UE MOBJ [4L103]

Jean-Paul CHAPUT
Jean-Paul.Chaput@lip6.fr

SESI

2018-2019

VI.1

```
<cell name="and2">
  <terms>
    <term name="i0" direction="In"  x="0" y="0"/>
    <term name="i1" direction="In"  x="0" y="0"/>
    <term name="q"  direction="Out" x="0" y="0"/>
  </terms>
  <symbol>
    <box x1="0" y1="0" x2="40" y2="60"/>
    <term name="i0" x1="0"  y1="10" align="top_left"/>
    <term name="i1" x1="0"  y1="50" align="top_left"/>
    <term name="q"  x1="40" y1="30" align="top_right"/>
  </symbol>
</cell>
```

VI.2

```
class Shape;

class Symbol {
  public:
                    Symbol          ( Cell* );
                   ~Symbol          ();
    Cell*           getCell         () const;
    Box             getBoundingBox  () const;
    int             getTermX        ( Term* ) const;
    void            add             ( Shape* );
    void            remove          ( Shape* );
  private:
    Cell*                   owner_;
    std::vector<Shape*>     shapes_;
};
```

VI.3

```
class BoxShape {
  public:
         BoxShape        ( Symbol*, const Box& );
        ~BoxShape        ();
    Box  getBoundingBox  () const;
  private:
    Symbol* owner_;
    Box     box_;
};

class LineShape {
  public:
         LineShape       ( Symbol*, int x1, int y1, int x2, int y2 );
        ~LineShape       ();
    Box  getBoundingBox  () const;
  private:
    Symbol* owner_;
    int     x1_, y1_, x2_, y2_;
};
```

VI.3

```
class TermShape {
  public:
                    TermShape       ( Symbol*, string name, int x1, int y1
                   ~TermShape       ();
            Box     getBoundingBox  () const;
    inline  Term*   getTerm         () const;
    inline  int     getX            () const;
    inline  int     getY            () const;
  private:
    Symbol* owner_;
    Term*   term_;
    int     x1_, y1_;
};
```

VI.3

```
BoxShape::BoxShape ( Symbol* owner, int x1, int y1, int x2, int y2 )
  : owner_(owner), box_(x1,y1,x2,y2)
{ owner_->add( this ); }

LineShape::LineShape ( Symbol* owner, int x1, int y1, int x2, int y2 )
  : owner_(owner), x1_(x1), y1_(y1), x2_(x2), y2_(y2)
{ owner_->add( this );  }

TermShape::TermShape ( Symbol* owner, string name, int x1, int y1 )
  : owner_(owner), term_(NULL), x1_(x1), y1_(y1)
{
  owner_->add( this );
  Cell* cell = getCell();
  term_ = cell->getTerm( name );
}
```

VI.4

```
class Shape {
  public:
                     Shape           ( Symbol* );
                    ~Shape           ();
    inline  Symbol* getSymbol       () const;
            Box     getBoundingBox  () const;
  private:
    Symbol* owner_;
};

inline Symbol* Shape::getSymbol () const { return owner_; }

Shape::Shape ( Symbol* owner ) : owner_(owner)
{ owner->add( this ); }

Shape::~Shape ()
{ owner_->remove( this ); }
```

VI.4

```
class BoxShape : public Shape {
  public:
         BoxShape        ( Symbol*, const Box& );
         BoxShape        ( Symbol*, int x1, int y1, int x2, int y2 );
        ~BoxShape        ();
    Box  getBoundingBox () const;
  private:
    Box  box_;
};

BoxShape::BoxShape ( Symbol* owner, const Box& box )
  : Shape(owner), box_(box)
{ }

BoxShape::~BoxShape ()
{ }

Box  BoxShape::getBoundingBox () const
{ return box_; }
```

VI.4

```
class TermShape : public Shape {
  public:
             TermShape ( Symbol*, string name, int x, int y );
            ~TermShape ();
            Box         getBoundingBox () const;
    inline  Term*       getTerm        () const;
    inline  int         getX           () const;
  private:
    Term* term_;
    int   x_, y_;
};
TermShape::TermShape ( Symbol* owner, string name, int x1, int y1 )
  : Shape(owner), term_(NULL), x1_(x1), y1_(y1) {
  Cell* cell = getCell();
  term_ = cell->getTerm( name );
}
TermShape::~TermShape () { }
Box  TermShape::getBoundingBox () const
{ return Box(x1_,y1_,x1_,y1_); }
```

VI.6

```
class Shape {
  public:
                     Shape           ( Symbol* );
    virtual         ~Shape           ();
    inline  Symbol* getSymbol        () const;
    virtual Box     getBoundingBox   () const = 0;
  private:
    Symbol* owner_;
};
```