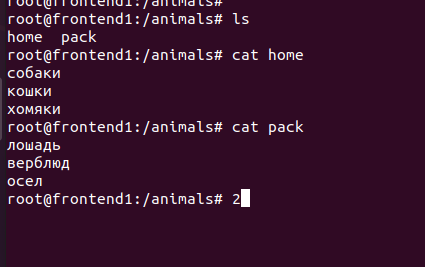
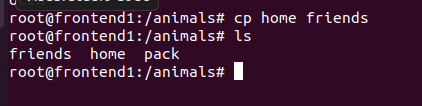
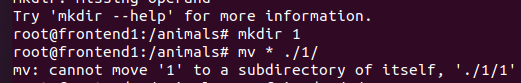
# Задание

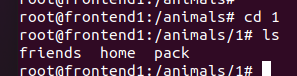
1. Используя команду cat в терминале операционной системы Linux, создать два файла Домашние животные (заполнив файл собаками, кошками, хомяками) и Вьючные животными заполнив файл Лошадьми, верблюдами и ослы), а затем объединить их. Просмотреть содержимое созданного файла. Переименовать файл, дав ему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





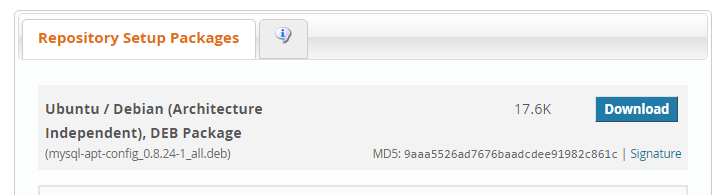
2. Создать директорию, переместить файл туда.





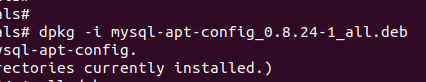
3. Подключить дополнительный репозиторий MySQL. Установить любой пакет из этого репозитория.

Качаем

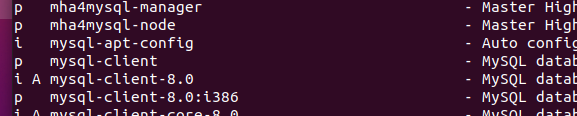


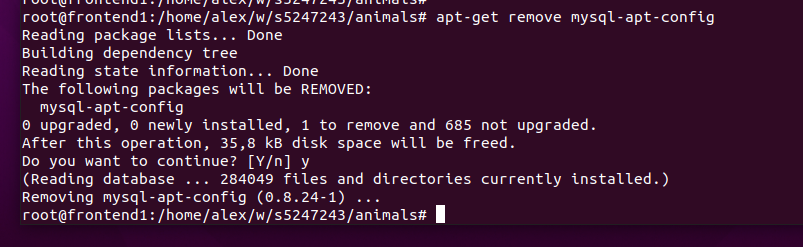


4. Установить и удалить deb-пакет с помощью dpkg.



Удалим пакет:

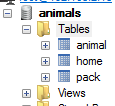




5. Выложить историю команд в терминале ubuntu

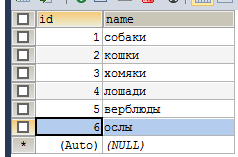
6. Нарисовать диаграмму, в которой есть класс родительский класс, домашние животные и вьючные животные, в составы которых в случае домашних животных войдут классы: собаки, кошки, хомяки, а в класс вьючные животные войдут: Лошади, верблюды и ослы).

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

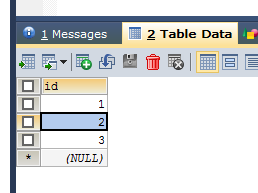


8. Создать таблицы с иерархией из диаграммы в БД

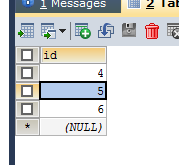
Основная таблица



Домашние:

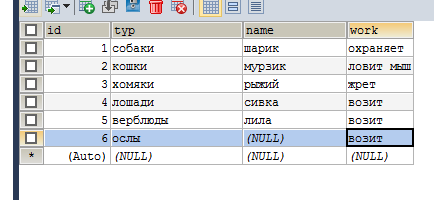


Вьючные

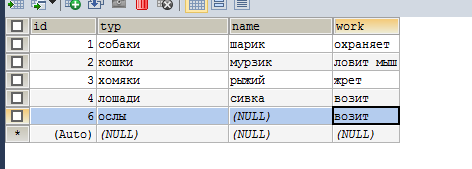


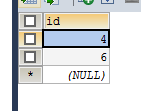
9. Заполнить низкоуровневые таблицы именами(животных), командами которые они выполняют и датами рождения

