BATCH

RECURSION

C AND C++

PROGRAMMING MASTERCLASS

AWESH ISLAM
BUET, CSE

C++ Class-05

SHAROARE HOSAN EMON
BME , BUET

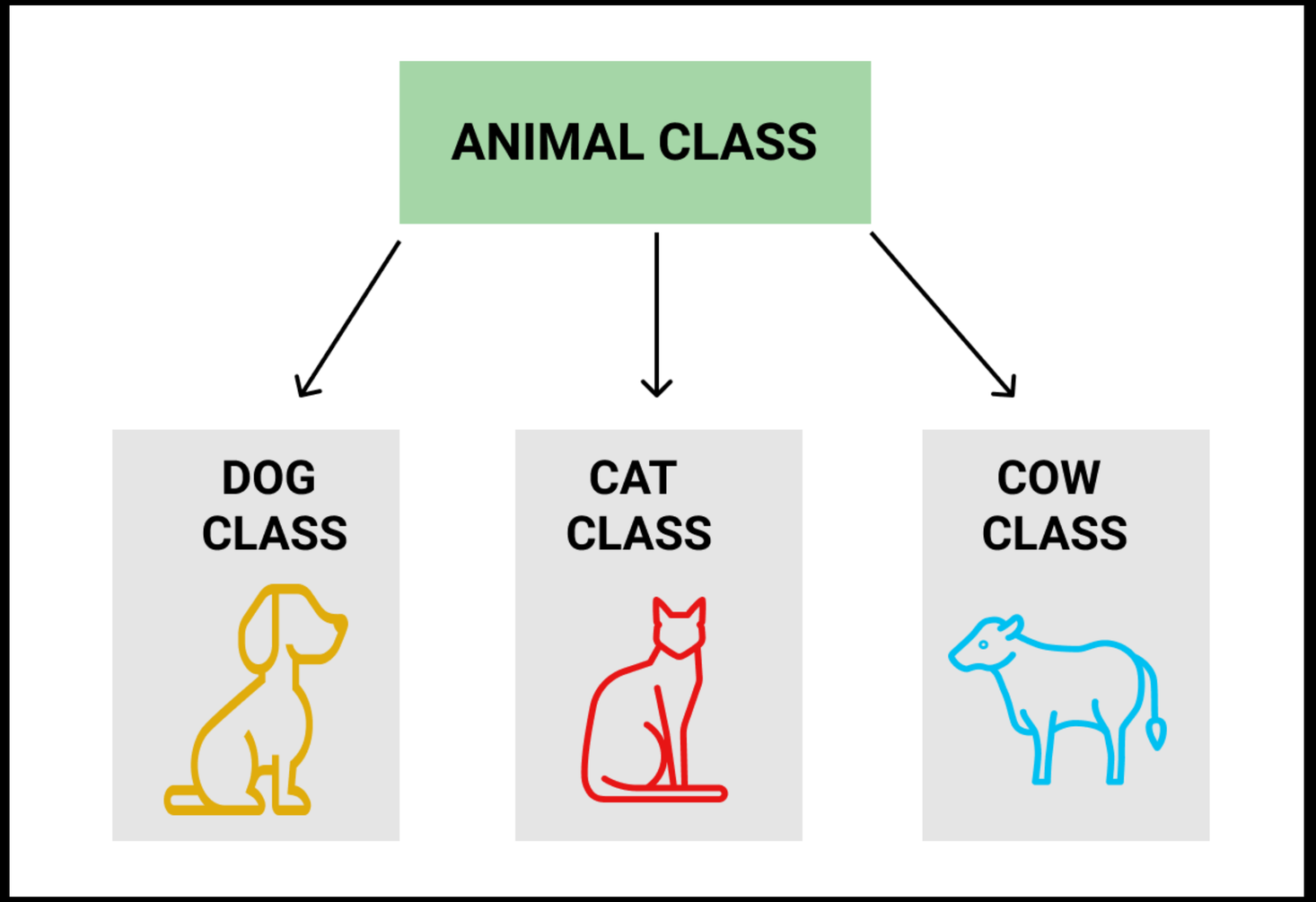
আমাদের সবগুলো ক্লাস দেখার জন্য ভিজিট করো

https://www.hsccrackers.com/

SCAN ME

Inheritance

What is inheritance?



