```
//
//  Animals.hpp
//  ZooOblig3
//
//  Created by Øyvind Nordbø and August Henninen on 04/03/2024.
//

#ifndef Animals_hpp
#define Animals_hpp

#include <stdio.h>
#include <iostream>
#include <string>

//————————————————————————————————ANIMAL
 Class————————————————————————————————————

class Animal
{

protected:
    //default constructor
    Animal();//kunne evt gitt konstruktøren under default argumenter, men
     ville beholde printen slik at en er sikker på at objektet har fått
     default verdiene eller ikke utenom å bruke en if sjekk i funksjoen

    //Constructor
    Animal(std::string name, std::string species,
         float weight, std::string movement);

    static inline int animal_count{ 0 }; //litt dum løsning, men fikk
     problemer da jeg ikke kunne lage objekter av dyre klassen. Kunne løst
     bedre, men nå har jeg tilgang til variabelen animal_count utenfor base
     klassen min

public:
    friend class Zoo;
    friend std::ostream& operator<<(std::ostream& output, const Animal&
     animal);

    //Destructor
    virtual ~Animal();

    void print_animal() const;
    void delete_animal();

    //Accessor functions (Getters)
    std::string get_name() const;
    std::string get_species() const;
    float get_weight() const;
    std::string get_movement() const;


    //Set functions (Setters)
    void set_name(std::string name);
```

```
    void set_species(std::string species);
    void set_weight(float weight);
    void set_movement(std::string movement);



private:
    //Member variables
    std::string its_name;
    std::string its_species;
    float its_weight;
    std::string its_movement;

};


//----------------------------------------Mammal
 Class-------------------------------------------------
class Mammal : public virtual Animal //hvem vet, kanskje programmet
 utvides. så lan på virtual her
{
public:
    explicit Mammal();
    Mammal(std::string name, std::string species, float weight, std::string
     movement);
    virtual ~Mammal();

private:
    static inline int mammal_count{ 0 };
};

//----------------------------------------Bird
 Class-------------------------------------------------
class Bird : public virtual Animal
{
public:
   explicit Bird();
    Bird(std::string name, std::string species, float weight, std::string
     movement);
    ~Bird();
private:
    static inline int bird_count{ 0 };
};

//----------------------------------------Fish
 Class-------------------------------------------------
class Fish : public virtual Animal
{
public:
    explicit Fish();
    Fish(std::string name, std::string species, float weight, std::string
     movement);
    ~Fish();
private:
    static inline int fish_count{ 0 };
```

```
};

#endif /* Animals_hpp */
```