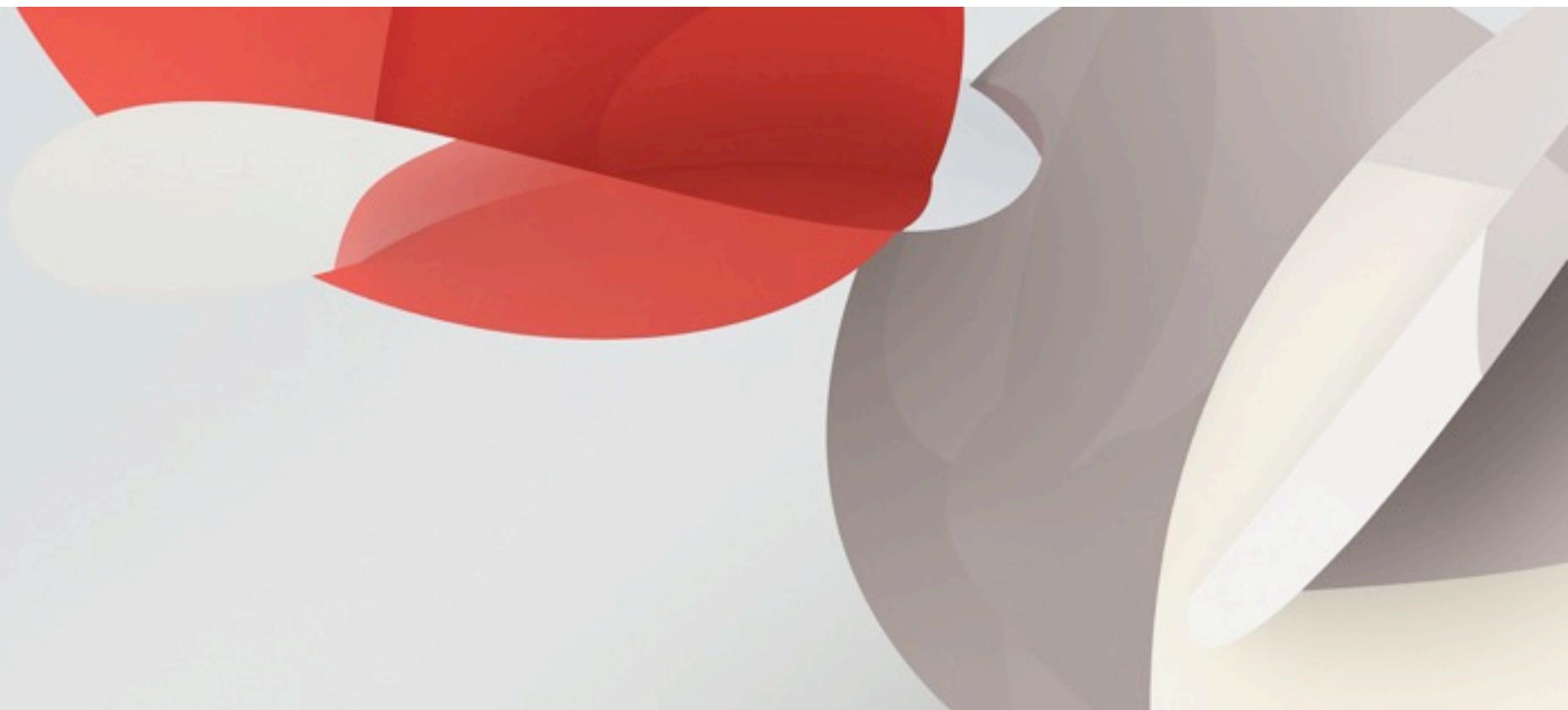


Inheritance Is The Base Class of Evil

Sean Parent | Principal Scientist



Recap from Seasoning Talk

- Goal: No Raw Pointers
- Indirection through pointers
 - Changes semantics of copy, assignment, and equality
 - Incidental data structures
 - Thread safety concerns
 - Inefficient
 - Shared pointer as good as a global variable

Another Deep problem

- Inheritance is intrusive

"Polymorphic Types"

- The requirement of a polymorphic type, by definition, comes from its use
- There are no polymorphic types, only a *polymorphic use* of similar types

"Polymorphic Types"

- By using inheritance to capture polymorphic use, we shift the burden of use to the type implementation, tightly coupling components
- Inheritance implies variable size, which implies heap allocation
- Heap allocation forces a further burden to manage the object lifetime
- Indirection, heap allocation, virtualization impacts performance
- Object lifetime management leads to garbage collection or reference counting
- This encourages *shared* ownership and the proliferation of *incidental data-structures*
- Shared ownership leads to synchronization issues, breaks local reasoning, and further impacts performance

Inheritance Is The Base Class of Evil

client

library

```
using object_t = int;

void draw(const object_t& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library

```
using object_t = int;

void draw(const object_t& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library

```
void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(const int& x) : self_(x)
    {}

    friend void draw(const object_t& x, ostream& out, size_t position)
    { draw(x.self_, out, position); }

private:
    int self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library

```
int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(1);
    document.emplace_back(2);
    document.emplace_back(3);

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(const int& x) : self_(x)
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { draw(x.self_, out, position); }

private:
    int self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(const int& x) : self_(x)
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { draw(x.self_, out, position); }

private:
    int self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library

+

```
class object_t {
public:
    object_t(const int& x) : self_(x)
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { draw(x.self_, out, position); }

private:
    int self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library



