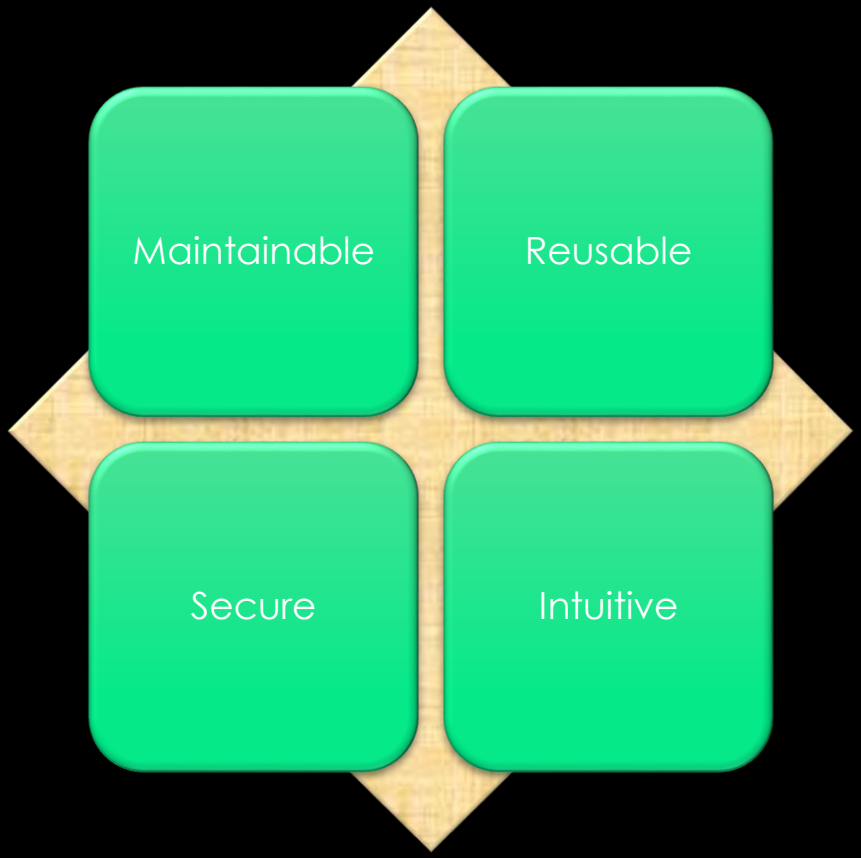
OBJECT ORIENTED PROGRAMMING

Sumeet Ranka
Naman Jain

WHY OOP ?

