Паттерн Observer, MVC

Есилевич Александр

20 декабря 2011 г.

Редактирование свойств фигур

- Пользователь выбирает фигуру
- На панели свойств отображаются свойства фигуры
- При изменении свойств в панели меняется фигура
- При изменении фигуры мышью меняются свойства в панели

Вопросы реализации

- Теперь у каждой фигуры есть состояние (набор координат, свойства) и несколько представлений этого состояния
- Как отделить состояние фигуры от её представления?
- Как добиться того, чтобы можно было легко добавлять новые представления?

Реализация

- Вынесем состояние каждой фигуры в отдельный класс "модели": `LineModel`, `CircleModel`, `RectangleModel`;
- Для каждой фигуры определим определим интерфейс "наблюдателя" за состоянием фигуры: `LineObserver`, `CircleObserver`, `RectangleObserver`;
- Добавим классы моделей методы для регистрации наблюдателей за состоянием модели;
- При изменении состояния модели будем оповещать об изменении всех наблюдателей.

Интерфейс LineObserver

```
public interface LineObserver {
    void update();
};
```

Класс LineModel

```
public class LineModel {
    public void attach(LineObserver o) {
        observers.add(o); }

    public void detach(LineObserver o) {
        observers.remove(o); }

    public void notify() {
     for each o in observers: o.notify(); }

    ...
```

Класс LineModel

```
    ...

    public Point getStart() {
        return new Point(x1, y1); }

    public void setStart(int px1, int py1) {
        x1 = px1; y1 = py1; notify(); }

    private int x1, x2, y1, y2;
    private LineObserverList observers;
};
```

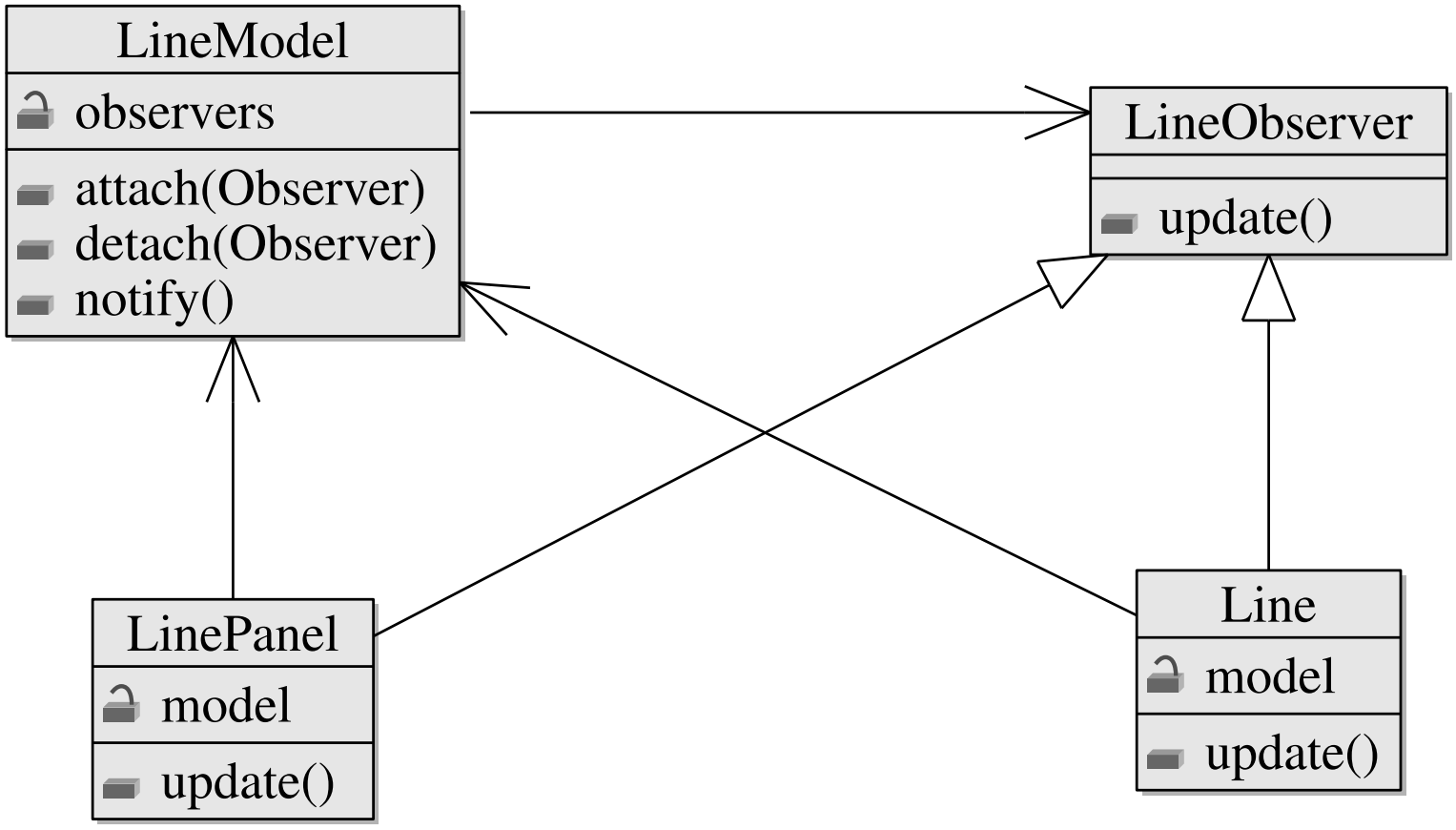
Класс Line

```
public class Line extends Shape implements LineObserver {
    public LineObserver(LineModel mdl) {
        model = mdl; }

    public Point getStart() { return model.getStart(); }
    public Point setStart(Point p) { model.setStart(p); }

    public void notify() {
        // notify scene about changes
    }

    private LineModel model;
};
```

