

# - Examples

In this lesson, we will take a look at different examples of friends.

## WE'LL COVER THE FOLLOWING ^

- Example 1
  - Explanation
- Example 2
  - Explanation
- Example 3
  - Explanation

## Example 1 #

```
// templateClassTemplateGeneralFriendship.cpp

#include <iostream>

template <typename T> void myFriendFunction(T);
template <typename U> class MyFriend;

class GrantingFriendshipAsClass{

    template <typename U> friend void myFriendFunction(U);
    template <typename U> friend class MyFriend;

private:
    std::string secret{"My secret from GrantingFriendshipAsClass."};

};

template <typename T>
class GrantingFriendshipAsClassTemplate{

    template <typename U> friend void myFriendFunction(U);
    template <typename U> friend class MyFriend;

private:
    std::string secret{"My secret from GrantingFriendshipAsClassTemplate."};
```

```

};

template <typename T>
void myFriendFunction(T){
    GrantingFriendshipAsClass myFriend;
    std::cout << myFriend.secret << std::endl;

    GrantingFriendshipAsClassTemplate<double> myFriend1;
    std::cout << myFriend1.secret << std::endl;
}

template <typename T>
class MyFriend{
public:
    MyFriend(){
        GrantingFriendshipAsClass myFriend;
        std::cout << myFriend.secret << std::endl;

        GrantingFriendshipAsClassTemplate<T> myFriend1;
        std::cout << myFriend1.secret << std::endl;
    }
};

int main(){

    std::cout << std::endl;

    int a{2011};
    myFriendFunction(a);

    MyFriend<double> myFriend;

    std::cout << std::endl;

}

```



## Explanation #

- In the above example, we created a function `myFriendFunction` and a class `MyFriend`. We have defined two classes: `GrantingFriendshipAsClass` and `GrantingFriendshipAsClassTemplate`.
- As the name mentions, we use one class with template and one without a template. The class `MyFriend` and the function `myFriendFunction` have access to the private members of the other classes by using a `friend` keyword.
- We defined a `private` variable `secret` that is of a string type and can be called with the object of `myFriendFunction` and `MyFriend`.

## Example 2 #

```
#include <iostream>

template <typename T> void myFriendFunction(T);
template <typename U> class MyFriend;

class GrantingFriendshipAsClass{

    friend void myFriendFunction<>(int);
    friend class MyFriend<int>;

private:
    std::string secret{"My secret from GrantingFriendshipAsClass."};

};

template <typename T>
class GrantingFriendshipAsClassTemplate{

    friend void myFriendFunction<>(int);
    friend class MyFriend<int>;
    friend class MyFriend<T>;

private:
    std::string secret{"My secret from GrantingFriendshipAsClassTemplate."};

};

template <typename T>
void myFriendFunction(T){
    GrantingFriendshipAsClass myFriend;
    std::cout << myFriend.secret << std::endl;

    GrantingFriendshipAsClassTemplate<T> myFriend1;
    std::cout << myFriend1.secret << std::endl;
}

template <typename T>
class MyFriend{
public:
    MyFriend(){
        GrantingFriendshipAsClass myFriend;
        std::cout << myFriend.secret << std::endl;

        GrantingFriendshipAsClassTemplate<int> myFriendInt;
        std::cout << myFriendInt.secret << std::endl;

        GrantingFriendshipAsClassTemplate<T> myFriendT;
        std::cout << myFriendT.secret << std::endl;
    }
};

int main(){

    std::cout << std::endl;

    int a{2011};
    myFriendFunction(a);
```

```
MyFriend<int> myFriend;

std::cout << std::endl;

}
```



## Explanation #

As we saw in example 1, similarly with the addition of explicitly stating the type of class template to `int`. The class template is called both for `int` and for any other type mentioned in the typename portion.

## Example 3 #

```
#include <iostream>

template <typename T>
class Bank{
    std::string secret{"Import secret from the bank."};
    friend T;
};

class Account{
public:
    Account(){
        Bank<Account> bank;
        std::cout << bank.secret << std::endl;
    }
};

int main(){

    std::cout << std::endl;

    Account acc;

    std::cout << std::endl;

}
```



## Explanation #

In the above code, we created an `Account` class that contains the `Bank` class object. We can access the `Bank` class member `secret` with the help of `friend`

object. We can access the `Bank` class member `secret` with the help of `friend`.  
Now, the value stored in `secret` is accessible in the `Account` class.

---

In the next lesson, we will learn about template parameters.