от `ДомашниеЖивотные`)
 - `Хомяк` (наследует от `ДомашниеЖивотные`)
3. ****Класс-наследник****: `ВыючныеЖивотные` (наследует от `Животное`)
 - Классы-наследники:
 - `Лошадь` (наследует от `ВыючныеЖивотные`)
 - `Верблюд` (наследует от `ВыючныеЖивотные`)
 - `Осло` (наследует от `ВыючныеЖивотные`)

Пример текстового представления диаграммы

...



...

Чтобы нарисовать диаграммы возможно использовать различные инструменты и программы. Вот несколько подходов:

1. Использование Microsoft Excel

2. Использование Google Sheets

3. Использование специализированных инструментов

- ****Lucidchart или Draw.io****.
- ****Microsoft Visio****.

4. Использование языков программирования

- ****Matplotlib (Python)****: Позволяет строить графики и диаграммы с помощью кода.
- ****D3.js (JavaScript)****: Библиотека для создания интерактивных графиков на веб-страницах.

Пример создания диаграммы с помощью Matplotlib (Python)

```
``python
import matplotlib.pyplot as plt
```

Для файла Python, который будет содержать диаграмму классов для вашего проекта, вы можете использовать следующее название:

Рекомендуемое название файла

```
**`animal_classes_diagram.py`**
```

Обоснование выбора названия

- ****`animal_classes`****: Это часть названия указывает на то, что файл связан с классами животных, что соответствует вашему проекту.
- ****`diagram`****: Указывает на то, что файл содержит код для создания диаграммы.
- ****`.py`****: Это стандартное расширение для файлов Python.

Пример кода для файла

```
``python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
```

```
# Создание фигуры и оси
fig, ax = plt.subplots(figsize=(10, 6))
```

Определение классов

```
classes = {
    "Животное": (0.5, 0.8),
    "ДомашниеЖивотные": (0.2, 0.6),
    "ВьючныеЖивотные": (0.8, 0.6),
    "Собака": (0.1, 0.4),
    "Кошка": (0.3, 0.4),
    "Хомяк": (0.5, 0.4),
    "Лошадь": (0.7, 0.4),
    "Верблюд": (0.9, 0.4),
    "Ослел": (1.1, 0.4)
}
```

Рисуем классы

```
for class_name, (x, y) in classes.items():
    ax.add_patch(mpatches.Rectangle((x - 0.1, y - 0.05), 0.2, 0.1, fill=True, edgecolor='black',
    facecolor='lightblue'))
    ax.text(x, y, class_name, fontsize=10, ha='center', va='center')
```

```

# Рисуем связи между классами
connections = [
    ("Животное", "ДомашниеЖивотные"),
    ("Животное", "ВьючныеЖивотные"),
    ("ДомашниеЖивотные", "Собака"),
    ("ДомашниеЖивотные", "Кошка"),
    ("ДомашниеЖивотные", "Хомяк"),
    ("ВьючныеЖивотные", "Лошадь"),
    ("ВьючныеЖивотные", "Верблюд"),
    ("ВьючныеЖивотные", "Осел")
]

for parent, child in connections:
    parent_pos = classes[parent]
    child_pos = classes[child]

    ax.plot([parent_pos[0], child_pos[0]], [parent_pos[1] - 0.05, child_pos[1] + 0.05], color='black', lw=1)

# Настройка осей
ax.set_xlim(0, 1.2)
ax.set_ylim(0, 1)
ax.axis('off') # Скрыть оси

# Заголовок
plt.title('Диаграмма классов животных', fontsize=14)

# Показать диаграмму
plt.show()

```

****Добавим файл в индекс Git**:**

```

`bash
git add animal_classes_diagram.py
`

```

****Создадим коммит**:**

```

`bash
git commit -m "Added animal class diagram file "
`

```

****Отправим изменения на удаленный репозиторий**:**

```

`bash
git push -u origin main
`

```

Задание 7: Создание базы данных “Друзья человека”

****Подключение к MySQL**:**

```

`bash
mysql -u root -p
`

```

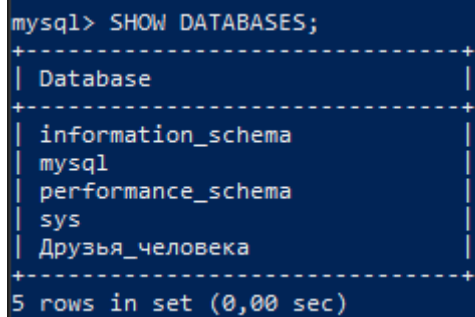
****Создадим базу данных**:**

```
```sql
CREATE DATABASE Друзья_человека;
```
```

