

- Examples

In this lesson, we'll look at a few examples of using templates with friends.

WE'LL COVER THE FOLLOWING ^

- Example 1: Class Template General Friendship
 - Explanation
- Example 2: Class Template Special Friendship
 - Explanation
- Example 3: Class Template Type Friendship
 - Explanation

Example 1: Class Template General Friendship

```
// 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 have created a function `myFriendFunction` and a class `MyFriend`. We have defined two classes: `GrantingFriendshipAsClass` and `GrantingFriendshipAsClassTemplate`. As the name mentioned as well, we are using 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 have defined a `private` variable `secret` which is of a string type and can be called with the object of `myFriendFunction` and `MyFriend`.

Example 2: Class Template Special Friendship



```
// templateClassTemplateSpecialFriendship.cpp

#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

This example is similar to example 1 with a small change; we have explicitly stated the type of class template to `int`. Now, the class template is called for `int` and also for any other type mentioned in the `typename` portion.

Example 3: Class Template Type Friendship

```
// templateClassTemplateTypeFriendship.cpp

#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 have created an `Account` class which contains the `Bank` class object. We can access the `Bank` class member `secret` with the help of

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

We'll solve an exercise in the next lesson.