Object Oriented Programming

OOP ’70

Alan Kay = “Smalltalk”

• Encapsulation: object hides its member variables
• Inheritance:
• Polymorphism:

• Inheritance: the ability to declare a class as an extension of another class.

```cpp
class Dog {
    int x;
    public:
        int y;
        void bark();
};

class Pug : public Dog {
    public:
        int z;
        void snore();
};

Dog fido;
Pug jeff;
```

Pug inherits from Dog

• The derived class inherits all members (variables & functions)
  from the base class except constructors
```cpp
void myFunction( Dog& d ){
    ...
}
myFunction( jeff );
Dog *p = &jeff;
Dog larry( jeff );
fido = jeff;
jeff = larry;
```

```cpp
class Dog {
    int x;
public:
    int y;
    void bark() { cout << "Woof!"; }
}

class Pug : public Dog {
public:
    int z;
    void snore();
    void bark() { cout << "puff!"; }
}
```

```cpp
Pug tim;
tim.bark();  // "puff!"
tim.Dog::bark();  // "Woof!"
```

The type of a derived class:
a derived class will be considered the type of the base class.

void myFunction( Dog& d ){
    ...
}
myFunction( jeff );
Dog *p = &jeff;
Dog larry( jeff );
fido = jeff;
jeff = larry;

An object of a derived class is compatible anywhere an object of the base class is requested.

Redefining member functions.
```
Inherited Private members:

```cpp
class Dog {
    protected:
        int x;
    public:
        int y;
        void bark();
}

class Pug : public Dog {
    public:
        int z;
        void snore() {
            x = 3;
        }
}
```

Constructors of derived classes:
- Any constructor of a derived class call a constructor of the base class before executing its own body.

```cpp
class Alpha {

    int x, y;
    Alpha() : x(0), y(0) {}  
    Alpha(int a, int b) : x(a), y(b) {} 
}

class Beta : public Alpha {

    int z;
    Beta() : Alpha(1, 1), z(1) {} 
}

class Gamma : public Beta {

    int w;
    Gamma() : w(0) {} 
}
```
Destructors of Derived Classes

The destructor of a derived class will automatically call the destructor of the base class after executing its own destructor.

class Alpha
{
    int *p;
    ~Alpha() { delete [] p;}
}

class Beta : public Alpha
{
    string *z;
    ~Beta() { delete z}
}

class Gamma : public Beta
{
    char *w;
    ~Gamma { delete w}
}