Object Oriented Programming

Procedure Oriented-Programming

- In procedure oriented approach things to be done in a sequence such as reading calculating and printing.
- It consist of basically writing a list of instructions for the computer to follow and organizing these instructions into a group called function.
- Example: COBOL, FORTRAN and C

Problem in POP:

- In a very large program it is very difficult to identify what data is used by which function.
- It does not model real world problem very well.

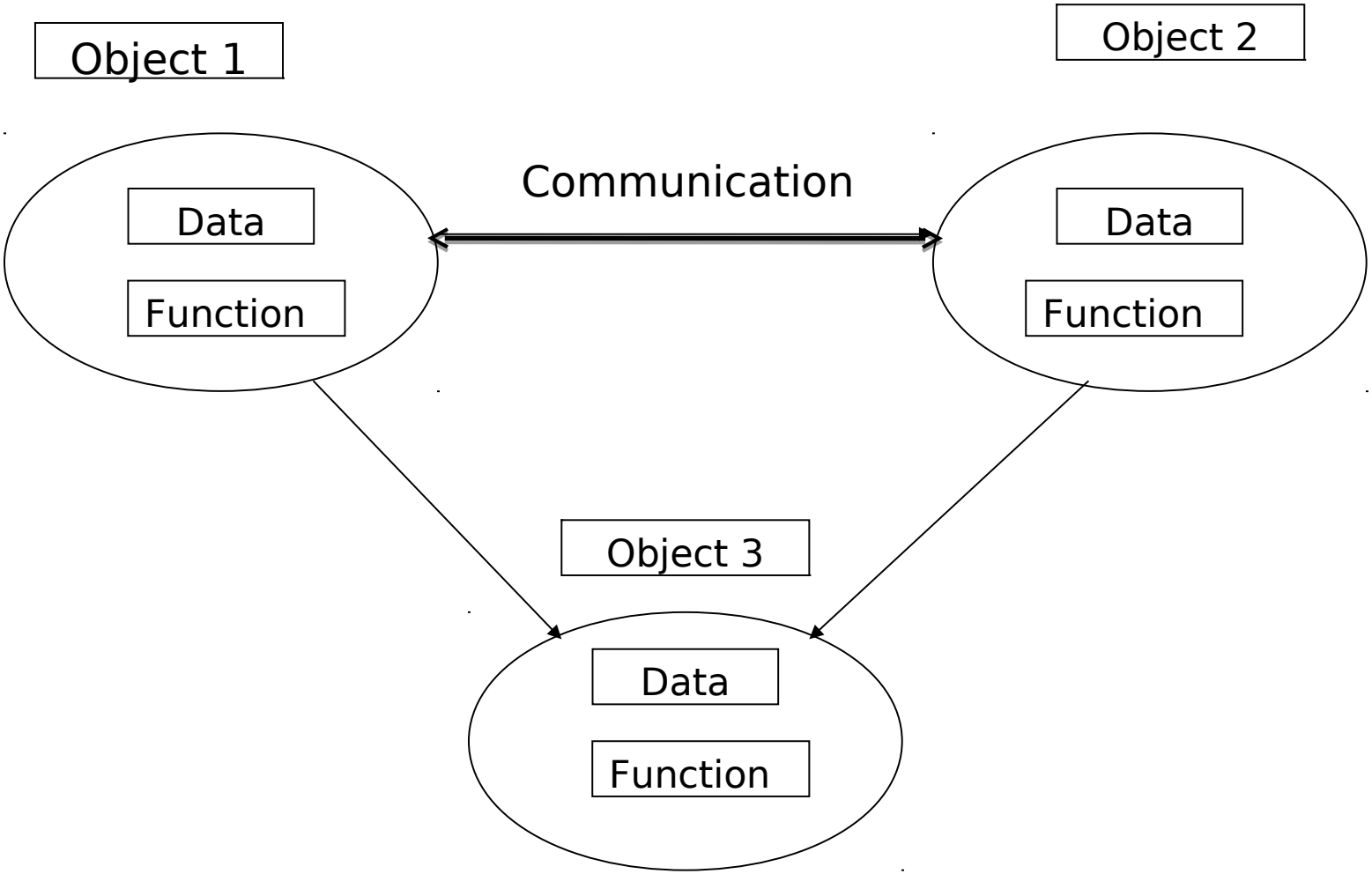
Object Oriented Programming

- OOP allow decomposition of a problem into a number of entities called object and builds data and functions around these objects.
- Objects have both data and methods
- Objects of the same class have the same data elements and methods
- Objects send and receive *messages* to invoke actions/functions