```
class object_t {  
public:  
    object_t(const int& x) : self_(x)  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { draw(x.self_, out, position); }  
  
private:  
    int self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;  
}
```

cout

guidelines

defects

client

library



```
class object_t {  
public:  
    object_t(const int& x) : self_(x)  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { draw(x.self_, out, position); }  
  
private:  
    int self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;  
}
```

cout

guidelines

defects

client

library



```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };  
  
    unique_ptr<int_model_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;
```



cout

guidelines

defects

client

library

```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    {}  
};
```

```
object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
{ }
```

```
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }
```

```
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```

```
    unique_ptr<int_model_t> self_;
```

```
};
```

```
using document_t = vector<object_t>;
```



cout

guidelines

defects



client

library



```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    { }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { }  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); self_ = move(tmp.self_); return *this; }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };  
    unique_ptr<int_model_t> self_;  
};
```



cout

guidelines

defects

client

library

```
int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(1);
    document.emplace_back(2);
    document.emplace_back(3);

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library



```
class object_t {
public:
    object_t(const int& x) : self_(new int_model_t(x))
    { }

    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    { }

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct int_model_t {
        int_model_t(const int& x) : data_(x) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };

    unique_ptr<int_model_t> self_;
};
```



cout

guidelines

defects

client

library



```
class object_t {
public:
    object_t(const int& x) : self_(new int_model_t(x))
    { cout << "ctor" << endl; }

    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    { cout << "copy" << endl; }
    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct int_model_t {
        int_model_t(const int& x) : data_(x) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };

    unique_ptr<int_model_t> self_;
};
```



cout

guidelines

defects

client

library

```
object_t func()
{
    object_t result = 5;
    return result;
}

int main()
{
    /*
        Quiz: What will this print?
    */

    object_t x = func();
}
```

cout

guidelines

defects

client

library

```
object_t func()
{
    object_t result = 5;
    return result;
}

int main()
{
    /*
        Quiz: What will this print?
    */
}
```

```
object_t x = 0;

x = func();
```

cout

guidelines

defects

client

library



```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    { cout << "ctor" << endl; }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { cout << "copy" << endl; }  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); self_ = move(tmp.self_); return *this; }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };  
    unique_ptr<int_model_t> self_;  
};
```



cout

guidelines

defects

client

library

```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    { cout << "ctor" << endl; }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { cout << "copy" << endl; }  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };  
  
    unique_ptr<int_model_t> self_;
```



cout

guidelines

defects



client

library

```
object_t func()
{
    object_t result = 5;
    return result;
}

int main()
{
    /*
        Quiz: What will this print?
    */

    object_t x = 0;
    x = func();
}
```

cout

guidelines

defects

client

library

```
int main()
{
    document_t document;
    document.reserve(5);

    document.emplace_back(0);
    document.emplace_back(1);
    document.emplace_back(2);
    document.emplace_back(3);

    reverse(document.begin(), document.end());

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    { cout << "ctor" << endl; }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { cout << "copy" << endl; }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```

+

cout

guidelines

defects

client

library

```
int main()
{
    document_t document;
    document.reserve(5);

    document.emplace_back(0);
    document.emplace_back(1);
    document.emplace_back(2);
    document.emplace_back(3);

    reverse(document.begin(), document.end());

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    { cout << "ctor" << endl; }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { cout << "copy" << endl; }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```



cout

guidelines

defects



client

library

```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    {}  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    {}  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```

+

cout

guidelines

defects

client

library



```
class object_t {
public:
    object_t(const int& x) : self_(new int_model_t(x))
    { }

    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct int_model_t {
        int_model_t(const int& x) : data_(x) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
};
```



cout

guidelines

defects

client

library



```
class object_t {  
public:  
    object_t(const int& x) : self_(new int_model_t(x))  
    { }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(const int& x) : data_(x) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```



cout

guidelines

defects

client

library

Pass sink arg by value and swap or move into place

```
class object_t {  
public:  
    object_t(int x) : self_(new int_model_t(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct int_model_t {  
        int_model_t(int x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```

A sink arg is any arg consumed or returned by func
- arg to assignment
- define move= for it

cout

guidelines

defects

client

library



```
class object_t {
public:
    object_t(int x) : self_(new int_model_t(move(x)))
    { }

    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
};
```



cout

guidelines

defects

client

library

+

```
public:  
    object_t(string x) : self_(new string_model_t(move(x)))  
    { }  
    object_t(int x) : self_(new int_model_t(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }
```

private:

```
    struct string_model_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };
```

+

cout

guidelines

defects

client

library

```
public:  
    object_t(string x) : self_(new string_model_t(move(x)))  
    { }  
    object_t(int x) : self_(new int_model_t(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct string_model_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };
```

cout

guidelines

defects

client

library

```
object_t(string x) : self_(new string_model_t(move(x)))
{ }
object_t(int x) : self_(new int_model_t(move(x)))
{ }

object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
```

cout

guidelines

defects

client

library

```
{ }  
object_t(int x) : self_(new int_model_t(move(x)))  
{ }  
  
object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
{ }  
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct string_model_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t {  
        int_model_t(int x) : data_(move(x)) { }  
    };
```



cout

guidelines

defects

client

library

```
object_t(int x) : self_(new int_model_t(move(x)))
{ }

object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { }
```

cout

guidelines

defects

client

library

{ }

+

```
object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

+

cout

guidelines

defects

client

library

