Friend Functions, Inheritance

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

Friend Function

- Private data member of a class can not be accessed by an object of another class
- Similarly protected data member function of a class can not be accessed by any other class except its derived classes
- The purpose of the friend function is to bring the objects of different classes on same platform and access its private and protected data members.

Friend Function

- Following points must be noted for the use of a friend function
  - Friend function must be defined outside of all classes for which friend function is declared or not
  - The prototype of friend function must be define in each class within scope of any access-specifier
  - The reserve word friend must be only preceded by the prototype of friend function not the definition part of friend function
  - The arguments of friend function may be of class type
  - Friend function can return data of any primitive type but sometime you can use void

Friend Function

- Use of friend function can enhance performance.

Example

```
#include "stdafx.h"
#include "iostream"
class number2;
class number1
{
private:
        int x;
public:
        number1(){
                x=5;
        }
        friend int sum(number1,number2);
};
```

```
class number2
{
private:
        int y;
public:
        number2(){
                y=20;
        }
        friend int sum(number1,number2);
};
```

```
int sum(number1 ob1,number2 ob2)
        {
                return ob1.x+ob2.y;
        }

        int _tmain(int argc, _TCHAR* argv[])
{
        number1 obj1;
        number2 obj2;
        std::cout<<"Sum of private data member
"<<sum(obj1,obj2)<<std::endl;
        system("PAUSE");
        return 0;
}
```

Example 2

```
#include "stdafx.h"
#include <iostream>
using namespace std;
 class count
 {
private:
        int x;
 public:
 count() : x( 0 )
 {
 }
 void print() const
 {
 cout << x << endl;
 }
friend void setX(count &c, int val);
 };
```

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Example 2

```
 void setX(count &c, int val)
{
 c.x=val;
}
 int _tmain(int argc, _TCHAR* argv[])
 {
 count obj;
 cout<<"counter.x after instantiation: ";
 obj.print();

setX(obj, 8);
 cout<<"counter.x after call to setX friend function: ";
 obj.print();
 system("PAUSE");
 return 0;
}
```

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Inheritance Basics

Tahir Naw
Assistant Pro
UOL Sargodha Campus

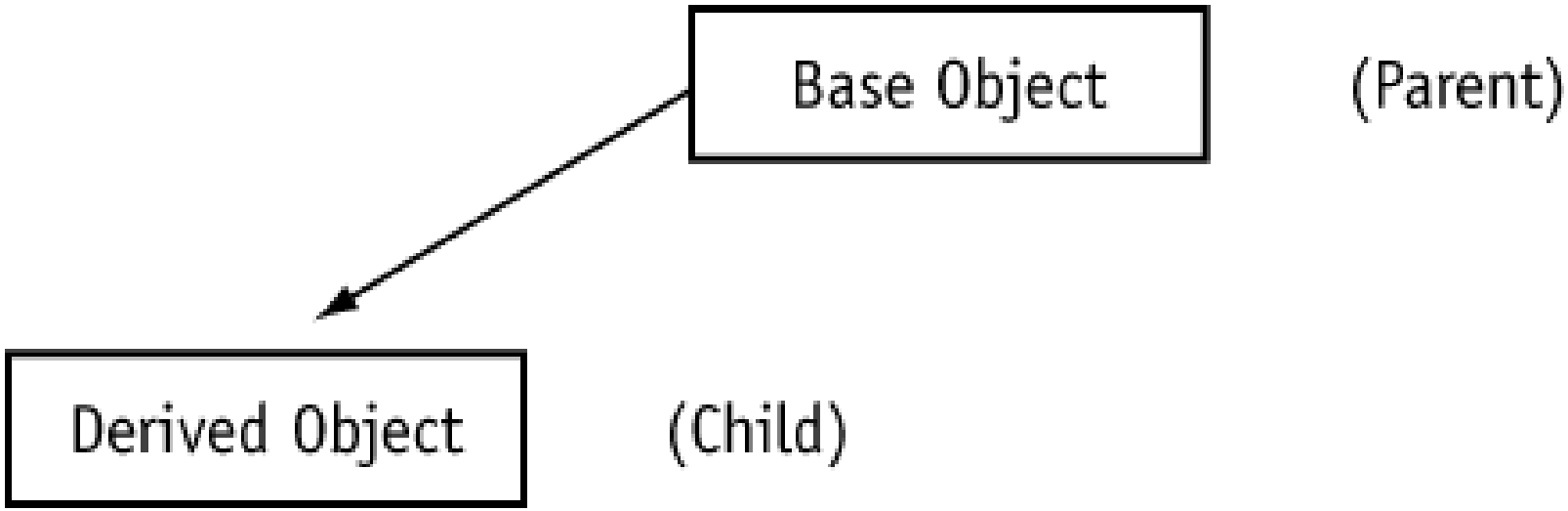
What is Inheritance?

