BAKI ALİ NEFT MƏKTƏBİ
BAKU HIGHER OIL SCHOOL

Programming and Computer Applications

Classes
A Deeper Look

Instructor : PhD, Associate Professor Leyla Muradkhanli

friend Functions and friend Classes

- A **friend** function of a class is defined outside that class's scope, yet has the right to access the non-public (and public) members of the class.
- Standalone functions, entire classes or member functions of other classes may be declared to be friends of another class.
- Using friend functions can enhance performance.
- Friendship is granted, not taken.
- The friendship relation is neither symmetric nor transitive.

```cpp
#include <iostream>
using namespace std;
class Myclass
{
    private:
        int x;
    public:
        Myclass(): x(0) { }
           //friend function
        friend int addFive(Myclass);
};
// friend function definition
int addFive(Myclass a)
{
    //accessing private data from non-member function
    a.x += 5;
     return a.x;
}
```

```
int main()
{
    Myclass A;
    cout<<"Myclass: "<< addFive(A)<<endl;

return 0;
}
```

```cpp
#include <iostream>
using namespace std;
class B;
class A {
    private:
      int numA;
    public:
      A(): numA(12) { }
      // friend function declaration
      friend int add(A, B);
};
class B {
    private:
       int numB;
    public:
       B(): numB(1) { }
       // friend function declaration
       friend int add(A , B);
};
```

```
// Function add() is the friend function of classes A and B
// that accesses the member variables numA and numB
int add(A objectA, B objectB)
{
   return (objectA.numA + objectB.numB);
}
int main()
{
    A objA;
    B objB;
    cout<<"Sum: "<< add(objA, objB)<<endl;
return 0;
}
```

`friend` Functions and `friend` Classes

Similarly, like a friend function, a class can also be made a friend of another class using keyword friend. For example:

```
class B;
class A
{
   // class B is a friend class of class A
   friend class B;
}
class B
{
   ... .. ...
}
```

friend Functions and friend Classes

When a class is made a friend class, all the member functions of that class becomes friend functions.

In this program, all member functions of class B will be friend functions of class A. Thus, any member function of class B can access the private and protected data of class A. But, member functions of class A cannot access the data of class B.

Remember, friend relation in C++ is only granted, not taken.

```cpp
#include <iostream>
using namespace std;
class A {
private:
    int a;
public:
    A() { a = 0; }
    friend class B; // Friend Class
};
class B {
private:
    int b;
public:
    void show(A& x)
    {
        // Since B is friend of A, it can access
        // private members of A
        cout << "a=" << x.a<<endl;
    }
};
```

```
int main()
{
    A a;
    B b;
    b.show(a);

    return 0;
}
```

```
#include <iostream>
using namespace std;
class MyClass
{
// Declare a friend class
friend class YourClass;
public:
MyClass() : Secret(0){}
void print()
{
cout << Secret << endl;
}
private:
int Secret;
};
```

```
class YourClass
{
public:
void change( MyClass& yourclass, int x )
{
yourclass.Secret = x;
}
};
int main()
{
MyClass a;
YourClass b;
a.print();
b.change(a, 5 );
a.print();
}
```