Cpp development

Lecture 4

Schedule

- Friend functions and classes
- Abstractions
- Mutual inheritance
- Operator predefinition
- Copy constructor

What can we do when we need to access a private property from another class ?

Friend function

- A special functions that can access private members of a class
- It is very useful when we need to implement a insider in a class.

Implementation

```
class Car

{

    int engineCondition;
    friend void repairEngine(Car &aCar);

}
```

Friend Class

- A class that can access private/protected member of another class

Implementation

Class Man

```
 {

     private:
     int amountOfMoneyInBankAccount;
     friend class theWomen

}
```

Task

- Write a class Mechanic with members priceForRepairingACar(Car &aCar), repairACar(Car &aCar).
- Write a class Car with members private bool needsARepair, public float priceOfCar.
- Write a class Man, with only function to crashACar(Car &aCar).
- The price for repairmen is 5% of the price of the car.
- Write a program where a Mechanic makes diagnostics and repairs a car.

Abstractions

Do you have problems
with your homework ?

Abstraction

- The abstraction is to have a objects/functions with same name and similar functions.

Abstraction example

- A fluffy bear toy and a real grizzly bear:
- They are bears but the one is being used for hugging the other will eat you with the clothes.

Abstractions in Cpp

- Two ways:
- Using virtual functions
- Using abstract classes

Virtualization

- Virtualization is used in Cpp for telling the compiler to make a function dynamic. That means that the function will be linked to the class at runtime, not when compiling.

How to virtualize

```
class Shape

{

    public:
    virtual float getPerimeter() { return 0;};
    virtual float getSurface() { return 0; };

}
```

How to virtualize

```
class Circle: public Shape

{

    float getPerimeter() { return 15;};
    float getSurface() {return 20;};

}
```

Do you see a problem ?

Abstract functions

- When we need just to mention that there is this function but not to implement it into the base class

How to virtualize

```
class Shape

{

    virtual float getPerimeter() = 0;
    virtual float getSurface() = 0;

}
```

How to virtualize

```
class Circle: public Shape

{

    float getPerimeter() { return 15;};
    float getSurface() {return 20;};

}
```

But what if we need to make a lot of abstract functions in a class

Abstract classes

- A class that defines names of functions, but not their implementation

How to abstract classes

```
abstract class Shape

{

    float getPerimeter();
    float getSurface();

}
```