Изменим таблицу с животными

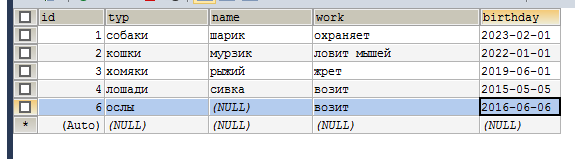


10. Удалив из таблицы верблюдов, т.к. верблюдов решили перевезти в другой питомник на зимовку. Объединить таблицы лошади, и ослы в одну таблицу.

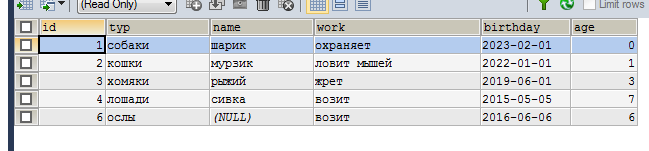




11. Создать новую таблицу “молодые животные” в которую попадут все животные старше 1 года, но младше 3 лет и в отдельном столбце с точностью до месяца подсчитать возраст животных в новой таблице



Новая таблица



Запрос на создание

select \*,

(YEAR(CURRENT\_DATE)-YEAR(a.birthday))-(RIGHT(CURRENT\_DATE,5)<RIGHT(a.birthday,5)) as age

from animal a;

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

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

package com.company;  
  
import java.sql.\*;  
  
public class Animal {  
 public int id;  
 public String typ;  
 public String name;  
 public String work;  
 public Date birthday;  
  
 // Чтение из базы данных  
 public void read( ResultSet r ) throws SQLException {  
 id = Integer.*parseInt*(r.getString("id"));  
 typ = r.getString("typ");  
 name = r.getString("name");  
 work = r.getString("work");  
 birthday = r.getDate("birthday");  
 }  
}

package com.company;  
  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
public class Role {  
 public int id;  
  
 // Чтение из базы данных  
 public void read( ResultSet r ) throws SQLException {  
 id = Integer.parseInt(r.getString("id"));  
 }  
}

14. Написать программу, имитирующую работу реестра домашних животных. В программе должен быть реализован следующий функционал:

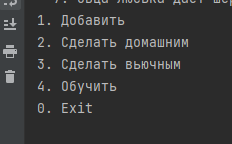
14.1 Завести новое животное

14.2 определять животное в правильный класс

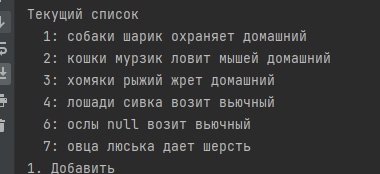
14.3 увидеть список команд, которое выполняет животное

14.4 обучить животное новым командам

14.5 Реализовать навигацию по меню



При работе постоянно показывается текущий список:



Животное можно обучить, можно присвоить роль домашнего или вьючного.

15. Создайте класс Счетчик, у которого есть метод add(), увеличивающий̆ значение внутренней̆ int переменной̆ на 1 при нажатии “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 заведении животного заполнены все поля.

package com.company;  
  
// Самозакрываемый класс счетчика  
public class Cnt implements AutoCloseable {  
 public int cnt = 0;  
  
 public void add() throws Exception  
 {  
 cnt++;  
 }  
  
 @Override  
 public void close() throws Exception {  
 cnt--;  
 }  
}

И работа с классом:

// Создаем без try  
// Cnt cnt = new Cnt();  
  
try(Cnt cnt = new Cnt())  
{  
 // Вот тут ошибка, если вызвать без try  
 cnt.add();  
  
 a = new Animal();

...

int res = q.executeUpdate();  
 out.println(" Добавлено " + res + " строк");  
}  
catch (Exception ex)  
{  
 ;  
} // \*/