Key idea in object-oriented:

- *The real world can be accurately described as a collection of objects that interact.*

OBJECT ORIENTED PROGRAMMING



Basic terminology

- ***object*** :It is a basic run time entities in an object-oritend system. Example: usually a person, place or thing (**a noun**), bank account.
- ***Class***: It is collection of similar kind of object. Example: mango, apple and orange are members of the class fruits.
- ***method***

- an action performed by an object (**a verb)**

- ***attribute***

- description of objects in a class

Why OOP?

- Save development time (and cost) by reusing code

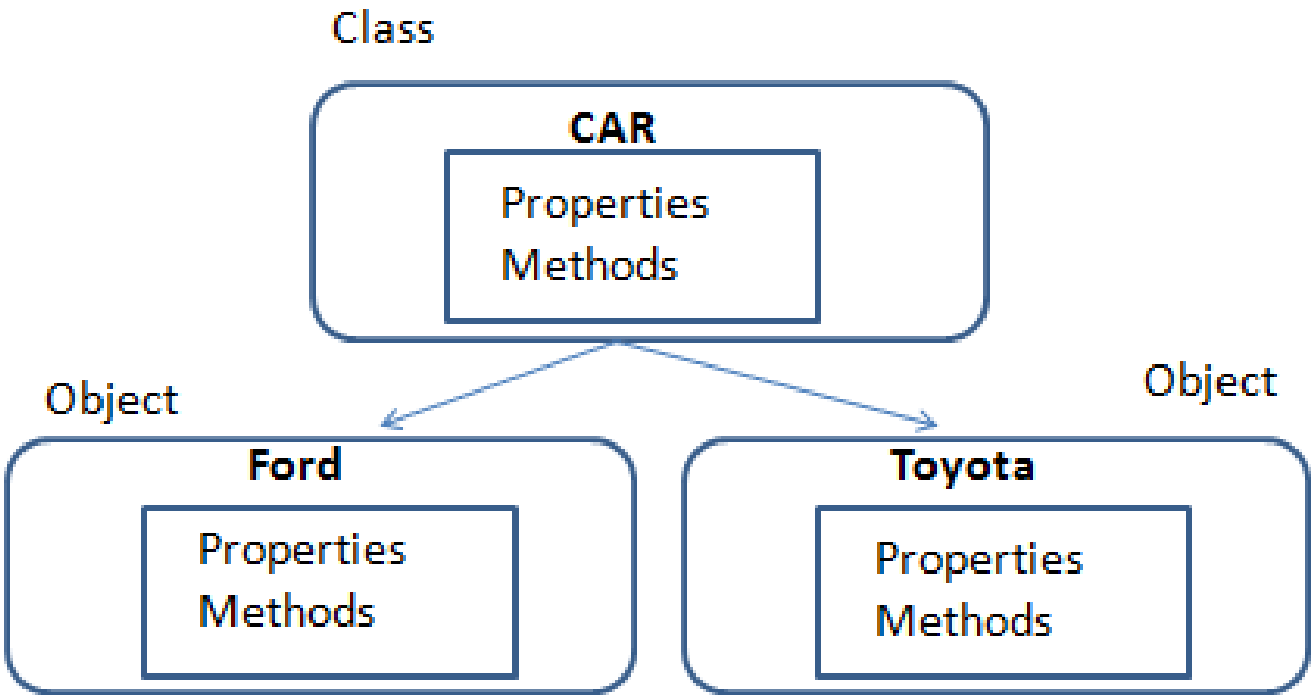
  once an object class is created it can be used in other applications

- Easier debugging

  classes can be tested independently

  reused objects have already been tested

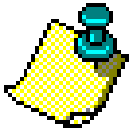
  - New data and functions can be easily added whenever necessary.


Class
CAR
Properties
Methods
Object
Ford
Properties
Methods
Object
Toyota
Properties
Methods

Properties of OOP

Four main properties of Object-Oriented Programming(OOP):

- Encapsulation
- Abstraction
- Polymorphism
- Inheritance



Encapsulation

- Wrapping up of data and functions into a single unit (class) is known as encapsulation.

- Data is not accessible to outside of the class, only those functions which are wrapped in the class can access it.

- It is also known as *data hiding*. Insulation of data from direct access by the program is called data hiding.