How to Inherit

```
#include <iostream>
using namespace std;
class vehicle{
    int speed;
public:
    void set_speed(int speed){
        this->speed = speed;
    }
    void show(){
        cout<<"Speed is: "<<speed<<endl;
    }
};
class car: public vehicle{
    int doors;
public:
    void set_doors(int doors){
        this->doors = doors;
    }
    void show_door(){
        cout<<"The no of doors of a car: "<<doors<<endl;
    }
};
int main(int argc, const char * argv[]) {
    car myCar;
    myCar.set_speed(200);
    myCar.set_doors(4);
    myCar.show();
    myCar.show_door();
    return 0;
}
```

Access Modifiers

There are three access modifiers:

Public

Private

Protected

Access Modifier: Public

Base class	Public	Private	Protected
Derived Class	Public	Do not have access	Protected

Public members of base class are available to derived class as public members.

```cpp
#include <iostream>
using namespace std;
class vehicle{
    int speed;
public:
    void set_speed(int speed){
        this->speed = speed;
    }
    void show(){
        cout<<"Speed is: "<<speed<<endl;
    }
};
class car: public vehicle{
    int doors;
public:
    void set_doors(int doors){
        this->doors = doors;
    }
    void show_door(){
        cout<<"The no of doors of a car: "<<doors<<endl;
    }
};
int main(int argc, const char * argv[]) {
    car myCar;
    myCar.set_speed(200);
    myCar.set_doors(4);
    myCar.show();
    myCar.show_door();
    return 0;
}
```

Private member not accesible

Private members of base class is not available to derived class.

```
8  #include <iostream>
9  using namespace std;
10 class vehicle{
11     int speed;
12 public:
13     void set_speed(int speed){
14         this->speed = speed;
15     }
16     void show(){
17         cout<<"Speed is: "<<speed<<endl;
18     }
19 };
20 class car: public vehicle{
21     int doors;
22 public:
23     void set_doors(int doors){
24         this->doors = doors;
25     }
26     void show_door(){
27         cout<<"Speed is : "<<speed<<endl;          'speed' is a private member of 'vehicle'
28         cout<<"The no of doors of a car: "<<doors<<endl;
29     }
30 };
31 int main(int argc, const char * argv[]) {
32     car myCar;
33     myCar.set_speed(200);
34     myCar.set_doors(4);
35     myCar.show();
36     myCar.show_door();
37
38     myCar.speed;                                   2  'speed' is a private member of 'vehicle'
39
40     return 0;
41 }
42
```

Access Modifier: Private

Public class	Public	Private	Protected
Derived Class	Private	Do not have access	Private

```cpp
#include <iostream>
using namespace std;
class vehicle{
    int speed;
public:
    void set_speed(int speed){
        this->speed = speed;
    }
    void show(){
        cout<<"Speed is: "<<speed<<endl;
    }
};
class car: private vehicle{
    int doors;
public:
    void set_doors(int doors){
        this->doors = doors;
    }
    void show_door(){
        show();
        cout<<"The no of doors of a car: "<<doors<<endl;
    }
};
int main(int argc, const char * argv[]) {
    car myCar;
    myCar.set_speed(200);
    myCar.set_doors(4);
    myCar.show();
    myCar.show_door();

    return 0;
}
```

Show is available to the class as private

But show and set speed is not available to the class publicly

'set_speed' is a private member of 'vehicle'

'show' is a private member of 'vehicle'

Example of protected

```
8  #include <iostream>
9  using namespace std;
10 class vehicle{
11     int speed;
12 protected:
13     int registration_number;
14 public:
15     string color;
16
17 };
18
19 int main(int argc, const char * argv[]) {
20     vehicle car;
21     car.color = "red";
22     car.registration_number = 1000;
23     return 0;
24 }
```

'registration_number' is a protected member of 'vehicle'

Protected members are like private members
They are not available outside the class

Example of protected

```
8  #include <iostream>
9  using namespace std;
10 class vehicle{
11     int speed;
12 protected:
13     int registration_number;
14 public:
15     string color;
16
17 };
18 class car: public vehicle{
19     int doors;
20 public:
21     void set_doors(int doors){
22         this->doors = doors;
23     }
24     void show_door(){
25         cout<<"The registration number is: "<<registration_number<<endl;
26         cout<<"The no of doors of a car: "<<doors<<endl;
27     }
28 };
29 int main(int argc, const char * argv[]) {
30     vehicle car;
31     car.color = "red";
32     return 0;
33 }
```

But protected member is available to derived class

Access Modifier: Protected

Public class	Public	Private	Protected
Derived Class	Protected	Do not have access	Protected

Access Modifier: Protected

```
8  #include <iostream>
9  using namespace std;
10 class vehicle{
11     int speed;
12 protected:
13     int registration_number;
14 public:
15     string color;
16
17 };
18 class car: protected vehicle{
19     int doors;
20 public:
21     void set_doors(int doors){
22         this->doors = doors;
23     }
24     void show_door(){
25         cout<<"Color is : "<<color<<endl;
26         cout<<"The registration number is: "<<registration_number<<endl;
27         cout<<"The no of doors of a car: "<<doors<<endl;
28     }
29 };
30 int main(int argc, const char * argv[]) {
31     car myCar;
32     myCar.show_door();
33     myCar.color;
34     return 0;
35 }
```

