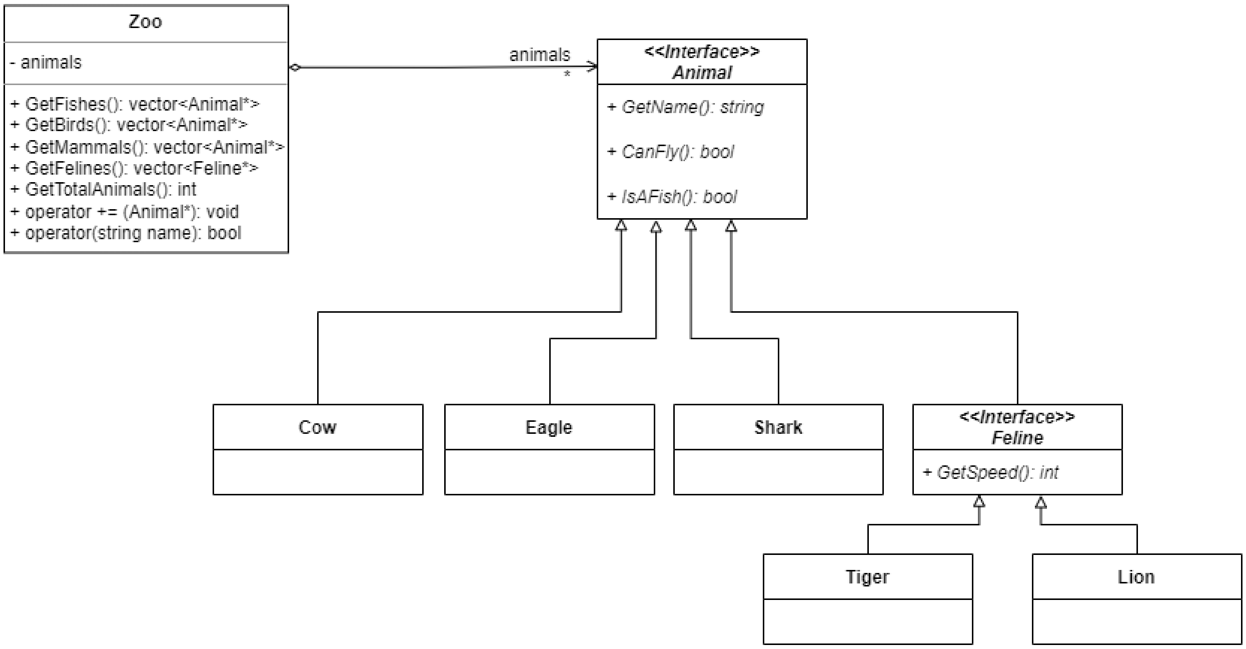
Lab exam (part 2)

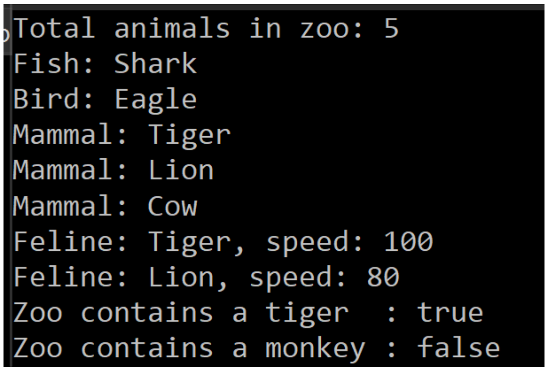
Consider the following UML diagram:


Zoo
- animals
+ GetFishes(): vector<Animal*>
+ GetBirds(): vector<Animal*>
+ GetMammals(): vector<Animal*>
+ GetFelines(): vector<Feline*>
+ GetTotalAnimals(): int
+ operator += (Animal*): void
+ operator(string name): bool
animals
*
<<Interface>>
Animal
+ GetName(): string
+ CanFly(): bool
+ IsAFish(): bool
Cow
Eagle
Shark
<<Interface>>
Feline
+ GetSpeed(): int
Tiger
Lion

Build all the files with the classes described in the above diagram so that the following code:

```
#include <iostream>
#include "Zoo.h"
#include "Shark.h"
#include "Eagle.h"
#include "Lion.h"
#include "Tiger.h"
#include "Cow.h"
int main()
{
    Zoo z;
    z += new Shark();
    z += new Eagle();
    z += new Tiger();
    z += new Lion();
    z += new Cow();
    std::cout << "Total animals in zoo: " << z.GetTotalAnimals() << std::endl;
    for (auto a : z.GetFishes())
        std::cout << "Fish: " << a->GetName() << std::endl;
    for (auto a : z.GetBirds())
        std::cout << "Bird: " << a->GetName() << std::endl;
    for (auto a : z.GetMammals())
        std::cout << "Mammal: " << a->GetName() << std::endl;
    for (auto a : z.GetFelines())
        std::cout << "Feline: " << a->GetName() << ", speed: " << a->GetSpeed() << std::endl;
    std::cout << "Zoo contains a tiger  : " << std::boolalpha << z("Tiger") << std::endl;
    std::cout << "Zoo contains a monkey : " << std::boolalpha << z("Monkey") << std::endl;
    return 0;
}
```

will out upon execution:



```
Total animals in zoo: 5
Fish: Shark
Bird: Eagle
Mammal: Tiger
Mammal: Lion
Mammal: Cow
Feline: Tiger, speed: 100
Feline: Lion, speed: 80
Zoo contains a tiger  : true
Zoo contains a monkey : false
```

Observations:

1. Animal class and all of its derived classes are inlined (there is no .cpp file, only a header file)
2. Except for Zoo class, none of the rest of the classes have any data members ! Adding other data members and/or methods will impact $G_2$2, $G_2$6 and $G_2$8
3. It is recommended to use STL containers to solve this problem.

Grading:

$G_2$1	Organize your project in 10 files: **main.cpp**, **Animal.h**, **Feline.h**, **Lion.h**, **Tiger.h**, **Shark.h**, **Eagle.h**, **Cow.h**, **Zoo.h**, and **Zoo.cpp**	1p **WP**
$G_2$2	Organize the file **Zoo.h** to correspond to the UML diagram (one data member, two operators, 5 getter methods).	2p **WP**
$G_2$3	Implementation of Zoo::GetTotalAnimals()	1p
$G_2$4	**Zoo::GetFishes**(), **Zoo::GetBirds**() and **Zoo::GetMammals**() (3p each correctly implemented method)	9p **AP**
$G_2$6	Correctly implement all virtual methods in classes Tiger, Lion, Eagle, Cow and Shark. (1p for each)	5p
$G_2$6	Implementation of operator **+=**	1p
$G_2$7	Implementation of operator function call (to test if an animal exists in the zoo)	3p **AP**
$G_2$8	Implementation of Zoo::GetFelines() (requires using dynamic cast for runtime checks)	5p **AP**
$G_2$9	Program compiles and upon execution produces the expected results	3p

Criteria explanation:

AP ⇒ the capability to correctly apply OOP principles (inheritance, polymorphism, etc)
WP⇒ the capability to write C++ programs based on specifications
The evaluation will consider all the relevant points for a criteria.