Чтобы убедиться, что база данных создана, выполним команду:

```
```sql
SHOW DATABASES;
```
```

- Мы должны увидеть вашу новую базу данных в списке.



```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
| Друзья_человека  |
+-----+
5 rows in set (0,00 sec)
```

****Выберем созданную базу данных**:**

```
```sql
USE Друзья_человека;
```
```

Задание 8: Создание таблиц с иерархией из диаграммы

****Создадим таблицы для домашних и выючных животных**:**

```
```sql
CREATE TABLE Животные (
 id INT AUTO_INCREMENT PRIMARY KEY,
 имя VARCHAR(50),
 возраст INT,
 порода VARCHAR(50),
 тип ENUM('домашнее', 'вьючное')
);

CREATE TABLE Собака (
 id INT AUTO_INCREMENT PRIMARY KEY,
 животное_id INT,
 FOREIGN KEY (животное_id) REFERENCES Животные(id)
);

CREATE TABLE Кошка (
 id INT AUTO_INCREMENT PRIMARY KEY,
 животное_id INT,
 FOREIGN KEY (животное_id) REFERENCES Животные(id)
);

CREATE TABLE Хомяк (
 id INT AUTO_INCREMENT PRIMARY KEY,
```

```
 животное_id INT,
 FOREIGN KEY (животное_id) REFERENCES Животные(id)
);
```

```
CREATE TABLE Лошадь (
 id INT AUTO_INCREMENT PRIMARY KEY,
 животное_id INT,
 FOREIGN KEY (животное_id) REFERENCES Животные(id)
);
```

```
CREATE TABLE Верблюды (
 id INT AUTO_INCREMENT PRIMARY KEY,
 животное_id INT,
 FOREIGN KEY (животное_id) REFERENCES Животные(id)
);
```

```
CREATE TABLE Осел (
 id INT AUTO_INCREMENT PRIMARY KEY,
 животное_id INT,
 FOREIGN KEY (животное_id) REFERENCES Животные(id)
);
...
```

### Задание 9: Заполнение низкоуровневых таблиц

**\*\*Заполним таблицу "Животные"\*\*:**

```
```sql  
INSERT INTO Животные (имя, возраст, порода, тип) VALUES  
(('Барсик', 2, 'Кошка', 'домашнее'),  
(('Шарик', 3, 'Собака', 'домашнее'),  
(('Хомка', 1, 'Хомяк', 'домашнее'),  
(('Магнус', 4, 'Лошадь', 'вьючное'),  
(('Верблюд-1', 5, 'Верблюд', 'вьючное'),  
(('Ослик', 2, 'Осел', 'вьючное');  
...`
```

Задание 10: Удаление верблюдов и объединение таблиц лошадей и ослов

****Удалим верблюдов**:**

```
```sql  
DELETE FROM Животные WHERE порода = 'Верблюд';
...`
```

**\*\*Объединим таблицы "Лошадь" и "Осел" в одну таблицу "ВьючныеЖивотные"\*\*:**

```
```sql  
CREATE TABLE ВьючныеЖивотные AS  
SELECT * FROM Лошадь  
UNION ALL  
SELECT * FROM Осло;  
...`
```

Задание 11: Создание новой таблицы “молодые животные”

****Создадим новую таблицу для молодых животных**:**

```

```sql
CREATE TABLE МолодыеЖивотные AS
SELECT *,
 TIMESTAMPDIFF(MONTH, DATE_SUB(CURDATE(), INTERVAL возраст YEAR), CURDATE())
AS возраст_в_месяцах
FROM Животные
WHERE возраст > 1 AND возраст < 3;
```

```

Задание 12: Объединение всех таблиц в одну

****Объединим все таблицы в одну**:**

```

```sql
CREATE TABLE ВсеЖивотные AS
SELECT *, 'домашнее' AS старый_тип FROM Животные
UNION ALL
SELECT *, 'вьючное' AS старый_тип FROM ВьючныеЖивотные;
```

```

возникла ошибка "Unknown column 'имя' in 'field list'" указывает на то, что в таблице `ВьючныеЖивотные` нет столбца с названием `имя`. Это может произойти, если в таблицах, которые мы пытаемся объединить, разные структуры.