How to abstract classes

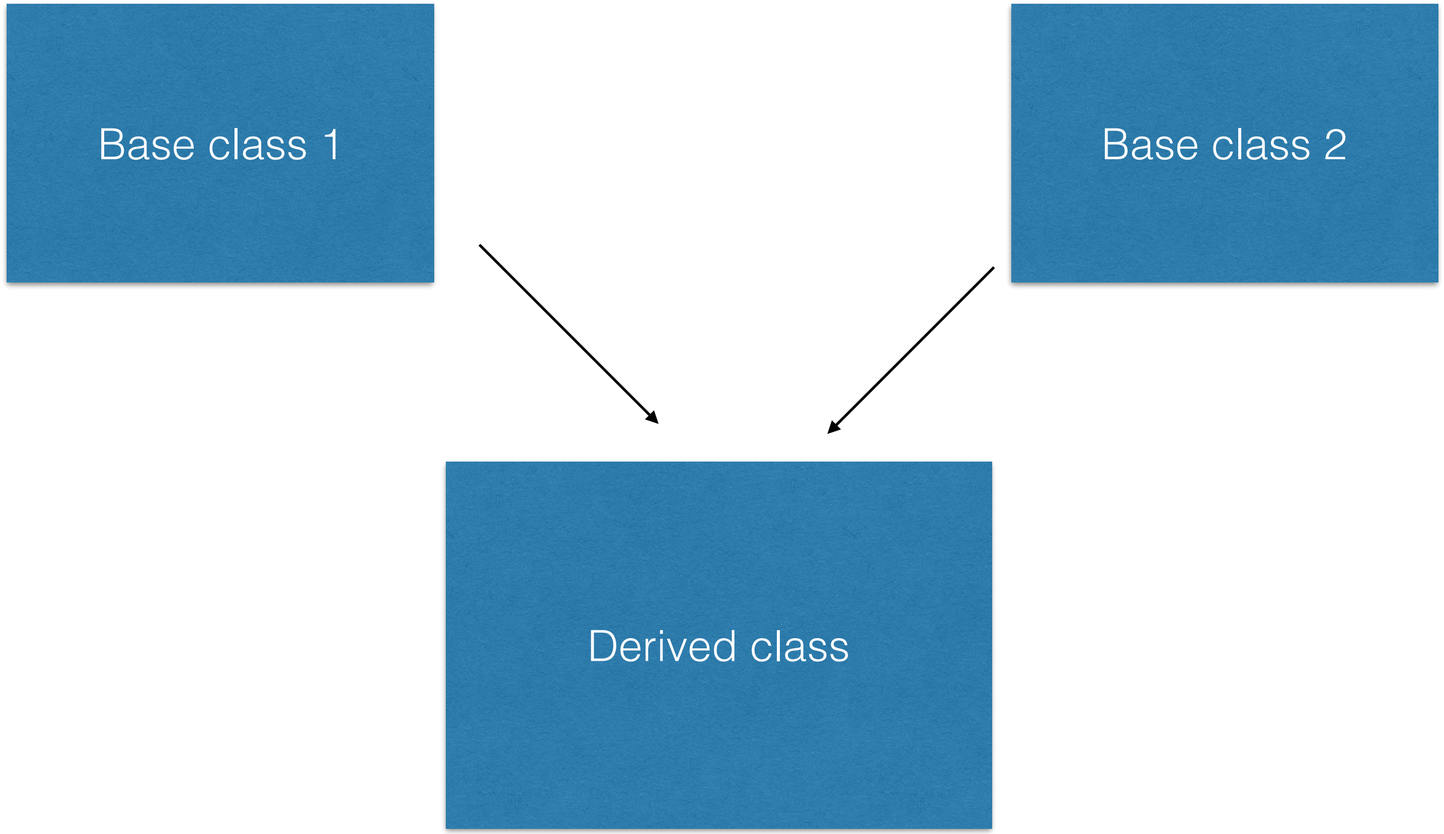
```
class Circle

{

    float getPerimeter() { return 15;};
    float getSurface() {return 20;};

}
```