+

```
object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

+

cout

guidelines

defects

client

library

```
object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };

```



cout

guidelines

defects

client

library



```
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
};
```



cout

guidelines

defects

client

library

```
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct string_model_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t {  
        int_model_t(int x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```

+

cout

guidelines

defects

client

library



```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };

    unique_ptr<int_model_t> self_;
```



cout

guidelines

defects

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
    unique_ptr<int_model_t> self_;
};
```

cout

guidelines

defects

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
    unique_ptr<int_model_t> self_;
};
```

cout

guidelines

defects

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
    unique_ptr<concept_t> self_;
};
```

cout

guidelines

defects

client**library**

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct string_model_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
    unique_ptr<concept_t> self_;
};
```

**cout****guidelines****defects**

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
};
```



cout

guidelines

defects

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
}
```

+

cout

guidelines

defects

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
}
```

cout

guidelines

defects

client

library

+

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

+

cout

guidelines

defects

client

library

```
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
    };
```



cout

guidelines

defects

client

library

```
{ }  
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const
```



cout

guidelines

defects

client

library

```
object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
    };

```



cout

guidelines

defects

client

library



```
object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
```



cout

guidelines

defects

client

library

```
{ }

object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
};
```



cout

guidelines

defects



client

library

```
object_t(int x) : self_(new int_model_t(move(x)))
{ }

object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
}
```

+

cout

guidelines

defects

client

library

```
{ }  
object_t(int x) : self_(new int_model_t(move(x)))  
{ }  
  
object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
{ }  
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
    };
```



cout

guidelines

defects



client

library

```
object_t(string x) : self_(new string_model_t(move(x)))
{ }
object_t(int x) : self_(new int_model_t(move(x)))
{ }

object_t(const object_t& x) : self_(new int_model_t(*x.self_))
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

cout

guidelines

defects

```
public:  
    object_t(string x) : self_(new string_model_t(move(x)))  
    { }  
    object_t(int x) : self_(new int_model_t(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
    };
```

client

library

```
class object_t {  
public:  
    object_t(string x) : self_(new string_model_t(move(x)))  
    {}  
    object_t(int x) : self_(new int_model_t(move(x)))  
    {}  
  
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))  
    {}  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) {}  
    };
```



cout

guidelines

defects



client

library



```
class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    { }
    object_t(int x) : self_(new int_model_t(move(x)))
    { }
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {

```



cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    { }
    object_t(int x) : self_(new int_model_t(move(x)))
    { }
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
}
```



cout

guidelines

defects

client

library

```
void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    {}
    object_t(int x) : self_(new int_model_t(move(x)))
    {}
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    {}
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
```



cout

guidelines

defects

client

library

```
void draw(const string& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    {}
    object_t(int x) : self_(new int_model_t(move(x)))
    {}
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    {}
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
```



cout

guidelines

defects

client

library

```
void draw(const string& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    {}
    object_t(int x) : self_(new int_model_t(move(x)))
    {}
    object_t(const object_t& x) : self_(new int_model_t(*x.self_))
    {}
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
```



cout

guidelines

defects

client

library

```
void draw(const string& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    {}
    object_t(int x) : self_(new int_model_t(move(x)))
    {}
    object_t(const object_t& x) : self_(x.self_->copy_())
    {}
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
```



cout

guidelines

defects

client

library

```
void draw(const string& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    {}
    object_t(int x) : self_(new int_model_t(move(x)))
    {}
    object_t(const object_t& x) : self_(x.self_->copy_())
    {}
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
```



cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << x << endl; }

void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    {}
    object_t(int x) : self_(new int_model_t(move(x)))
    {}
    object_t(const object_t& x) : self_(x.self_->copy_())
    {}
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {

```

cout

guidelines

defects

client

library



```
void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    { }
    object_t(int x) : self_(new int_model_t(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }
```

private:

```
struct concept_t {
    virtual ~concept_t() = default;
```



cout

guidelines

defects

client

library

```
void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    {}
    object_t(int x) : self_(new int_model_t(move(x)))
    {}
    object_t(const object_t& x) : self_(x.self_->copy_())
    {}
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
```

cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << x << endl; }
```

```
class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    { }
    object_t(int x) : self_(new int_model_t(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
};
```



cout

guidelines

defects

client

library



```
class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    { }
    object_t(int x) : self_(new int_model_t(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {

```



cout

guidelines

defects

client

library

```
class object_t {  
public:  
    object_t(string x) : self_(new string_model_t(move(x)))  
    {}  
    object_t(int x) : self_(new int_model_t(move(x)))  
    {}  
  
    object_t(const object_t& x) : self_(x.self_->copy_())  
    {}  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) {}  
    };
```



cout

guidelines

defects



```
public:  
    object_t(string x) : self_(new string_model_t(move(x)))  
    { }  
    object_t(int x) : self_(new int_model_t(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(x.self_->copy_())  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
    };
```

client

library

```
object_t(string x) : self_(new string_model_t(move(x)))
{ }
object_t(int x) : self_(new int_model_t(move(x)))
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

cout

guidelines

defects

client

library

```
{ }  
object_t(int x) : self_(new int_model_t(move(x)))  
{ }  
  
object_t(const object_t& x) : self_(x.self_->copy_())  
{ }  
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
    };
```



cout

guidelines

defects



client

library

```
object_t(int x) : self_(new int_model_t(move(x)))
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
}
```

+

cout

guidelines

defects

client

library



```
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
}
```



cout

guidelines

defects

client

library