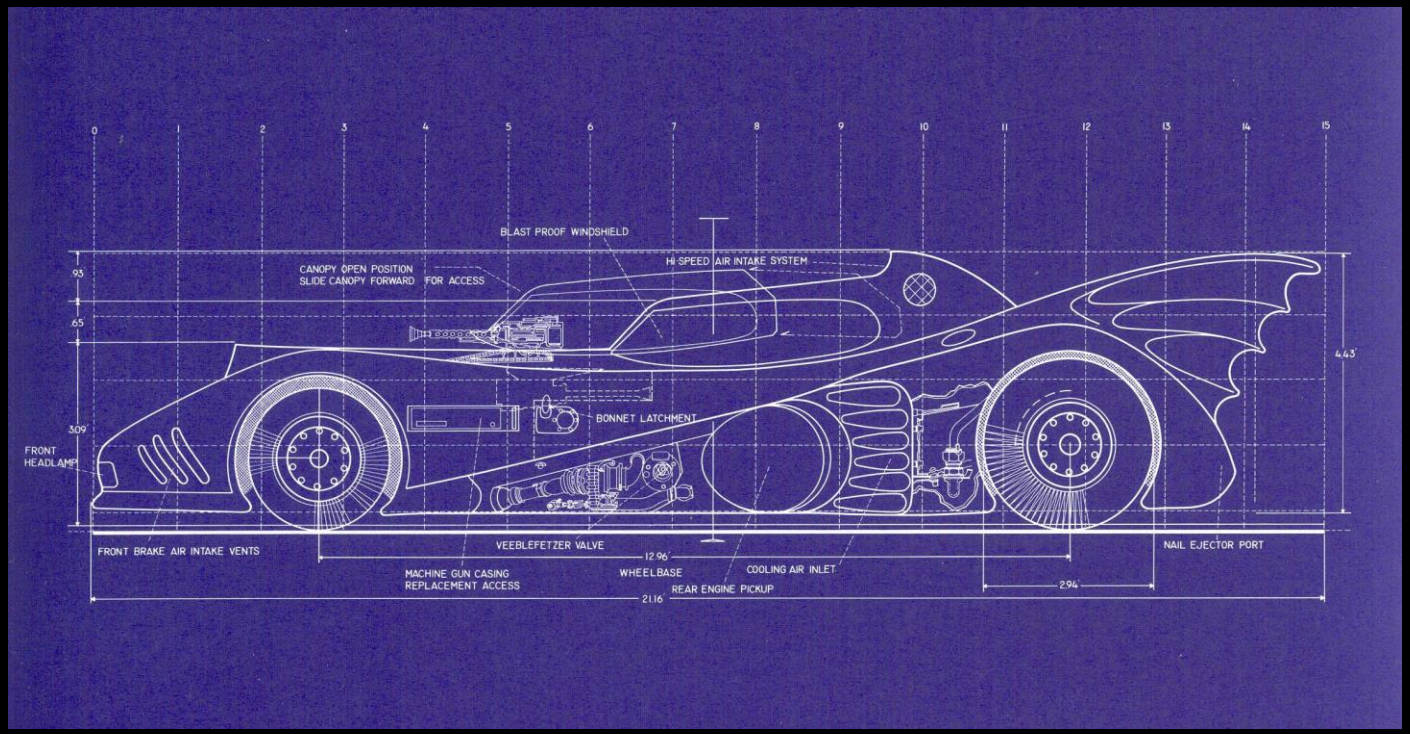


BUT

Size

Effort

CLASSES


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
BLAST PROOF WINDSHIELD
HI SPEED AIR INTAKE SYSTEM
CANOPY OPEN POSITION
SLIDE CANOPY FORWARD FOR ACCESS
.93
.65
3.09
4.43
BONNET LATCHMENT
FRONT
HEADLAMP
FRONT BRAKE AIR INTAKE VENTS
VEEBLEFETZER VALVE
12.96
WHEELBASE
MACHINE GUN CASING
REPLACEMENT ACCESS
COOLING AIR INLET
REAR ENGINE PICKUP
2.94
NAIL EJECTOR PORT
21.16

OBJECTS

AN EXAMPLE

```
class smallobj {     //define a class
    private:
        int somedata;
    public:
        void setdata(int d) {      //member function to set data
            somedata = d;
        }
        void showdata() {            //member function to display data
            cout << "Data is " << somedata << endl;
        }
};
```

```
void main() {

    smallobj s1, s2;       //define two objects of class smallobj
    s1.setdata(1066);
    s2.setdata(1776);    //call member function to set data
    s1.showdata();
    s2.showdata();

}
```

DATA-HIDING

Private

- Data and functions are directly accessible only from within the class defining them.

Public

- Data and functions are directly accessible globally.

CONSTRUCTORS

- Member functions that are called automatically when an object is instantiated
- Posses the same name as that of the class and don't have a return type

```
smallobj( int init){    // Constructor

    cout << "   Initialized to "<<init  << endl;
    somedata=init;

}
```

DESTRUCTORS

- Parameterless member functions that are called automatically when an object is erased or it goes out of scope
- Starts with a tilde "~" followed by the class name
- Used for cleanup like deallocating dynamic memory, closing opened files, network connections etc.

```
~smallobj(){    // Destructor

    cout << "   Initialization undone "  << endl;
}
```

COMMON TRAITS OF OOP LANGUAGES

- Encapsulation
- Inheritance
- Polymorphism

ENCAPSULATION

- Encapsulation is the first defining characteristic of the object model.
- Objects and encapsulation are synonymous.
- Seals attributes and behaviors together into a single unit.

STATIC

- How is memory allocated for object?
- Are objects kind enough to share? YES!

STATIC

- Do they really share?
- Does they belong to class or objects?
- If belong to class then how can we call?

STATIC

- Are only data member static?
- Can it access other variables? Because NO OBJECTS REQUIRED!!!

CONSTANT

- How to protect from modifying any data member?
  - Declare all of them constant! – Tedious
  - What if you want to change but not always?
- Const member functions
- Const data members
- Const objects