- Inheritance allows a new class to be based on an existing class. The new class inherits all the member variables and functions of the class it is based on.

- **"The mechanism by which one class acquires the properties of another class"**

Inheritance

This existing class is called the base class, and the new class is called the derived class.
Other programming languages, such as Java and C#, refer to the base class as the superclass and the derived class as the subclass. A derived class represents a *more specialized* group of objects.



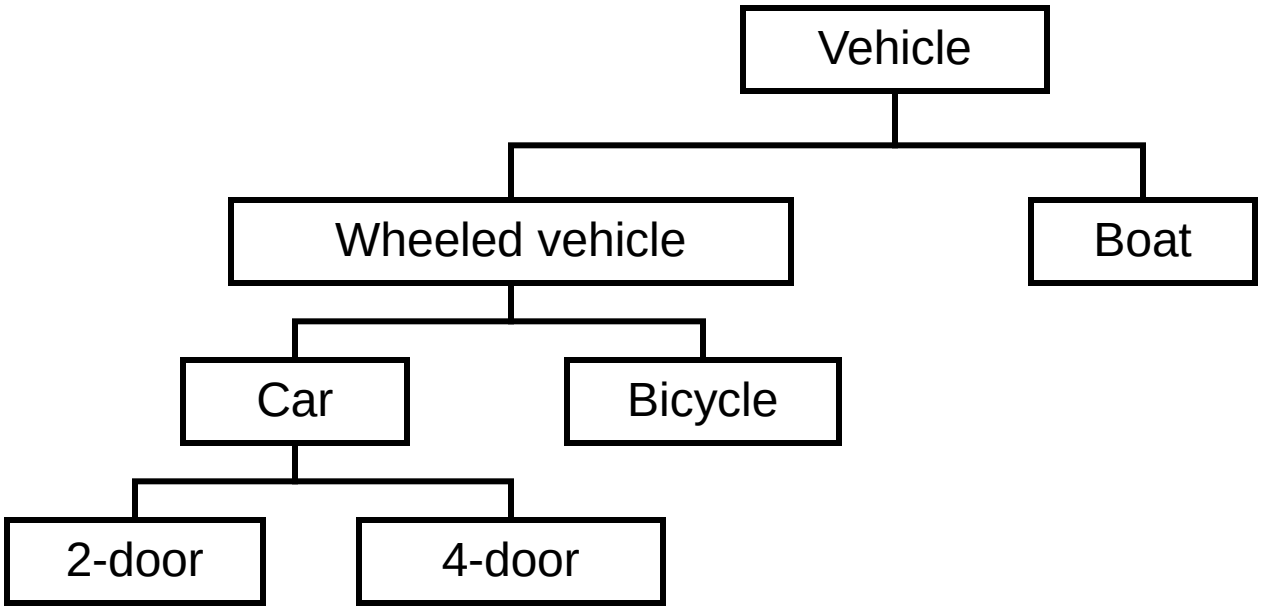
Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Relationship between classes

- In object oriented programming there are two types of relationships between classes which are commonly used
  - Is-a relationship:
    - The *is-a* relationship represents inheritance. In an *is-a* relationship, an object of a derived class also can be treated as an object of its base class e.g, a Car *is a* Vehicle, so any attributes and behaviors of a Vehicle are also attributes and behaviors of a Car
  - Has-a relationship:
    - In a *has-a* relationship, an object *contains* one or more objects of other classes as members. e.g, a Car has many components—it *has a* steering wheel, *has a* brake pedal, *has a* transmission, etc

Arrange concepts into an inheritance hierarchy

- Concepts at higher levels are more general
- Concepts at lower levels are more specific (inherit properties of concepts at higher levels)



Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Protected Members and Use in Derived Class

- Protected members of a base class are like private members, but they may be accessed by derived classes. The base class access specification determines how private, protected, and public base class members may be accessed by derived classes.

