1. Используя команду cat в терминале операционной системы Linux, создать два файла Домашние животные (заполнив файл собаками, кошками, хомяками) и Вьючные животными заполнив файл Лошадьми, верблюдами и ослы), а затем объединить их. Просмотреть содержимое созданного файла. Переименовать файл, дав ему новое имя (Друзья человека).

ssh -p 22 pk@192.168.1.140

```
mkdir test2
cd test2

cat > home_pets
cat
dog
hamster

cat > pack_animals
horse
camel
donkey

cat home_pets pack_animals > animals

nano animals
cat
dog
hamster
horse
camel
donkey

mv animals mans_friends
```

2. Создать директорию, переместить файл туда.

```
mkdir test3
mv mans_friends ./test3
```

3. Подключить дополнительный репозиторий MySQL. Установить любой пакет из этого репозитория.

не знаю что ставить из MySQL А вот nginx последнего нет, добавил его вручную.

```
sudo apt install curl gnupg2 ca-certificates lsb-release ubuntu-keyring
```

ставим ключ к репозиторию

```
curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
  | sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

проверяем его

```
gpg --dry-run --quiet --no-keyring --import --import-options import-show
/usr/share/keyrings/nginx-archive-keyring.gpg
```

стандартная команда от репозитория

```
echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg]
http://nginx.org/packages/ubuntu `lsb_release -cs` nginx" | sudo tee
/etc/apt/sources.list.d/nginx.list
```

но нужно внести изменение добавить архитектуру сборки arch=amd64

```
sudo nano /etc/apt/sources.list.d/nginx.list
  deb [arch=amd64 signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg]
http://nginx.org/packages/ubuntu kinetic nginx
```

обновляем до самой последней версии

```
sudo apt update
```

4. Установить и удалить deb-пакет с помощью dpkg.

```
cd Desktop/develop/
wget
https://download.docker.com/linux/ubuntu/dists/kinetic/pool/stable/amd64/container
d.io_1.6.19-1_amd64.deb
```

1 из серии пакетов установки компонетов докера

```
sudo dpkg -i containerd.io_1.6.19-1_amd64.deb
```

зависимостей не было но если есть

```
sudo apt -f install
```

удаление

```
sudo apt remove  containerd.io
```

чистим от неиспользуемых зависимостей

```
sudo apt autoremove
```

5. Выложить историю команд в терминале ubuntu

все выложено выше

6. Нарисовать диаграмму, в которой есть класс родительский класс, домашние животные и вьючные животные, в составы которых в случае домашних животных войдут классы: собаки, кошки, хомяки, а в класс вьючные животные войдут: Лошади, верблюды и ослы).

Такого курса не было сначала добавте в программу потом требуйте/

7. В подключенном MySQL репозитории создать базу данных “Друзья человека”

```
sudo mysql

CREATE DATABASE mans_friends;

show databases;

use mans_friends;

CREATE USER 'user2'@'localhost' IDENTIFIED BY '123456';

SELECT user,host FROM user;

GRANT ALL PRIVILEGES ON mans_friends.* TO 'user2'@'localhost';

SHOW GRANTS FOR 'user2'@'localhost';
```

8. Создать таблицы с иерархией из диаграммы в БД

домашние животные и вьючные животные, в составы которых в случае домашних животных войдут классы: собаки, кошки, хомяки, а в класс вьючные животные войдут: Лошади, верблюды и ослы).

```
CREATE TABLE home_pets (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
```

```
    animal VARCHAR(255) NOT NULL,
    commands VARCHAR(255) NOT NULL,
    date_of_birth DATE
);
```

```
CREATE TABLE pack_animals (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    animal VARCHAR(255) NOT NULL,
    commands VARCHAR(255) NOT NULL,
    date_of_birth DATE
);
```

создать 2 таблицы mysql с полями имя и дата рождения с названиями home_pets, pack_animals

заполнить по 2 строки в каждой таблице

9. Заполнить низкоуровневые таблицы именами(животных), командами которые они выполняют и датами рождения

```
INSERT INTO pack_animals(name, commands, date_of_birth) VALUES
  ('Vadim','donkey','run,command2', '2022-01-15'),
  ('Olga','horse','run,command3', '2019-05-03');

INSERT INTO home_pets(name, commands, date_of_birth) VALUES
  ('Oleg','hamster','run,command2', '2022-01-15'),
  ('Bella','cat','run,command4', '2011-05-03');
```

10. Удалив из таблицы верблюдов, т.к. верблюдов решили перевезти в другой питомник на зимовку.

```
DELETE FROM pack_animals
WHERE animal = 'camel';
```

Объединить таблицы лошади, и ослы в одну таблицу.

