Паттерн Visitor

Есилевич Александр

2 декабря 2011 г.

Выделение фигур

- Пользователь кликает мышкой на фигуру
- Появляются точки выделения фигуры
- Пользователь либо перемещает точки выделения, либо перемещает всю фигуру

Вопросы реализации

- Как определить, что кликом в определённой точке выбирается какая-то фигура?
- Как определить какие точки выделения есть у объекта класса `Shape`?

Вопросы реализации

- Для каждой конкретной фигуры можно определить область выделения
- Для каждой конкретной фигуры известны точки выделения

Можно расширить интерфейс класа `Shape` так, чтобы каждый объект этого класса умел определять, выделен ли он кликом в заданной точке, и умел возвращать список точек выделения

Класс SelectPoint

```
public abstract class SelectPoint {
    public abstract Point getPos();
    public abstract void setPos(Point newPos);
}
```

Расширенный интерфейс класса Shape

```
public class Shape {
    ...

    public abstract boolean trySelect(int x, int y);
    public abstract int getSelectPointsCount();
    public abstract SelectPoint getSelectPoint(int index);
}
```

Реализация класса SelectTool

- Четыре состояния: Started, Selected, Move, MovePoint
- В начальном состоянии при клике SelectTool вызывает `trySelect` для всех фигур и определяет выбранную
- В состоянии Selected для отрисовки вызывается `getSelectPoint` для определния положения точек выделения. При нажатии мыши по выделенной фигуре осуществляется переход в состояние Move, При клике на точке – переходв в состояние MovePoint
- При движении мыши в состоянии MovePoint у выбранной точки выделения меняется положение, точка выделения меняет всю фигуру.

Реализация точек редактирования для линии

```
class LineStartSelectPoint extends SelectPoint {
    public LineStartSelectPoint(Line l) {
        line = l;
    }

    public Point getPos() {
        return line.getStart();
    }

    public void setPos(Point newPos) {
        line.setStart(newPos);
    }

    private Line line;
}
```

Реализация точек редактирования для линии

```
class LineEndSelectPoint extends SelectPoint {
    public LineEndSelectPoint(Line l) {
        line = l;
    }

    public Point getPos() {
        return line.getEnd();
    }

    public void setPos(Point newPos) {
        line.setEnd(newPos);
    }

    private Line line;
}
```

Реализация точек редактирования для линии

```
class Line extends Shape {
    ...

    public int getSelectPointsCount() {
        return 2;
    }

    public SelectPoint getSelectPoint(int index) {
        if(index == 0)
            return new LineStartSelectPoint(this);
        else
            return new LineEndSelectPoint(this);
    }

    ...
}
```

Недостатки

- В класс `Shape` добавились "лишние" методы, которые относятся к конкретному инструменту, а не к фигуре
- При увеличении количества инструментов, зависящих от конкретных фигур, интерфейс класса `Shape` будет сильно разрастаться (пример инструмента - линии-коннекторы)

Ключевая проблема

Нужен универсальный способ перебора всех фигур в списке и выполнения определённых действий в зависимости от типа фигуры.

Решение

Паттерн Visitor – описывает операцию, выполняемую с каждым оъектом некоторой структуры, при этом позволяя определить новую операцию не меняя классы объектов

Применимость

- Присутствуют объекты многих классов и требуется выполнять над ними операции, зависящие от конкретных классов
- Выполняемые операции не связаны между собой, поэтому не хочется засорять ими базовый класс всех объектов
- Классы изменяются редко, а операции добавляются часто