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Inheritance

Subgroupings with respect to a parent are called

- Subclass
- Derived Class
- Children

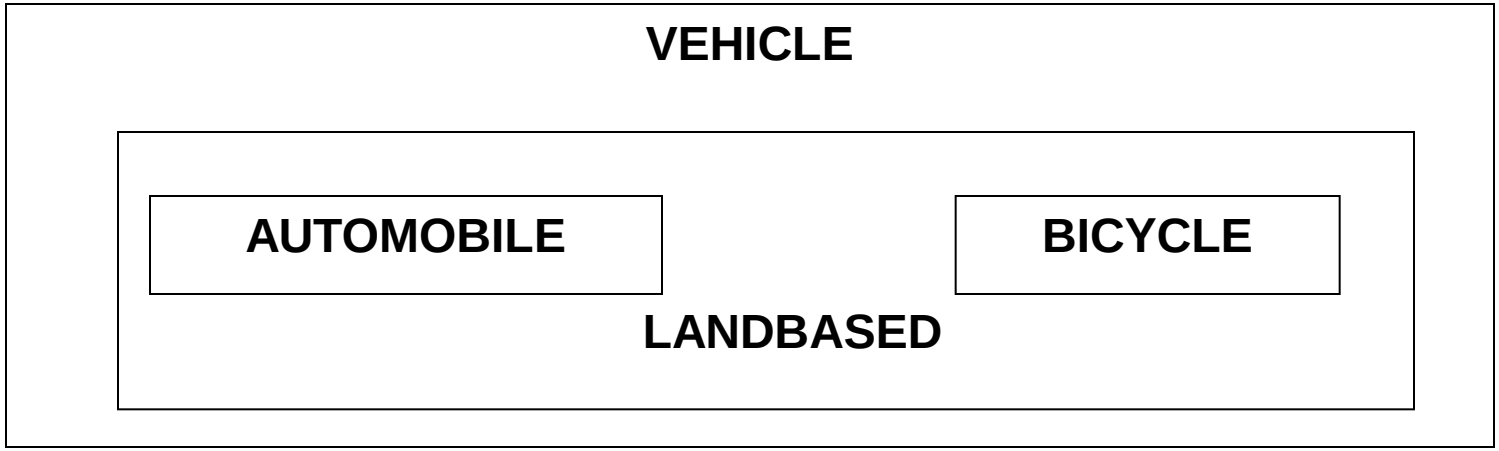
The derived class

- inherits from the parent all
  - characteristics
  - properties
  - capabilities
- can modify or extend inherited abilities

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Inheritance

- Study the vehicle hierarchy:



- Note that each entity is also a class
- Classes are used in an object centered paradigm as a means of encapsulating common data and functions shared by members of the class

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Types of Inheritance according to Number of base classes