# Исходный код

package com.company;  
  
import java.io.PrintStream;  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
import static java.nio.charset.StandardCharsets.*UTF\_8*;  
  
public class Main {  
  
 public static void main(String[] args) {  
 try  
 {  
 new Main().run();  
 } catch (SQLException e)  
 {  
 e.printStackTrace();  
 }  
 }  
  
 Connection conn;  
 PrintStream out = new PrintStream(System.*out*, true, *UTF\_8*);  
 Scanner in = new Scanner( System.*in* );  
  
 void run() throws SQLException {  
 conn = DriverManager.*getConnection*("jdbc:mysql://192.168.2.18/animals"  
 // "?useUnicode=false&characterEncoding=u",  
 , "root", "12345678");  
  
 Animal a;  
 PreparedStatement q;  
 ResultSet r;  
  
 List<Integer> home, pack;  
  
 home = new ArrayList<>();  
 pack = new ArrayList<>();  
  
 q = conn.prepareStatement("select \* from home;");  
 r = q.executeQuery();  
  
 while (r.next())  
 home.add( r.getInt("id") );  
  
 q = conn.prepareStatement("select \* from pack;");  
 r = q.executeQuery();  
  
 while (r.next())  
 pack.add( r.getInt("id") );  
  
 while (true)  
 {  
  
 q = conn.prepareStatement( "select \* from animal;" );  
 r = q.executeQuery();  
  
 out.println("Текущий список");  
 while (r.next())  
 {  
 a = new Animal();  
 a.read(r);  
  
 out.print( " " + a.id + ": " + a.typ + " " + a.name + " " + a.work );  
  
 if (home.indexOf(a.id) >= 0)  
 out.print( " домашний");  
 if (pack.indexOf(a.id) >= 0)  
 out.print( " вьючный");  
  
 out.println("");  
 }  
  
 out.println( "1. Добавить");  
 out.println( "2. Сделать домашним");  
 out.println( "3. Сделать вьючным");  
 out.println( "4. Обучить");  
 out.println( "0. Exit" );  
  
 int n = Integer.*parseInt*(in.nextLine());  
 if (n == 0) break;  
 if (n == 1) {  
 // Добавить  
  
 // Создаем без try  
 // Cnt cnt = new Cnt();  
  
 try(Cnt cnt = new Cnt())  
 {  
 // Вот тут ошибка, если вызвать без try  
 cnt.add();  
  
 a = new Animal();  
  
 out.print("Введите тип: ");  
 a.typ = in.nextLine();  
 out.print("Введите имя: ");  
 a.name = in.nextLine();  
  
 q = conn.prepareStatement("insert into animal (typ, name) values (?, ?);");  
 q.setString(1, a.typ);  
 q.setString(2, a.name);  
  
 int res = q.executeUpdate();  
 out.println(" Добавлено " + res + " строк");  
 }  
 catch (Exception ex)  
 {  
 ;  
 } // \*/  
 }  
 if ((n == 2) | (n == 3)) {  
 ; // Сделать домашним или cделать вьючным  
  
 out.print( "Введите ид: ");  
 int id = Integer.*parseInt*(in.nextLine());  
  
 if ((n == 2) & (home.indexOf(id) >= 0)) {  
 out.println(" Такой ИД уже есть в таблице");  
 continue;  
 }  
  
 if ((n == 3) & (pack.indexOf(id) >= 0)) {  
 out.println(" Такой ИД уже есть в таблице");  
 continue;  
 }  
  
 if (n == 2) {  
 q = conn.prepareStatement("insert into home (id) values (?);");  
 home.add(id);  
 }  
 else {  
 q = conn.prepareStatement("insert into pack (id) values (?);");  
 pack.add(id);  
 }  
  
 q.setInt(1, id);  
  
 int res = q.executeUpdate();  
 out.println( " Добавлено " + res + " строк");  
 }  
 if (n == 4) {  
 // Обучить  
 out.print( "Введите ид: ");  
 int id = Integer.*parseInt*(in.nextLine());  
  
 out.print( "Введите новое умение: ");  
 String u = in.nextLine();  
  
 q = conn.prepareStatement( "update animal " +  
 "set work=CONCAT( IFNULL( work, '' ), ? ) " +  
 "where id=?;" );  
 q.setString(1, u);  
 q.setInt(2, id);  
  
 int res = q.executeUpdate();  
 out.println( " обновлено " + res + " строк");  
 }  
 }  
 }  
}