- Only object's methods can modify information in the object.

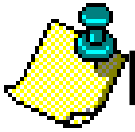
Analogy:

Abstraction

- Abstraction represent the essential features without including the background detail.
- Focus only on the important facts about the problem.
- Classes include the concept of abstraction and defined as list of abstracted attributes such as size, weight and cost.
- Since the classes use the concept of data abstraction, they are known as Abstract data Types (ADT).

Analogy:

- When you drive a car, you don't have to know how the gasoline and air are mixed and ignited.
- Instead you only have to know how to use the controls.
- Draw map



Polymorphism

- Ability to take more than one form.
- An operation may exhibit different behavior at different instance.

Operator Overloading:

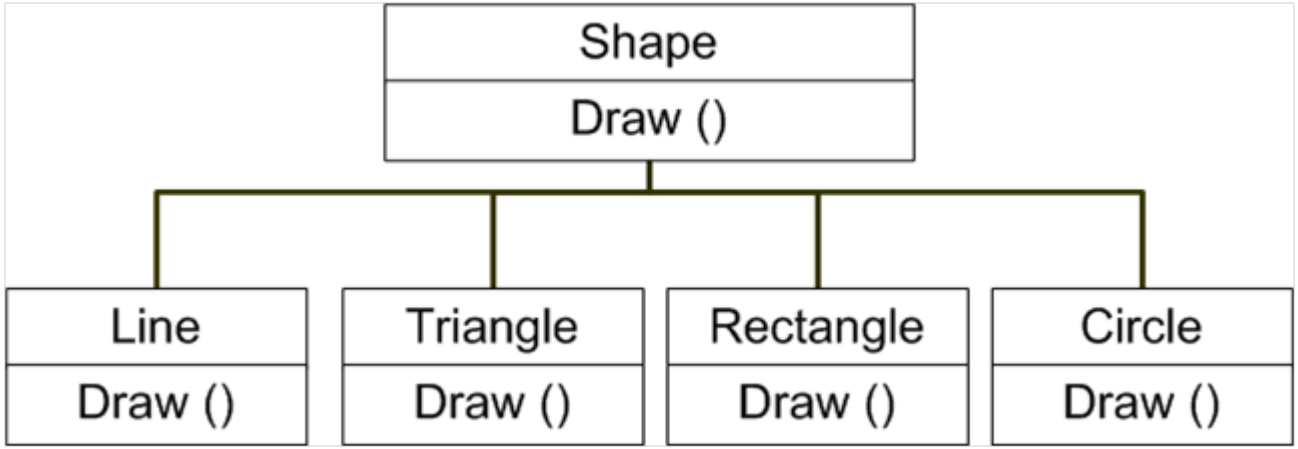
For two numbers the operation will generate a sum. If operands are string then operation would produce third string by concatenation.

The process of making an operator to exhibit different behavior in different instances is known as operator overloading.

For example, we can overload an operator '+' in a class like String so that we can concatenate two strings by just using +.
Other example classes where arithmetic operators may be overloaded are Complex Number,

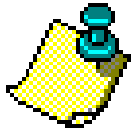
Function Overloading

- Using of single function name to perform different type of tasks is known as function overloading.
- Single function name can be used with different argument type and number.




```
int max(int a, int b) {
    if (a>= b)
        return a;
    else
        return b;
}
```

