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

```
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