+

```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
```

+

cout

guidelines

defects

client

library

```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
    };

```

defects

client

library

```
{ }  
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const
```



cout

guidelines

defects

client

library

```
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
    };
```



cout

guidelines

defects

client

library

+

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

+

cout

guidelines

defects

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
}
```

cout

guidelines

defects

client

library

```
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```



cout

guidelines

defects

client

library

```
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```

+

cout

guidelines

defects

client

library



```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };

unique_ptr<concept_t> self_;
```



cout

guidelines

defects

client

library

```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
    unique_ptr<concept_t> self_;
};
```

cout

guidelines

defects

```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }
```

private:

```
struct concept_t {
    virtual ~concept_t() = default;
    virtual concept_t* copy_() const = 0;
    virtual void draw_(ostream&, size_t) const = 0;
};
```

```
struct string_model_t : concept_t {
    string_model_t(string x) : data_(move(x)) { }
    concept_t* copy_() const { return new string_model_t(*this); }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    string data_;
};
```

```
struct int_model_t : concept_t {
    int_model_t(int x) : data_(move(x)) { }
    concept_t* copy_() const { return new int_model_t(*this); }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    int data_;
};
```



client

library

```
int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(1);
    document.emplace_back(2);
    document.emplace_back(3);

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(string("Hello!"));
    document.emplace_back(2);
    document.emplace_back(3);

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
void draw(const string& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

void draw(const int& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    object_t(string x) : self_(new string_model_t(move(x)))
    { }
    object_t(int x) : self_(new int_model_t(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
```



cout

guidelines

defects

```
template <typename T>
void draw(const T& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }
```

```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }
```

```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }
```

```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
```



```
template <typename T>
void draw(const T& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
```



client

library

```
void draw(const T& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
}
```

cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << x << endl; }
```

```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
}
```



cout

guidelines

defects

client

library

+

```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {

```

+

cout

guidelines

defects

client

library

```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(x.self_->copy_())  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
    };
```



cout

guidelines

defects



client

library

```
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(x.self_->copy_())  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        concept_t* copy_() const { return new string_model_t(*this); }  
    };
```

cout

guidelines

defects

client

library

```
template <typename T>
object_t(T x) : self_(new model<T>(move(x)))
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
    };

```

cout

guidelines

defects

client

library

```
object_t(T x) : self_(new model<T>(+move(x)))
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

+

cout

guidelines

defects

client

library



```
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```



cout

guidelines

defects

client

library



```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
}
```



cout

guidelines

defects

client

library

```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
};
```



cout

guidelines

defects

client

library

```
{ }  
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        concept_t* copy_() const { return new string_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {
```



cout

guidelines

defects



client

library

```
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        concept_t* copy_() const { return new string_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) { }  
    };
```



cout

guidelines

defects

client

library



```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        concept_t* copy_() const { return new int_model_t(*this); }
    };
}
```



cout

guidelines

defects

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        concept_t* copy_() const { return new int_model_t(*this); }
        void draw_(ostream& out, size_t position) const
    };

```



client

library

```
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        concept_t* copy_() const { return new string_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) { }  
        concept_t* copy_() const { return new int_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
    };
```

cout

guidelines

defects

client

library

```
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) {}  
        concept_t* copy_() const { return new string_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) {}  
        concept_t* copy_() const { return new int_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
    };
```



cout

guidelines

defects

client

library



```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        concept_t* copy_() const { return new int_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
}
```



cout

guidelines

defects

client

library

```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        concept_t* copy_() const { return new int_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };

```

+

cout

guidelines

defects

client

library

```
{ x.self_->draw_(out, position); } +  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        concept_t* copy_() const { return new string_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) { }  
        concept_t* copy_() const { return new int_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };
```

+

cout

guidelines

defects

client

library



```
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    struct string_model_t : concept_t {  
        string_model_t(string x) : data_(move(x)) { }  
        concept_t* copy_() const { return new string_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        string data_;  
    };  
    struct int_model_t : concept_t {  
        int_model_t(int x) : data_(move(x)) { }  
        concept_t* copy_() const { return new int_model_t(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        int data_;  
    };  
  
    unique_ptr<concept_t> self_;
```



cout

guidelines

defects

client

library



```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        concept_t* copy_() const { return new int_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
    unique_ptr<concept_t> self_;
};
```



cout

guidelines

defects

client

library

+

```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    struct string_model_t : concept_t {
        string_model_t(string x) : data_(move(x)) { }
        concept_t* copy_() const { return new string_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        string data_;
    };
    struct int_model_t : concept_t {
        int_model_t(int x) : data_(move(x)) { }
        concept_t* copy_() const { return new int_model_t(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        int data_;
    };
    unique_ptr<concept_t> self_;
};
```