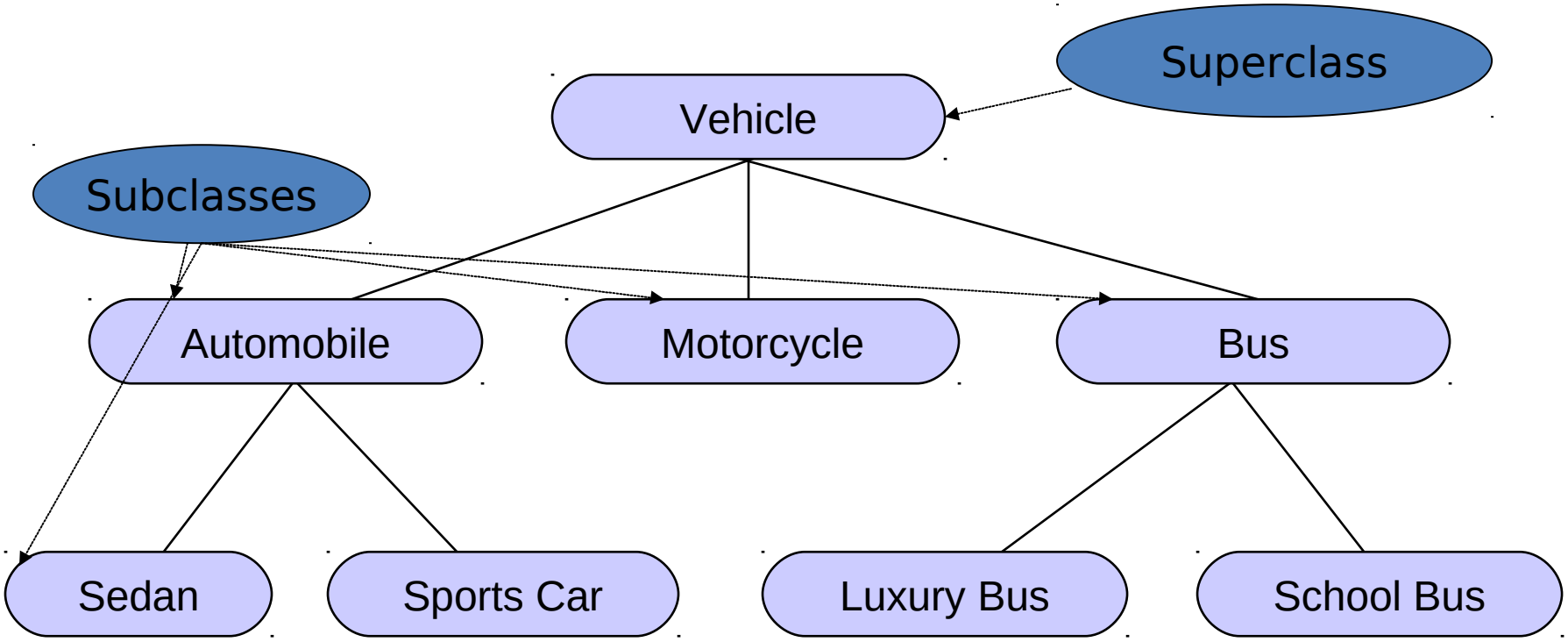
```
float max(float a, float b) {
    if (a>= b)
        return a;
    else
        return b;
}
```



Inheritance

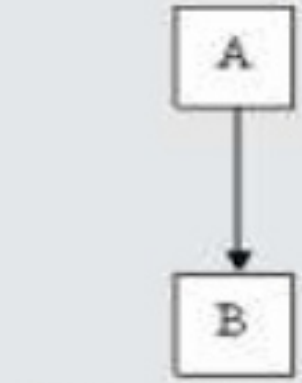
- *Inheritance*—It is the process by which object of one class acquire the properties of object of another class.
- Classes with properties in common can be grouped so that their common properties are only defined once and can be used by other classes.
- Inheritance provide the concept of code reusability.
- When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an

An Inheritance Hierarchy

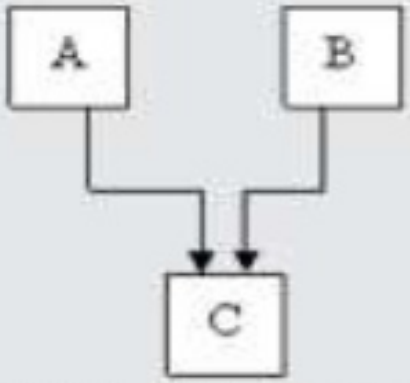


What properties does each vehicle inherit from the types of vehicles above it in the diagram?

