- Base vs derived class
    - Base
        - parent class from which other classes (Derived Classes) inherit
    - Derived
        - subclass that inherits properties and behaviors of a Base Class

```
class Base {
public:
   void display() {
      cout << "This is the Base class." << endl;
   }
};

class Derived : public Base {
public:
   void show() {
      cout << "This is the Derived class." << endl;
   }
};
```

- Overloading vs overriding functions
    - Overloading
        - Same function name with different parameter lists in the same or related classes
    - Overriding
        - Derived Class redefines a function from the Base Class using the same signature

```
class Base {
public:
   void display(int x) { // Overloading in Base
      cout << "Base Display with int: " << x << endl;
   }
   virtual void show() { // Virtual function to enable overriding
      cout << "Base Show" << endl;
   }
};

class Derived : public Base {
public:
   void display(double y) { // Overloading in Derived
```

```
        cout << "Derived Display with double: " << y << endl;
    }
    void show() override { // Overriding
        cout << "Derived Show" << endl;
    }
};
```

- constructor and destructor for derived class
    - Constructor
        - Derived Class constructor can initialize both its own members and those of the Base Class
    - Destructor
        - Derived Class destructor automatically calls the Base Class destructor after finishing its own cleanup

```
class Base {
public:
    Base() { cout << "Base Constructor" << endl; }
    virtual ~Base() { cout << "Base Destructor" << endl; }
};

class Derived : public Base {
public:
    Derived() { cout << "Derived Constructor" << endl; }
    ~Derived() { cout << "Derived Destructor" << endl; }
};
```

- Public vs private vs protected
    - Public
        - Can be changed anywhere
    - Private
        - Can only be changed in Base
    - Protected
        - Can be changed in Base and Derived

```
class Base {
protected:
    int protectedVar;
private:
    int privateVar; // Not inherited
```

```
public:
    int publicVar;
};

class Derived : public Base { // Public Inheritance
public:
    void display() {
        protectedVar = 10; // Accessible
        publicVar = 20;    // Accessible
    }
};
```

- UML diagram