+

cout

guidelines

defects

client

library



```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    concept_t* copy_() const { return new model(*this); }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
}
```



cout

guidelines

defects

client

library

```
class my_class_t {  
    /* ... */  
};  
  
void draw(const my_class_t&, ostream& out, size_t position)  
{ out << string(position, ' ') << "my_class_t" << endl; }  
  
int main()  
{  
    document_t document;  
  
    document.emplace_back(0);  
    document.emplace_back(string("Hello!"));  
    document.emplace_back(2);  
    document.emplace_back(my_class_t());  
  
    draw(document, cout, 0);  
}
```

cout

guidelines

defects

client

library

```
class my_class_t {  
    /* ... */  
};  
  
void draw(const my_class_t&, ostream& out, size_t position)  
{ out << string(position, ' ') << "my_class_t" << endl; }  
  
int main()  
{  
    document_t document;  
  
    document.emplace_back(0);  
    document.emplace_back(string("Hello!"));  
    document.emplace_back(document);  
    document.emplace_back(my_class_t());  
  
    draw(document, cout, 0);  
}
```

cout

guidelines

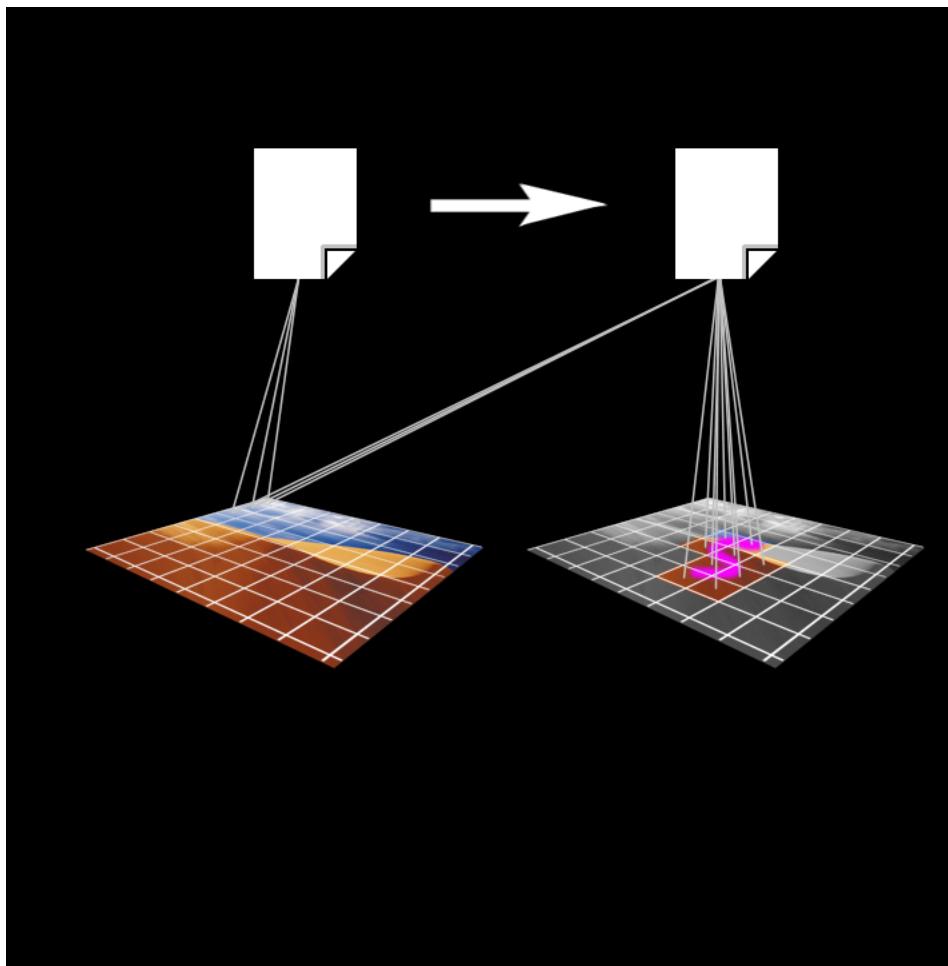
defects

Polymorphic Use

- Shifting polymorphism from type to use allows for greater reuse and fewer dependencies
- Using regular semantics for the common basis operations, copy, assignment, and move helps to reduce shared objects
- Regular types promote interoperability of software components, increases productivity as well as quality, security, and performance
- There is no performance penalty to using regular semantics, and often times there are performance benefits from a decreased use of the heap

Photoshop History

Photoshop History



client

library

```
concept_t* copy_() const { return new model(*this); }
void draw_(ostream& out, size_t position) const
{ draw(data_, out, position); }

T data_;
};

unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library

```
concept_t* copy_() const { return new model(*this); }
void draw_(ostream& out, size_t position) const
{ draw(data_, out, position); }

T data_;
};

unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

```
using history_t = vector<document_t>

void commit(history_t& x) { assert(x.size()); x.push_back(x.back()); }
void undo(history_t& x) { assert(x.size()); x.pop_back(); }
document_t& current(history_t& x) { assert(x.size()); return x.back(); }
```

cout

guidelines

defects

client

library

```
model(T x) : data_(move(x)) { }  
concept_t* copy_() const { return new model(*this); }  
void draw_(ostream& out, size_t position) const  
{ draw(data_, out, position); }  
  
T data_;  
};  
  
unique_ptr<concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;  
}  
  
using history_t = vector<document_t>;  
  
void commit(history_t& x) { assert(x.size()); x.push_back(x.back()); }  
void undo(history_t& x) { assert(x.size()); x.pop_back(); }  
document_t& current(history_t& x) { assert(x.size()); return x.back(); }
```

cout

guidelines

defects

client

library

```
struct model : concept_t {  
    model(T x) : data_(move(x)) {}  
    concept_t* copy_() const { return new model(*this); }  
    void draw_(ostream& out, size_t position) const  
    { draw(data_, out, position); }  
  
    T data_;  
};  
  
unique_ptr<concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;  
}  
  
using history_t = vector<document_t>;  
  
void commit(history_t& x) { assert(x.size()); x.push_back(x.back()); }  
void undo(history_t& x) { assert(x.size()); x.pop_back(); }
```