Шаги для устранения проблемы

1. ****Проверьте структуру таблиц**:**

Убедитесь, что в таблице `Животные` и в таблице `ВьючныеЖивотные` есть одинаковые столбцы. Вы можете проверить структуру таблиц с помощью следующей команды:

```

```sql
DESCRIBE Животные;
DESCRIBE ВьючныеЖивотные;
```

```

2. ****Исправим запрос**:**

Если в таблице `ВьючныеЖивотные` нет столбца `имя`, нам нужно изменить запрос, чтобы он соответствовал структуре обеих таблиц. Например, если в таблице `ВьючныеЖивотные` нет столбца `имя`, вы можете использовать только те столбцы, которые есть в обеих таблицах.

Пример исправленного запроса

Если в таблице `ВьючныеЖивотные` есть только `id`, `животное_id`, и другие поля, вы можете изменить запрос следующим образом:

```

```sql
CREATE TABLE ВсеЖивотные AS
SELECT id, имя, возраст, порода, тип, 'домашнее' AS старый_тип FROM Животные
UNION ALL
SELECT животное_id AS id, NULL AS имя, NULL AS возраст, NULL AS порода, NULL AS тип,
'вьючное' AS старый_тип FROM ВьючныеЖивотные;
```

```

Объяснение

- **NULL**: Используется для заполнения отсутствующих значений для столбцов `имя`, `возраст`, и `порода` из таблицы `ВыучныеЖивотные`.
- **AS**: Позволяет переименовать столбцы для соответствия.

Просмотр содержимого базы данных

****Выведем список всех баз данных****:

```
```sql
SHOW DATABASES;
```
```

- Эта команда покажет все базы данных, доступные на сервере.

****Выберем базу данных****, с которой хотим работать (например, `Друзья_человека`):

```
```sql
USE Друзья_человека;
```
```

- После выполнения этой команды мы увидим сообщение `Database changed`, что означает, что база данных выбрана.

****Выведем список таблиц**** в выбранной базе данных:

```
```sql
SHOW TABLES;
```
```

- Эта команда покажет все таблицы, которые находятся в текущей базе данных.

****Просмотрим структуру конкретной таблицы**** (например, `Животные`):

```
```sql
DESCRIBE Животные;
```
```

- Эта команда покажет информацию о каждом столбце в таблице: имя столбца, тип данных, возможность NULL и другие параметры.

****Посмотрим данные в таблице****:

```
```sql
SELECT * FROM Животные;
```
```

- Эта команда выведет все записи из таблицы `Животные`. Если таблица содержит много записей, вы можете ограничить вывод:

```
```sql
SELECT * FROM Животные LIMIT 10;
```
```

- Это покажет только первые 10 записей.


```
mysql> SHOW TABLES;
+-----+
| Tables_in_Друзья_человека |
+-----+
| Верблюд                     |
| ВсеЖивотные                 |
| ВьючныеЖивотные            |
| Животные                   |
| Кошка                       |
| Лошадь                     |
| МолодыеЖивотные            |
| Ослел                      |
| Собака                     |
| Хомяк                      |
+-----+
10 rows in set (0,00 sec)

mysql> DESCRIBE Животные;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| имя   | varchar(50) | YES |     | NULL    |               |
| возраст | int  | YES |     | NULL    |               |
| порода | varchar(50) | YES |     | NULL    |               |
| тип   | enum('домашнее', 'вьючное') | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0,00 sec)

mysql> SELECT * FROM Животные;
+----+-----+-----+-----+-----+
| id | имя   | возраст | порода | тип   |
+----+-----+-----+-----+-----+
| 1  | Барсик | 2       | Кошка | домашнее |
| 2  | Шарик  | 3       | Собака | домашнее |
| 3  | Хомка  | 1       | Хомяк | домашнее |
| 4  | Магнус | 4       | Лошадь | вьючное |
| 6  | Ослик  | 2       | Ослел | вьючное |
+----+-----+-----+-----+-----+
5 rows in set (0,00 sec)
```