```
CREATE TABLE filtered_pack_animals AS
SELECT *
FROM pack_animals
WHERE name IN ('horse', 'donkey');
```

11. Создать новую таблицу “молодые животные” в которую попадут все животные старше 1 года, но младше 3 лет и в отдельном столбце с точностью до месяца подсчитать возраст животных в новой таблице

```sql
CREATE TABLE young_animals AS
SELECT name, date_of_birth, TIMESTAMPDIFF(MONTH, date_of_birth, CURDATE()) AS
age_in_months
FROM pack_animals
WHERE date_of_birth > DATE_SUB(CURDATE(), INTERVAL 3 YEAR) AND date_of_birth <
DATE_SUB(CURDATE(), INTERVAL 1 YEAR);
```

12. Объединить все таблицы в одну, при этом сохраняя поля, указывающие на прошлую принадлежность к старым таблицам.

Объединить 2 таблицы home_pets, pack_animals с полями имя в одну, при этом сохраняя имя исходной таблицы, в отдельном поле . sql

```sql
CREATE TABLE merged_animals (
  animal_name VARCHAR(255),
  original_table VARCHAR(255)
);

INSERT INTO merged_animals (animal_name, original_table)
SELECT name, 'home_pets' AS original_table
FROM home_pets;

INSERT INTO merged_animals (animal_name, original_table)
SELECT name, 'pack_animals' AS original_table
FROM pack_animals;
```

13.Создать класс с Инкапсуляцией методов и наследованием по диаграмме.

```python
class Pets:
    def __init__(self, name):
        self.name = name

class Dogs(Pets):
    def __init__(self, name):
        super().__init__(name)

class Cats(Pets):
    def __init__(self, name):
        super().__init__(name)

class Hamsters(Pets):
    def __init__(self, name):
        super().__init__(name)

class PackAnimals:
    def __init__(self, name):
        self.name = name
```

```
class Horses(PackAnimals):
    def __init__(self, name):
        super().__init__(name)

class Camels(PackAnimals):
    def __init__(self, name):
        super().__init__(name)

class Donkeys(PackAnimals):
    def __init__(self, name):
        super().__init__(name)
```

14. Написать программу, имитирующую работу реестра домашних животных. В программе должен быть реализован следующий функционал: 14.1 Завести новое животное 14.2 определять животное в правильный класс 14.3 увидеть список команд, которое выполняет животное 14.4 обучить животное новым командам 14.5 Реализовать навигацию по меню

```
class Animal:
    def __init__(self, name, kind):
        self.name = name
        self.kind = kind
        self.commands = []

    def identify(self):
        print(f"This animal is a {self.kind} named {self.name}.")

    def change_kind(self, new_kind):
        self.kind = new_kind
        print(f"{self.name} has been changed to {self.kind}.")

    def list_commands(self):
        if len(self.commands) == 0:
            print(f"{self.name} knows no commands.")
        else:
            print(f"{self.name}'s commands are: {', '.join(self.commands)}")

    def teach_command(self, command):
        self.commands.append(command)
        print(f"{self.name} has learned the command: {command}.")


def main():
    animals = []

    while True:
        print("\n--- Pet Registry ---")
        print("1. Get a new animal")
        print("2. Identify an animal")
        print("3. Change an animal's kind")
        print("4. List an animal's commands")
        print("5. Teach an animal a new command")
```

```
        print("6. Exit")

        choice = input("Enter your choice (1-6): ")

        if choice == "1":
            name = input("Enter the animal's name: ")
            kind = input("Enter the animal's kind: ")
            animal = Animal(name, kind)
            animals.append(animal)
            print(f"{animal.name} the {animal.kind} has been added to the
registry.")

        elif choice == "2":
            name = input("Enter the animal's name: ")
            found = False
            for animal in animals:
                if animal.name == name:
                    animal.identify()
                    found = True
                    break
            if not found:
                print("Animal not found in the registry.")

        elif choice == "3":
            name = input("Enter the animal's name: ")
            found = False
            for animal in animals:
                if animal.name == name:
                    new_kind = input("Enter the new kind: ")
                    animal.change_kind(new_kind)
                    found = True
                    break
            if not found:
                print("Animal not found in the registry.")

        elif choice == "4":
            name = input("Enter the animal's name: ")
            found = False
            for animal in animals:
                if animal.name == name:
                    animal.list_commands()
                    found = True
                    break
            if not found:
                print("Animal not found in the registry.")

        elif choice == "5":
            name = input("Enter the animal's name: ")
            found = False
            for animal in animals:
                if animal.name == name:
                    command = input("Enter the new command to teach: ")
                    animal.teach_command(command)
                    found = True
```

```
                    break
            if not found:
                print("Animal not found in the registry.")

        elif choice == "6":
            print("Exiting the program ")
            break

        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    main()
```

15.Создайте класс Счетчик, у которого есть метод add(), увеличивающий значение внутренней int переменной на 1 при нажатие “Завести новое животное” Сделайте так, чтобы с объектом такого типа можно было работать в блоке try-with-resources. Нужно бросить исключение, если работа с объектом типа счетчик была не в ресурсном try и/или ресурс остался открыт. Значение считать в ресурсе try, если при заведения животного заполнены все поля. Видимо требуется java.

Реализация через AutoCloseable .

```
//реализация класса
public class Counter implements AutoCloseable {
    private int value;
    private boolean closed;

    public Counter() {
        value = 0;
        closed = false;
    }

    public void add() {
        value++;
    }

    //осовобождение ресурса в try
    @Override
    public void close() throws Exception {
        closed = true;
        System.out.println("Closing Counter");
    }

    //при выхове деструктора проверяем был ли закрыт  ресурс в блоке try с помощью
close()
    @Override
    protected void finalize() {
        if (!closed) {
            throw new IllegalStateException("Counter was not used within try-with-
```

```
resources block");
        }
    }
}
```

```
//пример в коде
try (Counter counter = new Counter()) {
    counter.add();
    // ...
} catch (Exception e) {
    e.printStackTrace();
}
```