FORMS OF INHERITANCE

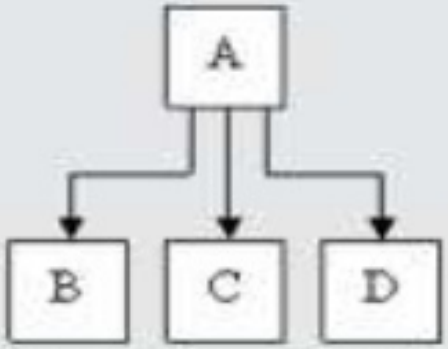


SINGLE INHERITANCE

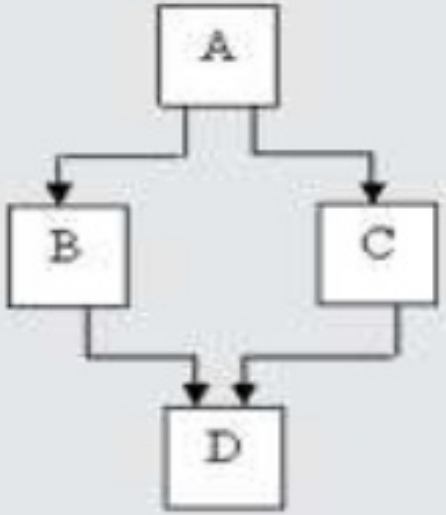


MULTIPLE INHERITANCE

MULTILEVEL INHERITANCE



HIERARCHICAL INHERITANCE



HYBRID INHERITANCE

Benefits of oop

- Through inheritance we can eliminate redundant code and extend the use of existing class.
- Data hiding helps the programmer to build secure programs.
- It is easy to partition the work in a project based on objects.
- Software complexity can be easily managed

Object-Oriented Programming Languages

- Pure OO Languages

  Smalltalk, Java

- Hybrid OO Languages

  C++, Objective-C, Object-Pascal

Introduction

- C++ is the C programmer’s answer to Object-Oriented Programming (OOP).
- C++ is an enhanced version of the C language.
- C++ adds support for OOP without sacrificing any of C’s power, elegance, or flexibility.
- The object-oriented features in C++ allow the programmers to build large programs with clarity, extensibility and ease of maintenance.

Classes and Objects

- A **class** is a data type that allows programmers to create objects. A class provides a definition for an object, describing an object's attributes (data) and methods (operations).
- An object is an *instance* of a class. With one class, you can have as many objects as required.
- This is analogous to a variable and a data type, the class is the data type and the object is the variable.

Two Versions of C++

- A traditional-style C++ program -

```
#include <iostream.h>

int main()
{
        /* program code */
        return 0;
}
```

Two Versions of C++ (cont.)

- A modern-style C++ program that uses the new-style headers and a namespace -

```
#include <iostream>
using namespace std;

int main()
{
        /* program code */
        return 0;
}
```

Namespaces

- Namespace is a new concept introduced by ANSI c++ standards committee.
- This defines the scope for the identifiers used in the program.
- It localizes the names of identifiers to avoid name collisions.
- **std** is a namespace where ANSI c++ standard libraries are defined.
- A newly created class, function or global variable can put in an existing namespace, a new namespace, or it may not be associated with any namespace.

C++ Program

```cpp
#include <iostream>
using namespace std;

int main()
{
      int divisor, dividend, quotient, remainder;

      cout << "Enter dividend: ";
      cin >> dividend;

      cout << "Enter divisor: ";
      cin >> divisor;

      quotient = dividend / divisor;
      remainder = dividend % divisor;

      cout << "Quotient = " << quotient << endl;
      cout << "Remainder = " << remainder;

      return 0;
}
```

Output

```
Enter dividend: 13
Enter divisor: 4
Quotient = 3
Remainder = 1
```

C++ Program

```cpp
#include <iostream>
using namespace std;

class Student {
    int rollNo;
    char name[30];

    public:
    void getData() {
        cout<<"Enter student id and name\n";
        cin>>rollNo>>name;
    }
    void display() {
        cout<<"\nStudent Id : "<<rollNo<<"\n";
        cout<<"Student Name : "<<name<<"\n";
    }
};

int main() {
    Student s1;
    s1.getData();
    s1.display();
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

class Student {
    int rollNo;
    char name[30];

    public:
    void getData();
    void display();
};

void Student :: getData() {
    cout<<"Enter student id and name\n";
    cin>>rollNo>>name;
}
void Student :: display() {
    cout<<"\nStudent Id : "<<rollNo<<"\n";
    cout<<"Student Name : "<<name<<"\n";
}

int main() {
    Student s1;
    s1.getData();
    s1.display();
    return 0;
}
```

Procedure Oriented Programming (POP) Vs. Object Oriented Programming (OOP)

	POP	OOP
Divided Into	Functions	Objects
Importance	Functions	Data
Approach	Top-down	Bottom up
Access Specifiers	No	Yes, Private, Public etc.
Data	Can move freely	Objects can move and communicate
Moving	Function to Function	Objects can move through member functions
Expansion	Adding data s not easy	Adding data is easy
Data Access	Use global data for sharing	Data can not be move easily. Controlled by Acesss specifiers
Data Hiding	Does not have proper way of hiding	Provides Data hiding
Overloading	No	Yes