Класс LinePanel

```
public class LinePanel extends SystemComponent
                       implements LineObserver {

    public LinePanel(LineModel mdl) {
        model = mdl;
        // make edit boxes
    }

    public void notify() {
        Point start = model.getStart();
        Point end = mode.getEnd();

        startX.setText(Integer.toString(start.x));
        startY.setText(Integer.toString(start.y));
        endX.setText(Integer.toString(end.x));
        endY.setText(Integer.toString(end.y));
    }

    private LineModel model;
    private SystemEditBox startX, startY, endX, endY;
};
```

Диаграмма классов



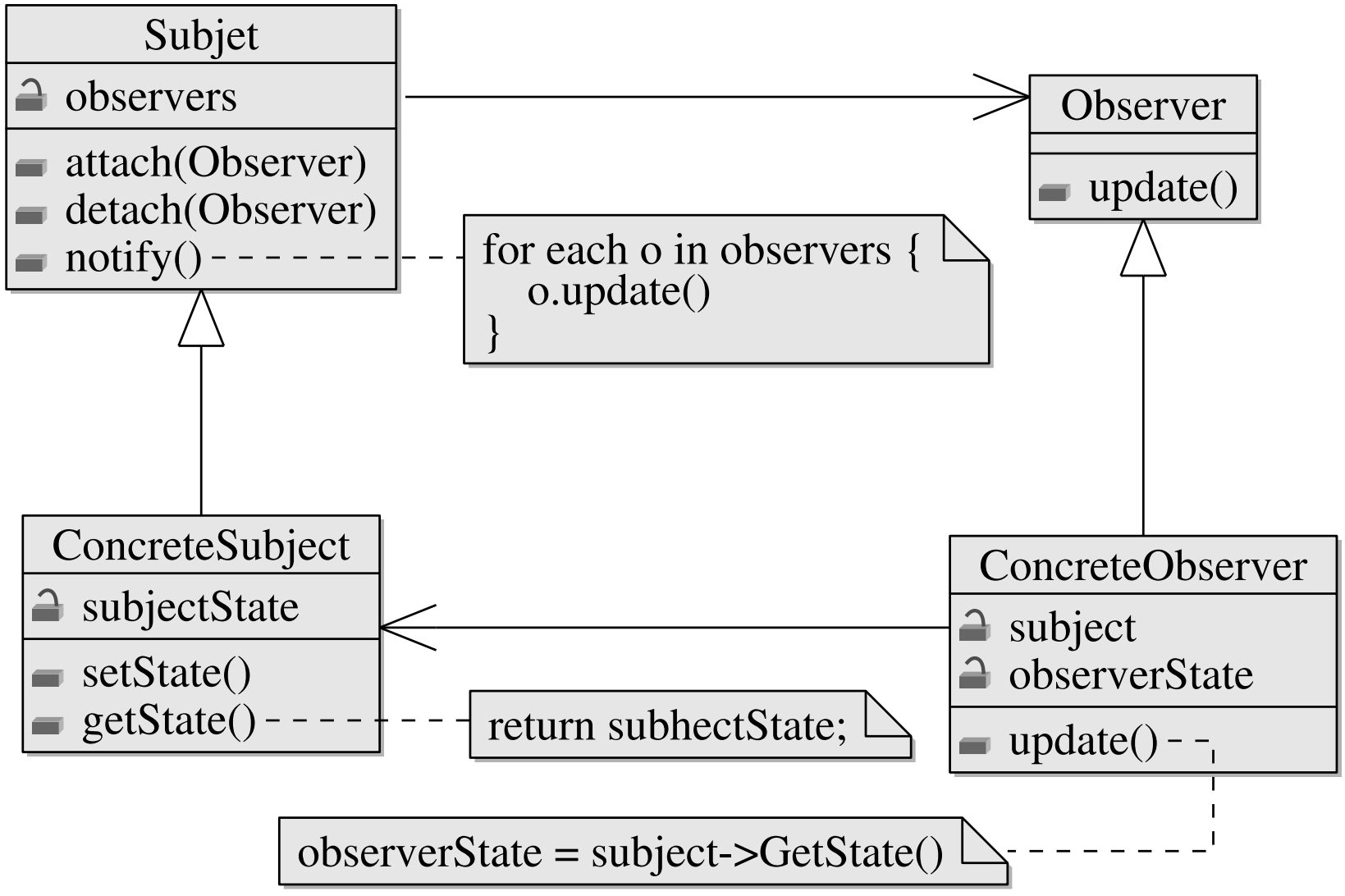
