Lecture 4.2

Classes and Objects

How to define a class?

- Using the keyword class followed by a programmer-specified name followed by the class definition in braces.
- The class definition contains the class members (data) and the class methods (functions).

How to define a class (cont)

```
class Dog {
    public:
        void setAge(int age);
        int getAge();
        void setWeight(int weight);
        int getWeight();
        void speak();
    private:
        int age;
        int weight;
};
```


Some important concepts

- **private** indicates that the two members, age and weight, cannot be directly accessed from outside of the class.
- **public** indicates that the methods, can be called from code outside of the class. They may be called from other parts of a program.
- Allowing access and manipulation of data members only through methods is referred to as **data hiding**.

Method implementation

- The methods were declared but not defined. That is, an implementation for each method must be written.

```
void Dog::setAge(int age)
{ this->age = age; }

int Dog::getAge() { return age; }
```


Method implementation (cont)

```
void Dog::setWeight(int weight)
{ this->weight = weight; }

int Dog::getWeight()
{ return weight; }

void Dog::speak()
{ cout << "BARK!!" << endl; }
```


Some important things!

- The methods are implemented outside of the class definition, they must be identified as belonging to that class.
- This is done with the scope resolution operator, "**::**".
- Every object has a special pointer call "this", which refers to the object itself.
- The members of the Dog class can be referred to as this->age or this->weight, as well as, age or weight.

And some more important things (cont)

- If there is no ambiguity, no qualification is required.
- In the getWeight method, "weight" can be used instead of "this->weight".

```
int Dog::getWeight()
{ return weight; }
```

- Here, the scope resolution operator must be used.

```
void Dog::setWeight(int weight)
{ this->weight = weight; }
```

Data Abstraction

- Describing the functionality of a class independent of its implementation is called data abstraction and c++ classes define so-called abstract data types.
- We make project in Dev C++ to separate declaration of methods with their implementations. Technique of strictly hiding the implementations is outside the scope of this course.
- This is done to hide the implementation details from the clients of the classes.

Constructors

- Each class also has a special method, the constructor, which is called when an object of the class is instantiated (created).
- The constructor can be used to initialize variables, dynamically allocate memory or setup any needed resources.

Destructor

- Another special method, the destructor, is called when an object is destroyed.
- An object is destroyed when it goes out of scope.
- If an object is created within a function, it will go out of scope when the function exits.

Destructor (cont)

- Since your program is the "main" function, all its objects go out of scope when the program ends.
- The destructor is used to free any memory that was allocated and possible release other resources.