cout

guidelines

defects

client

library

```
template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    concept_t* copy_() const { return new model(*this); }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}

using history_t = vector<document_t>;

void commit(history_t& x) { assert(x.size()); x.push_back(x.back()); }
```



cout

guidelines

defects

client

library



```
};

template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    concept_t* copy_() const { return new model(*this); }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

unique_ptr<concept_t> self_;

};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}

using history_t = vector<document_t>;
```



cout

guidelines

defects

client

library

```
virtual void draw_(ostream&, size_t) const = 0;
};

template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    concept_t* copy_() const { return new model(*this); }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}

using history_t = vector<document_t>;
```

cout

guidelines

defects

client

library

```
virtual concept_t* copy_() const = 0;
virtual void draw_(ostream&, size_t) const = 0;
};

template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    concept_t* copy_() const { return new model(*this); }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

cout

guidelines

defects

client

library

```
virtual ~concept_t() = default; +  
virtual concept_t* copy_() const = 0;  
virtual void draw_(ostream&, size_t) const = 0;  
};  
template <typename T>  
struct model : concept_t {  
    model(T x) : data_(move(x)) {}  
    concept_t* copy_() const { return new model(*this); }  
    void draw_(ostream& out, size_t position) const  
    { draw(data_, out, position); }  
  
    T data_;  
};  
unique_ptr<concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;  
}
```



cout

guidelines

defects

client

library

```
struct concept_t {  
    virtual ~concept_t() = default;  
    virtual concept_t* copy_() const = 0;  
    virtual void draw_(ostream&, size_t) const = 0;  
};  
template <typename T>  
struct model : concept_t {  
    model(T x) : data_(move(x)) { }  
    concept_t* copy_() const { return new model(*this); }  
    void draw_(ostream& out, size_t position) const  
    { draw(data_, out, position); }  
  
    T data_;  
};  
  
unique_ptr<concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;
```

cout

guidelines

defects

client

library

```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
}
```

cout

guidelines

defects

client

library



```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
```



cout

guidelines

defects

client**library**

```
{ x.self_->draw_(out, position); } +  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        concept_t* copy_() const { return new model(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
    unique_ptr<concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{
```

+

cout**guidelines****defects**

client

library

```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>;
void draw(const document_t& x, ostream& out, size_t position)
```



cout

guidelines

defects

client

library



```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    unique_ptr<concept_t> self_;
};

using document_t = vector<object_t>;
```



cout

guidelines

defects

client

library

```
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        concept_t* copy_() const { return new model(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
    unique_ptr<concept_t> self_;  
};  
  
using document_t = vector<object_t>;
```

cout

guidelines

defects

client

library

```
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    unique_ptr<concept_t> self_;
};
```

+

cout

guidelines

defects

client

library

```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    unique_ptr<concept_t> self_;
};
```

cout

guidelines

defects

client

library



```
object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };

    unique_ptr<concept_t> self_;
}
```



cout

guidelines

defects

client

library

```
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        concept_t* copy_() const { return new model(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };
```



cout

guidelines

defects

client

library

```
{ }  
object_t(object_t&&) noexcept = default;  
  
object_t& operator=(const object_t& x)  
{ object_t tmp(x); *this = move(tmp); return *this; }  
object_t& operator=(object_t&&) noexcept = default;  
  
friend void draw(const object_t& x, ostream& out, size_t position)  
{ x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        concept_t* copy_() const { return new model(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };
```



cout

guidelines

defects

client

library

```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
}
```

cout

guidelines

defects

client

library



```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```



cout

guidelines

defects

client

library

{ }

+

```
object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }
```

+

cout

guidelines

defects

client

library

```
object_t(T x) : self_(new model<T>(+move(x)))
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
        void draw_(ostream& out, size_t position) const
        {
            out.write(data_, position);
        }
    };
};
```

cout

guidelines

defects

client

library

```
template <typename T>
object_t(T x) : self_(new model<T>(move(x)))
{ }

object_t(const object_t& x) : self_(x.self_->copy_())
{ }
object_t(object_t&&) noexcept = default;

object_t& operator=(const object_t& x)
{ object_t tmp(x); *this = move(tmp); return *this; }
object_t& operator=(object_t&&) noexcept = default;

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        concept_t* copy_() const { return new model(*this); }
    };

```



cout

guidelines

defects

client

library

```
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(x.self_->copy_())  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
    };
```



cout

guidelines

defects



client

library



```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    object_t(const object_t& x) : self_(x.self_->copy_())  
    { }  
    object_t(object_t&&) noexcept = default;  
  
    object_t& operator=(const object_t& x)  
    { object_t tmp(x); *this = move(tmp); return *this; }  
    object_t& operator=(object_t&&) noexcept = default;  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {
```



cout

guidelines

defects

client

library

```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
```



cout

guidelines

defects



client

library



```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_ ->copy())
    { cout << "copy" << endl; }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
```



cout

guidelines

defects

client

library

```
class my_class_t {
    /* ... */
};

void draw(const my_class_t&, ostream& out, size_t position)
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(string("Hello!"));
    document.emplace_back(document);
    document.emplace_back(my_class_t());

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library



```
/* ... */  
};  
  
void draw(const my_class_t&, ostream& out, size_t position)  
{ out << string(position, ' ') << "my_class_t" << endl; }  
  
int main()  
{  
    document_t document;  
  
    document.emplace_back(0);  
    document.emplace_back(string("Hello!"));  
    document.emplace_back(document);  
    document.emplace_back(my_class_t());  
  
    draw(document, cout, 0);  
}
```

cout

guidelines

defects

client

library

+

};