Single Inheritance

Each class or instance object has a single parent

Multiple Inheritance

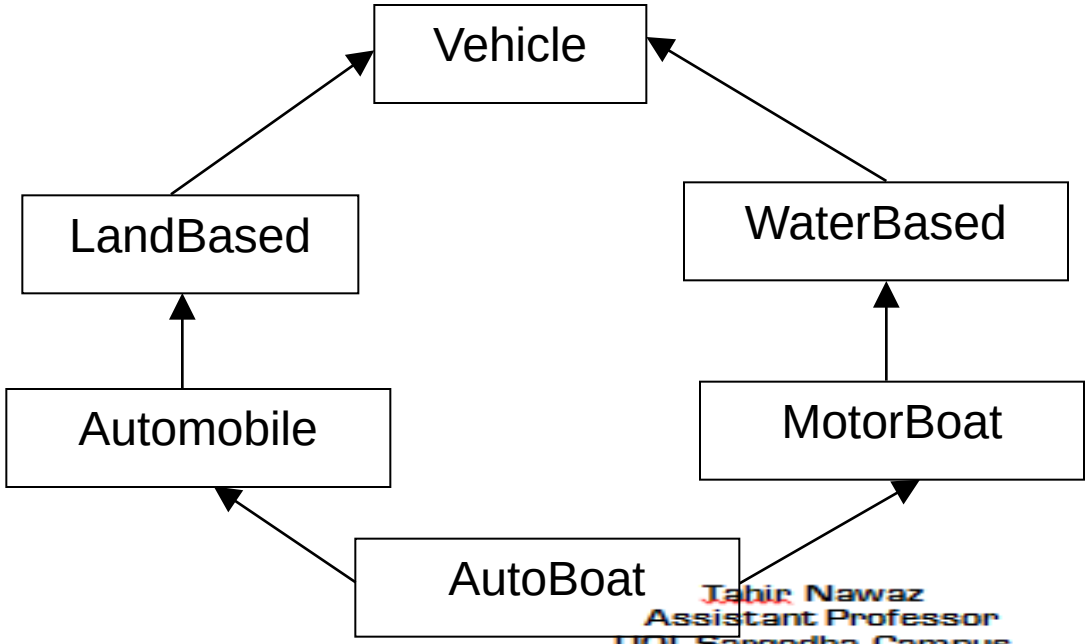
Classes inherit from multiple base classes (might not have same ancestors as shown in the example below)

Defines a relationship

Between several (independent) class types

Example:

Multiple Parents Common Ancestor



Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Types of Inheritance According to Accessibility

- A class(Derived Class) can be derived from another class(Base Class) by using access specifiers Private, Protected and Public with the name of base class. On the bases of using these access specifiers inheritance is divided into three types
  - Private Inheritance
  - Protected Inheritance
  - Public Inheritance

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

Private inheritance

- In this type of inheritance access specifiers private is used with the name of base class during declaration of derived class e.g.
  - class child : private parent

Protected inheritance

- In this type of inheritance access specifiers protected is used with the name of base class during declaration of derived class e.g.
  - Class child : protected parent

Public inheritance

- Public inheritance using public access specifier with the name of the base class during the declaration of derived classes.
  - class child : public parent
- A base class's private members are accessible only within its body and to the friends of that base class. In this section, we introduce the access specifier protected.
- Using protected access offers an intermediate level of protection between public and private access. A base class's protected members can be accessed within the body of that base class, by members and friends of that base class, and by members and friends of any classes derived from that base class

Inherited Member Initialization

- Initialized in two ways:
  - If the base class has only a default constructor=> initialize the member values in the body of the derived class constructor
  - If the base class has a constructor with arguments=> the initialization list is used to pass arguments to the base class constructors

syntax (for Single Base Class)

```
DerivedClass ( derivedClass args ) : BaseClass ( baseClass args )        {
        DerivedClass constructor body
        }
```

- The set of derived class constructor arguments may contain initialization values for the base class arguments

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

e.g.1

```
#include<iostream.h>
#include<conio.h>
 using namespace std;
 class A {
 int data;
public:
void f(int arg) {
data = arg;
 }
int g() {
 return data;
} };
```

```
class B : public A { };
void main() {
B obj;
obj.f(20);
cout << obj.g() << endl;
getch() }
```

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

e.g; 2

```
#include <iostream.h>
using namespace std;
// Base class
class Shape {
protected:
int width; int height;
 public:
 void setWidth(int w) {
width = w;
 }
void setHeight(int h) {
 height = h;
}
};
```

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

(Cont)

```
// Derived class
 class Rectangle: public Shape {
 public:
int getArea() {
 return (width * height);
 } };
void main(void) {
 Rectangle Rect;
Rect.setWidth(5);
Rect.setHeight(7);
// Print the area of the object
cout << "Total area: " << Rect.getArea() << endl;
return 0; }
```

Assignment #3

- Q1. Write a program to implement the inheritance process of following defined classes.
  - A base class named calculate which have data members a, b, c, d, e and %age using a member function.
  - Find the percentage of the five subjects marks.
- Q2. write a program to implement the inheritance process of following defined classes
  - A base class named as shape
  - A derived class named triangle. Data members and member functions must not b defined in both classes.
  - Find the area of triangle by using the derived classes.

Submission Date: 11/2/2014.