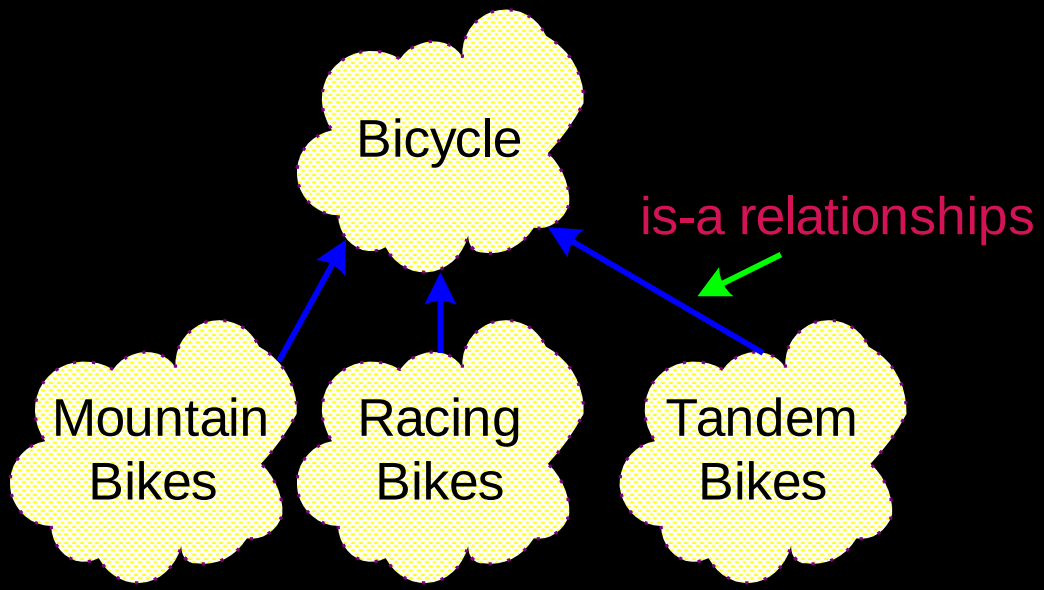
‘CONST’ TEST

- char* p = “Hello”;
- char p[] = “Hello”;
- const char* p = “Hello”;
- char* const p = “Hello”;
- char const *p = “Hello”;
- const char* const p = “Hello”;

- Apply operations:
  - p = p+1;
  - p = “Bye”;
  - *p = ‘M’;

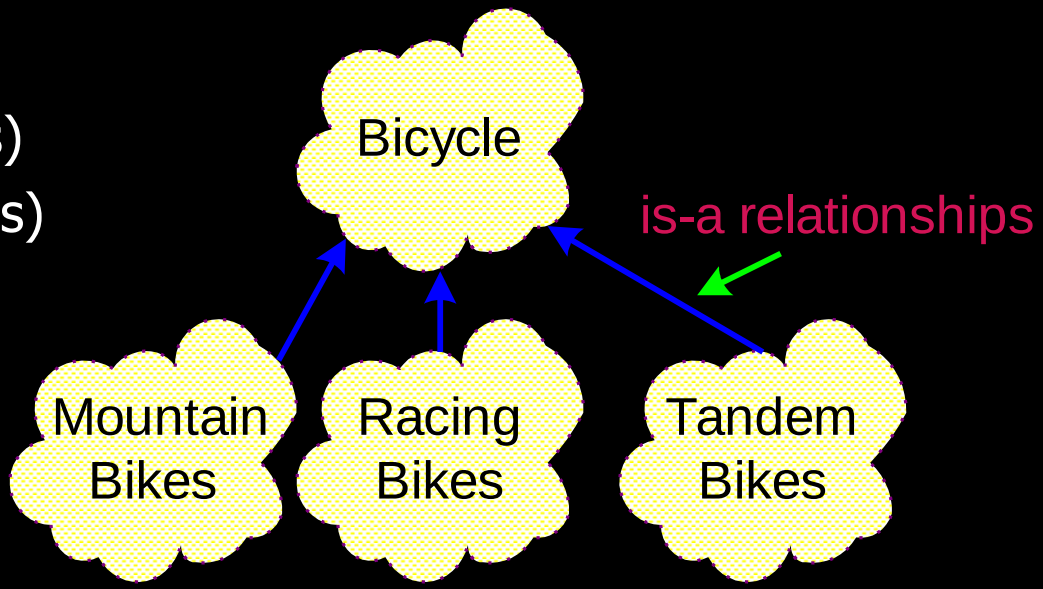
INHERITANCE

- Ability to define new classes of objects using existing classes as a basis
  - The new class inherits the attributes and behaviors of the parent classes
  - New class is a specialized version of the parent class



INHERITANCE

- A natural way to reuse code (implements Code Reusability)
  - Programming by extension rather than reinvention
  - Object-oriented paradigm is well-suited for this style of programming
- Terminology
  - Base class (superclass)
  - Derived class (subclass)



POLYMORPHISM

- Selection of the correct method is deferred until run-time
- when the selection is based on the current object
  - Early binding: Func to call known at compile time
  - Late binding: Func to call known at run time
- Objects respond differently to the same message

THANK YOU

- This covers the basic concepts of OOP
- Next class :
  - Inheritance
  - Friend
  - Virtual
  - Mutable