```
void draw(const my_class_t&, ostream& out, size_t position)
{ out << string(position, ' ') << "my_class_t" << endl; }
```

```
int main()
{
```

```
    document_t document;
```

```
    document.emplace_back(0);
    document.emplace_back(string("Hello!"));
    document.emplace_back(document);
    document.emplace_back(my_class_t());
```

```
    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

+

```
void draw(const my_class_t&, ostream& out, size_t position)
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(string("Hello!"));
    document.emplace_back(document);
    document.emplace_back(my_class_t());

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
void draw(const my_class_t&, ostream& out, size_t position)
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(string("Hello!"));
    document.emplace_back(document);
    document.emplace_back(my_class_t());

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(string("Hello!"));
    document.emplace_back(document);
    document.emplace_back(my_class_t());

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    document_t document;

    document.emplace_back(0);
    document.emplace_back(string("Hello!"));
    document.emplace_back(document);
    document.emplace_back(my_class_t());

    draw(document, cout, 0);
}
```

cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    history_t h(1);

    current(h).emplace_back(0);
    current(h).emplace_back(string("Hello!"));

    draw(current(h), cout, 0);
    cout << "-----" << endl;

    commit(h);

    current(h).emplace_back(current(h));
    current(h).emplace_back(my_class_t());
    current(h)[1] = string("World");

    draw(current(h), cout, 0);
    cout << "-----" << endl;

    undo(h);

    draw(current(h), cout, 0);
}
```

cout

guidelines

defects

client**library**

```
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    history_t h(1);

    current(h).emplace_back(0);
    current(h).emplace_back(string("Hello!"));

    draw(current(h), cout, 0);
    cout << "-----" << endl;

    commit(h);

    current(h).emplace_back(current(h));
    current(h).emplace_back(my_class_t());
    current(h)[1] = string("World");

    draw(current(h), cout, 0);
    cout << "-----" << endl;

    undo(h);

    draw(current(h), cout, 0);
}
```

cout**guidelines****defects**

client

library



```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    object_t(const object_t& x) : self_(x.self_->copy_())
    { cout << "copy" << endl; }
    object_t(object_t&&) noexcept = default;

    object_t& operator=(const object_t& x)
    { object_t tmp(x); *this = move(tmp); return *this; }
    object_t& operator=(object_t&&) noexcept = default;

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual concept_t* copy_() const = 0;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
```



cout

guidelines

defects

client

library



```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        concept_t* copy_() const { return new model(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };
```



cout

guidelines

defects

client

library

```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual concept_t* copy_() const = 0;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        concept_t* copy_() const { return new model(*this); }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };
```



cout

guidelines

defects



client

library



```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
  
    unique_ptr<concept_t> self_;  
};
```



cout

guidelines

defects

client

library



```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(new model<T>(move(x)))  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
  
    shared_ptr<concept_t> self_;  
};
```



cout

guidelines

defects

client

library



```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };

    shared_ptr<const concept_t> self_;
};
```



cout

guidelines

defects

client

library

```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(new model<T>(move(x)))
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };

    shared_ptr<const concept_t> self_;
};
```



cout

guidelines

defects



client

library

A shared ptr to an immutable (const) obj
has value semantics)

```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(make_shared<model<T>>(move(x)))  
    {}  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) {}  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
    shared_ptr<const concept_t> self_;  
};
```

Mutable polymorphic objs are the exception
Copy-on-write using shared_ptr::unique()



cout

guidelines

defects

client**library**

```
{ out << string(position, ' ') << "my_class_t" << endl; }

int main()
{
    history_t h(1);

    current(h).emplace_back(0);
    current(h).emplace_back(string("Hello!"));

    draw(current(h), cout, 0);
    cout << "-----" << endl;

    commit(h);

    current(h).emplace_back(current(h));
    current(h).emplace_back(my_class_t());
    current(h)[1] = string("World");

    draw(current(h), cout, 0);
    cout << "-----" << endl;

    undo(h);

    draw(current(h), cout, 0);
}
```

cout**guidelines****defects**

Compared To Inheritance Based Design

- More flexible
 - Non-intrusive design doesn't require class wrappers
- More efficient
 - Polymorphism is only paid for when needed
- Less error prone
 - Client doesn't do any heap allocation, worry about object ownership or lifetimes
 - Exception safe
- Thread safe

```
template <typename T>
void draw(const T& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    template <typename T>
    object_t(T x) : self_(make_shared<model<T>>(move(x)))
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
}
```



client

library

```
void draw(const T& x, ostream& out, size_t position)
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    template <typename T>
    object_t(T x) : self_(make_shared<model<T>>(move(x)))
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
};

+
```

cout

guidelines

defects

client

library

```
{ out << string(position, ' ') << x << endl; }

class object_t {
public:
    template <typename T>
    object_t(T x) : self_(make_shared<model<T>>(move(x)))
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
};
```

+

cout

guidelines

defects

client

library



```
class object_t {
public:
    template <typename T>
    object_t(T x) : self_(make_shared<model<T>>(move(x)))
    { }

    friend void draw(const object_t& x, ostream& out, size_t position)
    { x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };

    shared_ptr<const concept_t> self_;
}
```



cout

guidelines

defects

client

library