Задание 13: Создание классов с инкапсуляцией и наследованием

****Создадим новую папку для проекта**:**
`PetRegistry`.

****Создайте новый файл**:**
`pet_registry.py`.

Теперь мы создадим классы с инкапсуляцией и наследованием в соответствии с диаграммой.

```
``python
# pet_registry.py

class Animal:
    def __init__(self, name, age):
        self.__name = name # Инкапсуляция: имя животного
        self.__age = age   # Инкапсуляция: возраст животного
        self.__commands = [] # Список команд, которые выполняет животное

    def get_name(self):
        return self.__name

    def get_age(self):
        return self.__age

    def add_command(self, command):
        self.__commands.append(command)
```

```
def get_commands(self):
    return self.__commands
```

```
class DomesticAnimal(Animal):
    def __init__(self, name, age):
        super().__init__(name, age) # Вызов конструктора родительского класса
```

```
class Horse(Animal):
    def __init__(self, name, age):
        super().__init__(name, age)
```

```
class Dog(DomesticAnimal):
    def __init__(self, name, age):
        super().__init__(name, age)
```

```
class Cat(DomesticAnimal):
    def __init__(self, name, age):
        super().__init__(name, age)
```

```
class Hamster(DomesticAnimal):
    def __init__(self, name, age):
        super().__init__(name, age)
...
```

Задание 14: Программа для работы с реестром домашних животных

добавим функционал для управления животными.

```
```python
pet_registry.py (продолжение)

def main_menu():
 animals = []
 while True:
 print("\nМеню:")
 print("1. Завести новое животное")
 print("2. Посмотреть список животных")
 print("3. Обучить животное новым командам")
 print("4. Выход")

 choice = input("Выберите опцию: ")

 if choice == '1':
 add_animal(animals)
 elif choice == '2':
 list_animals(animals)
 elif choice == '3':
 train_animal(animals)
 elif choice == '4':
 break
 else:
 print("Некорректный выбор. Попробуйте снова.")
```

```

def add_animal(animals):
 name = input("Введите имя животного: ")
 age = int(input("Введите возраст животного: "))

 animal_type = input("Введите тип животного (собака/кошка/хомяк/лошадь): ").lower()

 if animal_type == 'собака':
 animals.append(Dog(name, age))
 elif animal_type == 'кошка':
 animals.append(Cat(name, age))
 elif animal_type == 'хомяк':
 animals.append(Hamster(name, age))
 elif animal_type == 'лошадь':
 animals.append(Horse(name, age))
 else:
 print("Некорректный тип животного.")

def list_animals(animals):
 if not animals:
 print("Список животных пуст.")
 else:
 for animal in animals:
 print(f"Имя: {animal.get_name()}, Возраст: {animal.get_age()}")

def train_animal(animals):
 name = input("Введите имя животного для обучения: ")

 for animal in animals:
 if animal.get_name() == name:
 command = input("Введите команду для обучения: ")
 animal.add_command(command)
 print(f"{name} обучен команде '{command}'.")
 return

 print(f"Животное с именем {name} не найдено.")

```

### Задание 15: Создание класса Счетчик

Теперь добавим класс `Counter`, который будет увеличивать значение при создании нового животного.

```

```python
# pet_registry.py (продолжение)

```

```

class Counter:
    def __init__(self):
        self.count = 0

    def add(self):
        self.count += 1

    def get_count(self):
        return self.count

```

```

# Использование счетчика в основном меню
def main_menu():

```

```

animals = []
counter = Counter() # Создаем объект счетчика
while True:
    print("\nМеню:")
    print("1. Завести новое животное")
    print("2. Посмотреть список животных")
    print("3. Обучить животное новым командам")
    print("4. Выход")

    choice = input("Выберите опцию: ")

    if choice == '1':
        add_animal(animals, counter) # Передаем счетчик
    elif choice == '2':
        list_animals(animals)
    elif choice == '3':
        train_animal(animals)
    elif choice == '4':
        break
    else:
        print("Некорректный выбор. Попробуйте снова.")

def add_animal(animals, counter): # Принимаем счетчик как аргумент
    name = input("Введите имя животного: ")
    age = int(input("Введите возраст животного: "))

    animal_type = input("Введите тип животного (собака/кошка/хомяк/лошадь): ").lower()

    if animal_type == 'собака':
        animals.append(Dog(name, age))
    elif animal_type == 'кошка':
        animals.append(Cat(name, age))
    elif animal_type == 'хомяк':
        animals.append(Hamster(name, age))
    elif animal_type == 'лошадь':
        animals.append(Horse(name, age))

    counter.add() # Увеличиваем счетчик
...

```