As the access modifier is protected Colour and Registration number became protected member of Car

So color is not available in main

Constructor Call

```
8  #include <iostream>
9  using namespace std;
10 class vehicle{
11     int speed;
12 public:
13     vehicle(){
14         cout<<"Construcor of vehicle..."<<endl;
15     }
16
17 };
18 class car: protected vehicle{
19     int doors;
20 public:
21     car(){
22         cout<<"Constructo of car..."<<endl;
23     }
24     void set_doors(int doors){
25         this->doors = doors;
26     }
27     void show_door(){
28         cout<<"The no of doors of a car: "<<doors<<endl;
29     }
30 };
31 int main(int argc, const char * argv[]) {
32     car myCar;
33     return 0;
34 }
```

```
Construcor of vehicle...
Constructo of car...
Program ended with exit code: 0
```

Destructor Call

```
10  class vehicle{
12  public:
13      vehicle(){
14          cout<<"Construcor of vehicle..."<<endl;
15      }
16      ~vehicle(){
17          cout<<"Destructing vehicle..."<<endl;
18      }
19  };
20  class car: protected vehicle{
21      int doors;
22  public:
23      car(){
24          cout<<"Constructo of car..."<<endl;
25      }
26      ~car(){
27          cout<<"Destructing car..."<<endl;
28      }
29      void set_doors(int doors){
30          this->doors = doors;
31      }
32      void show_door(){
33          cout<<"The no of doors of a car: "<<doors<<endl;
34      }
35  };
36  int main(int argc, const char * argv[]) {
37      car myCar;
38      return 0;
39  }
```

```
Construcor of vehicle...
Constructo of car...
Destructing car...
Destructing vehicle...
Program ended with exit code: 0
```

Constructor Call with arguements

```
10 class vehicle{
11     int speed;
12 public:
13     vehicle(int speed){
14         cout<<"Construcor of vehicle..."<<endl;
15     }
16     ~vehicle(){
17         cout<<"Destructing vehicle..."<<endl;
18     }
19     void show(){
20         cout<<"Speed is : "<<speed<<endl;
21     }
22 };
23 class car: protected vehicle{
24     int doors;
25 public:
26     car(int doors,int speed):vehicle(speed){
27         cout<<"Constructo of car..."<<endl;
28         this->doors = doors;
29     }
30     ~car(){
31         cout<<"Destructing car..."<<endl;
32     }
33     void set_doors(int doors){
34         this->doors = doors;
35     }
36     void show_door(){
37         cout<<"The no of doors of a car: "<<doors<<endl;
38     }
39 };
40 int main(int argc, const char * argv[]) {
41     car myCar(200,4);
42     myCar.show_door();
43     return 0;
44 }
```

Multiple Inheritance

There are two ways a derived class can inherit more than once base class

1) Multilevel Class hierarchy

2) Directly inherit from more than one class

Multilevel Inheritance

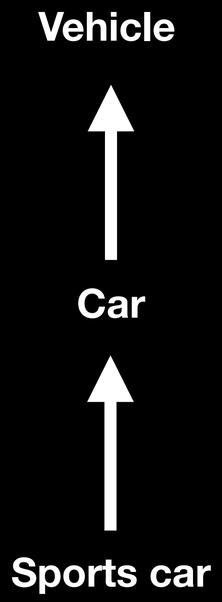
```cpp
#include <iostream>
using namespace std;
class vehicle{
    int speed;
public:
    vehicle(int speed){
        cout<<"Construcor of vehicle..."<<endl;
    }
    ~vehicle(){
        cout<<"Destructing vehicle..."<<endl;
    }
};
class car: public vehicle{
    int doors;
public:
    car(int doors,int speed):vehicle(speed){
        cout<<"Constructor of car..."<<endl;
        this->doors = doors;
    }
    ~car(){
        cout<<"Destructing car..."<<endl;
    }
};
class sports_car:public car{
    string color;
public:
    sports_car(int doors,int speed,string color):car(doors, speed){
        cout<<"Consturtor of sports car...";
        this->color = color;
    }
    ~sports_car(){
        cout<<"Destructing Sports Car..."<<endl;
    }
}
```

```cpp
int main(int argc, const char * argv[]) {
    sports_car(2,300,"red");
    return 0;
}
```

```
Construcor of vehicle...
Constructor of car...
Consturtor of sports car...
Destructing Sports Car...
Destructing car...
Destructing vehicle...
Program ended with exit code: 0
```

