

Hello,

I have 2 class :

Object.h

```
#include "ComponentManager.h"
class Object{
public:
    Object() : componentManager(*this){}
    /*
        Many public things
    */
private:
    ComponentManager componentManager;
};
```

ComponentManager.h

```
class Object;

class ComponentManager{
public:
    ComponentManager(Object& _object) : object(_object){}

    template<class T> T& function1(...)
    /*
        Many templated function that force me to write declaration in this .h file
    */
private:
    Object& object;
}
```

My both class are made to work together, ComponentManager make no sens without an object etc...

All is fine and work well. Untill, in one of my templated function in ComponentManager, I need to use my Object reference to call a function of object :

```
template <class T> T& CreateComponent(bool active, int position){
/*
    Working code
```

```

*/
Object& obj = object.GetScene().GetObjectManager().GetObjects()[i]; //This line is problematic
/*
    Working code
*/
}

```

since I need to use some function of Object, compiler need Object declaration. But no way I can give to ComponentManager the implementation of object because Object need Component implementation. I can't split my ComponentManager into .h and .cpp so I'm kind of blocked.

Someone have an idea of how I could trick the compiler ? do I will need to change the way my architecture work (only for a function :()).

here is the complete problematic chunk of code :

```

for(int i = 0; i < object.GetScene().GetObjectManager().GetObjectsCount(); i++){
    Object& obj = object.GetScene().GetObjectManager().GetObjects()[i];
    if(&obj != &object){
        if(obj.GetComponentManager().HasComponent<T>()){
            throw Exc("...");
        }
    }
}
}

```

I have thinked about externalise in a .cpp a function (not templated) which do the job and then I could have include in the .cpp the Object definition but since I need to call some Function with my templated class argument, I dont think I can work at all.