```
class object_t {  
public:  
    template <typename T>  
    object_t(T x) : self_(make_shared<model<T>>(move(x)))  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
  
    shared_ptr<const concept_t> self_;  
};
```

+

cout

guidelines

defects

client

library



```
public:  
    template <typename T>  
    object_t(T x) : self_(make_shared<model<T>>(move(x)))  
    { }  
  
    friend void draw(const object_t& x, ostream& out, size_t position)  
    { x.self_->draw_(out, position); }  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
  
    shared_ptr<const concept_t> self_;  
};
```



cout

guidelines

defects

client

library

```
template <typename T>
object_t(T x) : self_(make_shared<model<T>>(move(x)))
{ }

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };

    shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>;
```

cout

guidelines

defects

client

library

```
object_t(T x) : self_(make_shared<model<T>>(move(x)))
{ }

friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };

    shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>;
```

cout

guidelines

defects

client

library



{ }

```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }
```

private:

```
struct concept_t {
    virtual ~concept_t() = default;
    virtual void draw_(ostream&, size_t) const = 0;
};
```

```
template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }
```

```
    T data_;
};
```

```
    shared_ptr<const concept_t> self_;
};
```

```
using document_t = vector<object_t>;
```

```
void draw(const document_t& x, ostream& out, size_t position)
```



cout

guidelines

defects

client

library



```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
```



cout

guidelines

defects

client

library

```
friend void draw(const object_t& x, ostream& out, size_t position)
{ x.self_->draw_(out, position); }

private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
```

cout

guidelines

defects

client

library

```
{ x.self_->draw_(out, position); } +  
  
private:  
    struct concept_t {  
        virtual ~concept_t() = default;  
        virtual void draw_(ostream&, size_t) const = 0;  
    };  
    template <typename T>  
    struct model : concept_t {  
        model(T x) : data_(move(x)) { }  
        void draw_(ostream& out, size_t position) const  
        { draw(data_, out, position); }  
  
        T data_;  
    };  
  
    shared_ptr<const concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
} +
```

cout

guidelines

defects

client

library



```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
```



cout

guidelines

defects

client

library



```
private:
    struct concept_t {
        virtual ~concept_t() = default;
        virtual void draw_(ostream&, size_t) const = 0;
    };
    template <typename T>
    struct model : concept_t {
        model(T x) : data_(move(x)) { }
        void draw_(ostream& out, size_t position) const
        { draw(data_, out, position); }

        T data_;
    };
    shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```



cout

guidelines

defects

client

library

```
struct concept_t {  
    virtual ~concept_t() = default;  
    virtual void draw_(ostream&, size_t) const = 0;  
};  
template <typename T>  
struct model : concept_t {  
    model(T x) : data_(move(x)) { }  
    void draw_(ostream& out, size_t position) const  
    { draw(data_, out, position); }  
  
    T data_;  
};  
  
shared_ptr<const concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;  
}
```

cout

guidelines

defects

client

library

```
virtual ~concept_t() = default;
virtual void draw_(ostream&, size_t) const = 0;
};

template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}

using history_t = vector<document_t>;
```

cout

guidelines

defects

client

library

```
    virtual void draw_(ostream&, size_t) const = 0;
};

template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}

using history_t = vector<document_t>;
```



cout

guidelines

defects

client

library



```
};

template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

shared_ptr<const concept_t> self_;
```

};

```
using document_t = vector<object_t>;
```

```
void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}
```

```
using history_t = vector<document_t>;
```

```
void commit(history_t& x) { assert(x.size()); x.push_back(x.back()); }
```



cout

guidelines

defects

client

library

```
template <typename T>
struct model : concept_t {
    model(T x) : data_(move(x)) { }
    void draw_(ostream& out, size_t position) const
    { draw(data_, out, position); }

    T data_;
};

shared_ptr<const concept_t> self_;
};

using document_t = vector<object_t>;

void draw(const document_t& x, ostream& out, size_t position)
{
    out << string(position, ' ') << "<document>" << endl;
    for (const auto& e : x) draw(e, out, position + 2);
    out << string(position, ' ') << "</document>" << endl;
}

using history_t = vector<document_t>;

void commit(history_t& x) { assert(x.size()); x.push_back(x.back()); }
void undo(history_t& x) { assert(x.size()); x.pop_back(); }
```

cout

guidelines

defects

client

library

```
struct model : concept_t {  
    model(T x) : data_(move(x)) {}  
    void draw_(ostream& out, size_t position) const  
    { draw(data_, out, position); }  
  
    T data_;  
};  
  
shared_ptr<const concept_t> self_;  
};  
  
using document_t = vector<object_t>;  
  
void draw(const document_t& x, ostream& out, size_t position)  
{  
    out << string(position, ' ') << "<document>" << endl;  
    for (const auto& e : x) draw(e, out, position + 2);  
    out << string(position, ' ') << "</document>" << endl;  
}  
  
using history_t = vector<document_t>;  
  
void commit(history_t& x) { assert(x.size()); x.push_back(x.back()); }  
void undo(history_t& x) { assert(x.size()); x.pop_back(); }  
document_t& current(history_t& x) { assert(x.size()); return x.back(); }
```

cout

guidelines

defects

Concluding Remarks

- As we increasingly move to heavily threaded systems using promises, reactive programming, and task queues, value semantics becomes critical to avoid locking and to reason about code
- It is my hope that the language (and libraries) will evolve to make creating polymorphic types with value semantics easier
- Thanks to Alex Stepanov, Howard Hinnant, and Dave Abrahams