Диаграмма классов

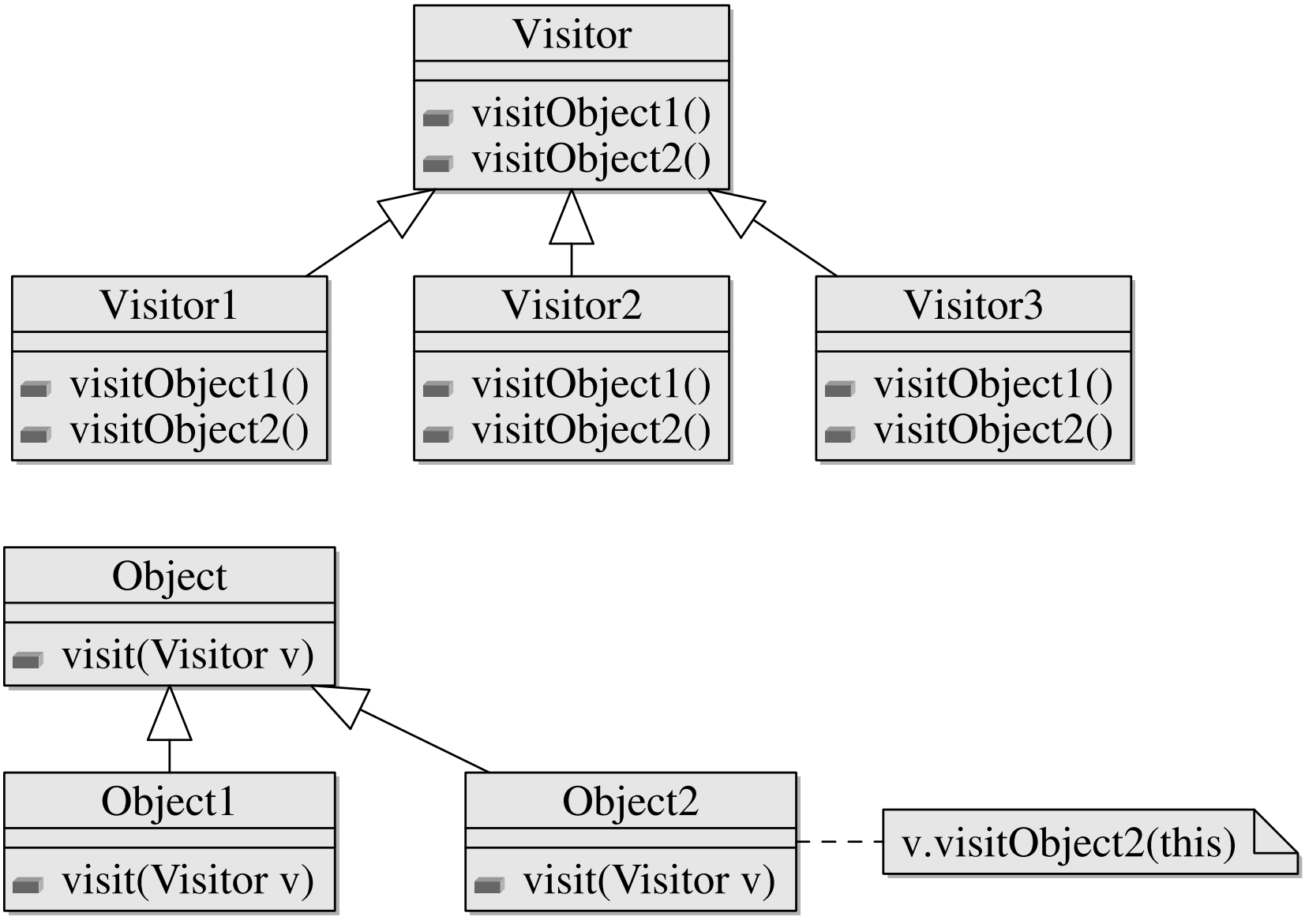
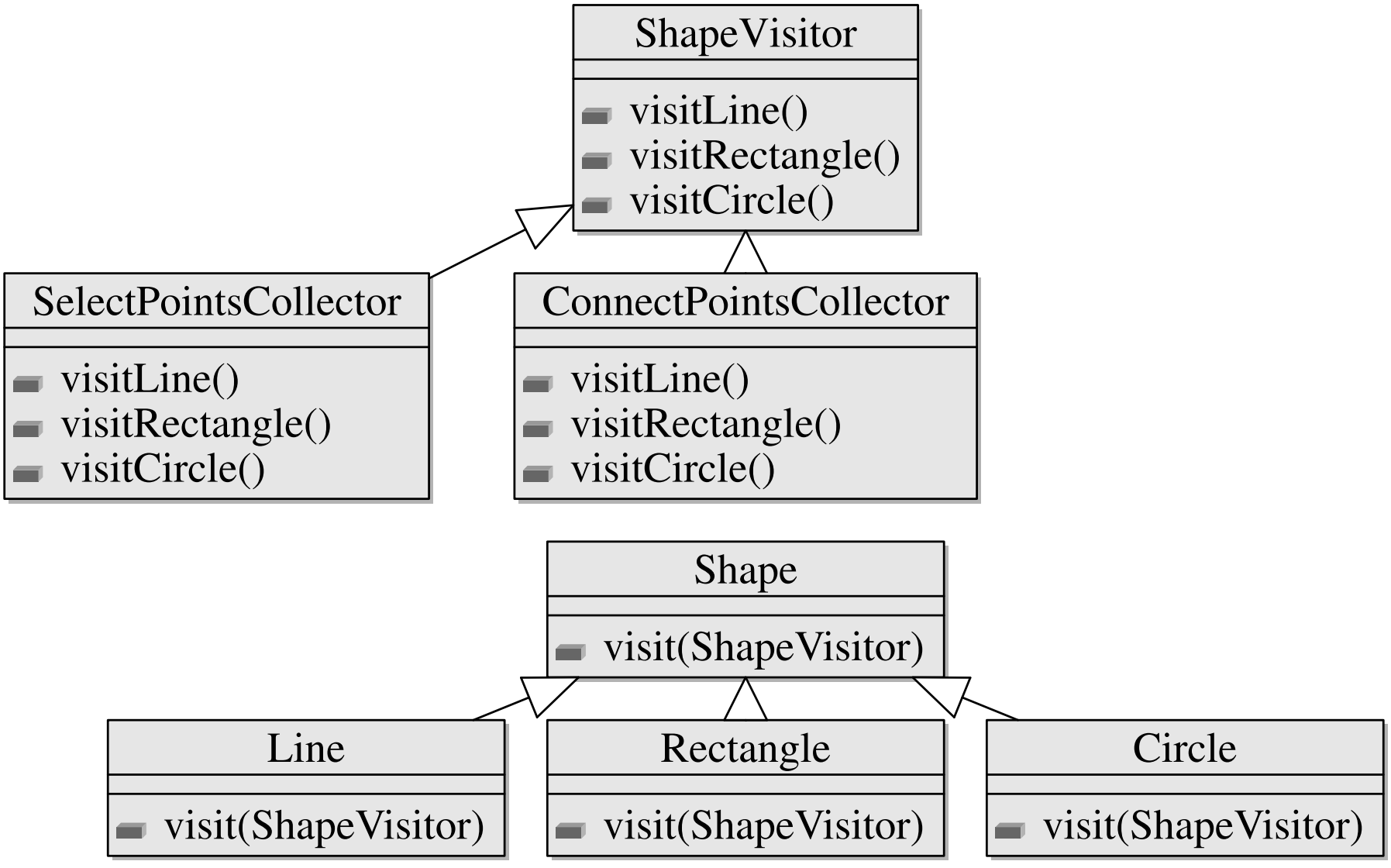


Диаграмма классов для ShapeVisitor



Реализация SelectPointsCollector для линии

```
class SelectPointsCollector extends ShapeVisitor {
    public SelectPointsCollector(Point p) {
        pos = p;
    }

    public void visitLine(Line shape) {
        points.add(new LineStartSelectPoint(shape));
        points.add(new LineEndSelectPoint(shape));
    }

    public SelectPointsList getPoints() { return points; }

    private Point pos;
    private SelectPointsList points;
}
```

Преимущества

- Упрощает добавление новых операций
- Объединяет родственные операции и отедяет друг от друга операции, не имеющие отношения друг к другу
- Позволяет посещать различные структуры, не обязательно однородные

Недостатки

- Затрудняет добавление новых классов в иерархию