Protocols

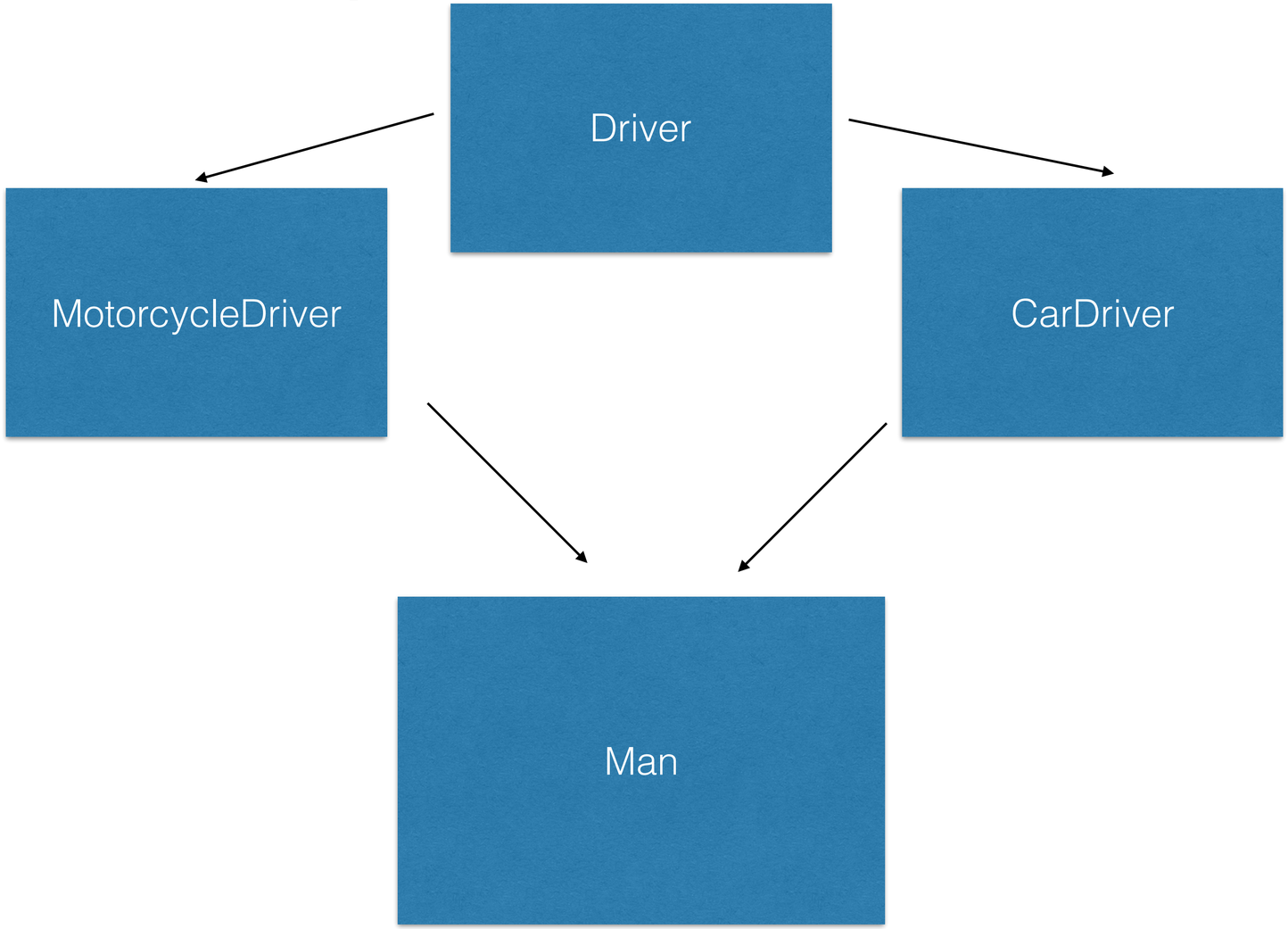
- An abstract class that is inherited by mupletple classes with simple functions. For example class for communication. It must have functions for send, receive and buffer.

Mutual inheritance

Multiple inheritance



Multiple inheritance



Multiple inheritance

- Used when you need to use 2 or more base classes. If you need to redefine members do not forget to make them virtual.

Task

- Make a classes for paint object (with color, size on X and Y axises, sheet size, backgroundColor)
- Make a class for a Whale (size, skinColor, mass)
- Make a class that is used for a paintWhale(adds scaleFactor function based on the maximum size of sheet)

Operator predefinition

Object 1 = Object 2?

Operator predefinition

- When we need to change an operator in a class.
- For example to use aClass = otherClass;

Implementation

```
class ClassA

{

    public:

    ClassA &operator=(const objectType &object);

}
```

Copy constructor

Copy constructor

- Copy constructors are used when we need to copy information from one object to another. They are like the default constructors, but they accept an argument of class type and reference. In the implementation you just copy what you need.

Implementation

```
class ClassA

{

    public:
    ClassA();
    ClassA(classB &obj): aMember(obj.member){};

}
```

Copy constructor with operator predefinition

- We are using the symbol ‘=‘ for making :
- aClass anObject = anotherObjectFromAnotherClass;

Task

- Make a class bankAccount with member balance.
- Make a class Person that is abstract.
- Make a class Man that has member bankAccount.
- Make a class Women that has member bankAccount and a single function wasteMoney.
- Make a main function that firstly initializes an object aMan with balance of money in his bankAccount. Then make an object aWomen that uses copy constructor to transit all the money from aMans bank account to her bank account and the end execute the function aWomen.wasteMoney() which is making the balance in aWomen bankAccount() = 0;