**Singleton**

**Problem**

Multiple instances of a class may not make sense or may create confusion.

**Solution**

Hide all constructors. Provide a static factory method that returns a pointer to the single instance of the class.

**Structure**

**Behavior**

**Discussion**

Assume an instance of the Manager class will be responsible for managing all components of an application. Having multiple managers could create confusion, so we hide all of the constructors. Java and C++ classes have a public implicit default constructor (i.e., a parameterless constructor) that must be made explicit and private. C++ classes also have an implicit copy constructor that must be made public.

class Manager  
{  
private:  
�� static Manager\* theManager;  
�� Manager() { ... }  
�� Manager(const Manager& m) { }  
�� ~Manager() { }  
public:  
�� // factory method:  
�� static Manager\* makeManager()  
�� {  
����� if (!theManager) { theManager = new Manager(); }  
����� return theManager;  
�� }  
�� // etc.  
};

Obviously the factory method and the pointer must be made static. In C++ memory must be allocated and initialized for the pointer separately:

Manager\* Manager::theManager = 0;

Here is some sample client code:

Manager\* m1 = Manager::makeManager();  
Manager\* m2 = Manager::makeManager(); // m1 == m2

delete m1; // error, destructor can't be called here  
m1 = new Manager(); // error, constructor can't be called here

Assume the following function is declared:

void display(Manager m) { ... }

However calling the function isn't allowed:

display(\*m1); // error, copy constructor can't be called here

Managers must be passed by reference:

void display(Manager& m) { ... }