Constructor are called from base to derived to derived2
And destructor is called opposite

Multilevel Inheritance



Multiple Inheritance

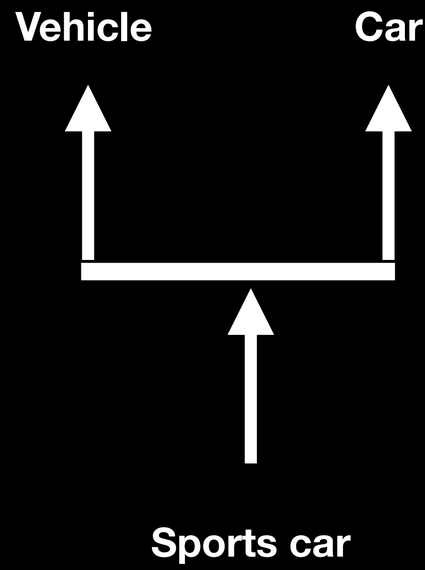
```cpp
#include <iostream>
using namespace std;
class vehicle{
    int speed;
public:
    vehicle(int speed){
        cout<<"Construcor of vehicle..."<<endl;
    }
    ~vehicle(){
        cout<<"Destructing vehicle..."<<endl;
    }
};
class car{
    int doors;
public:
    car(int doors,int speed){
        cout<<"Constructor of car..."<<endl;
        this->doors = doors;
    }
    ~car(){
        cout<<"Destructing car..."<<endl;
    }
};
class sports_car:public car,public vehicle{
    string color;
public:
    sports_car(int doors,int speed,string color):car(doors, speed),vehicle(speed){
        cout<<"Consturtor of sports car..."<<endl;
        this->color = color;
    }
    ~sports_car(){
        cout<<"Destructing Sports Car..."<<endl;
    }
};
```

```cpp
int main(int argc, const char * argv[]) {
    sports_car(2,300,"red");
    return 0;
}
```

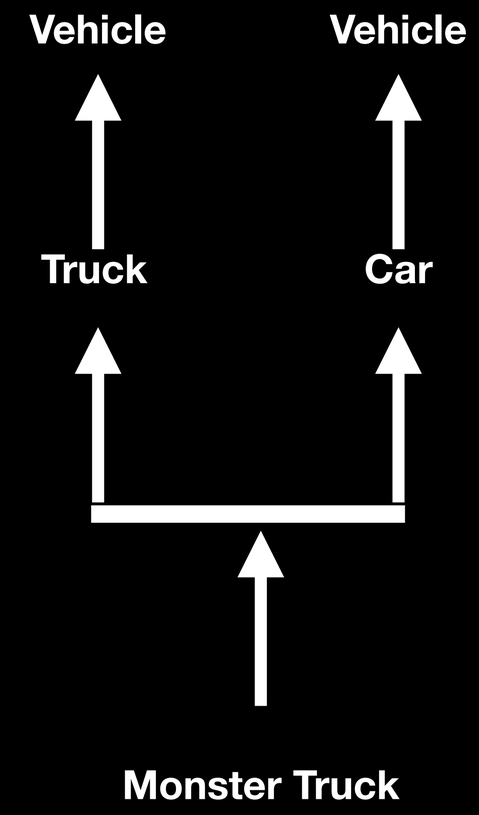
```
Constructor of car...
Construcor of vehicle...
Consturtor of sports car...
Destructing Sports Car...
Destructing vehicle...
Destructing car...
Program ended with exit code: 0
```

Constructor is called from
Left to right and destructor oppositely.

Multilevel Inheritance



Virtual Base Classes



Virtual Base Classes

```cpp
#include<iostream>
using namespace std;
class vehicle{
    int weight;
public:
    vehicle(int weight){
        this->weight = weight;
    }
    void show(){
        cout<<"the weight is: "<<weight<<endl;
    }
};
class truck:virtual public vehicle{
    int speed;
public:
    truck(int weight,int speed):vehicle(weight){
        this->speed = speed;
    }
    void show_truck(){
        cout<<"the speed is: "<<speed<<endl;
    }
};
class car:virtual public vehicle{
    int speed;
public:
    car(int weight,int speed):vehicle(weight){
        this->speed = speed;
    }
    void show_car(){
        cout<<"the speed is: "<<speed<<endl;
    }
};
```

```cpp
class monster_truck:public car,public truck{
    string color;
public:
    monster_truck(int weight,int speed,string color):car(weight, speed),truck(weight, speed),vehicle(weight){
        this->color = color;
    }
    void show_monster(){
        show();
        show_car();
        cout<<"the color is: "<<color<<endl;
    }
};
int main(){
    monster_truck myTruck(2000,200,"red");
    myTruck.show_monster();
}
```