Паттерн Observer

Определяет зависимость "один ко многим" таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом и автоматически обновляются.

Применимость

- у абстракции есть два аспекта, один из которых зависит от другого
- при модификации одного аспекта требуется изменить другие
- один объект должен оповещать других, не зная о конкретных оповещаемых объектах

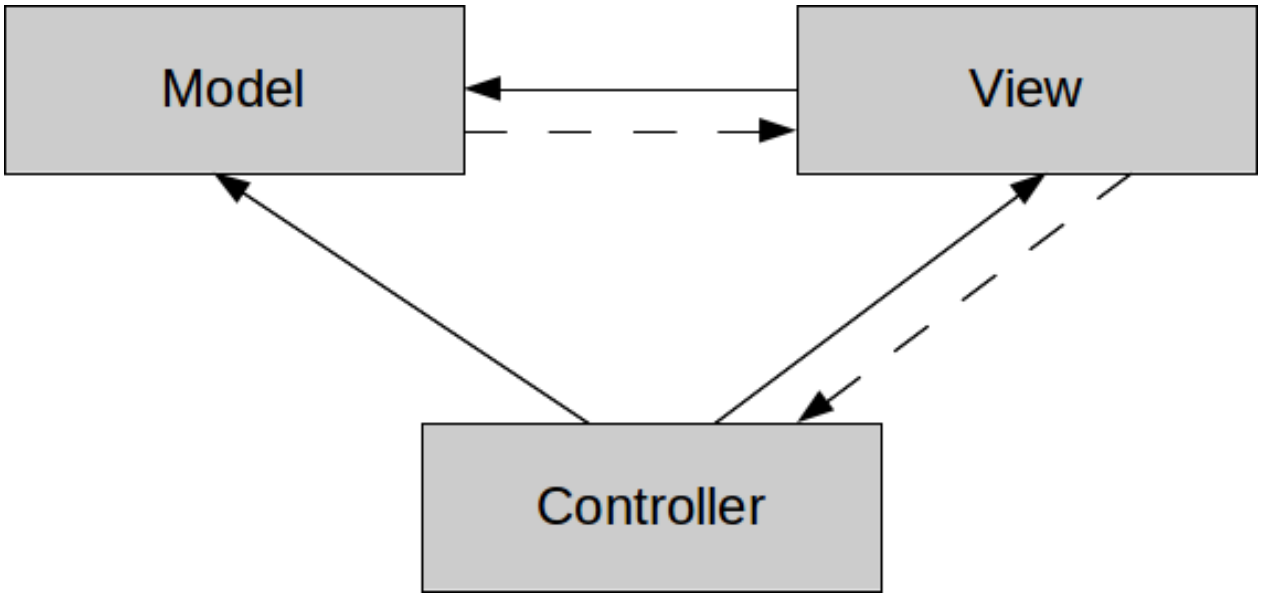
Диаграмма классов



Результаты

- минимизация связей между субъектом и наблюдателями
- поддержка широковещательных коммуникаций
- сложность отслеживания эффектов от изменения субъекта

MVC – Model-View-Controller



Участники

- Model - модель, хранит состояние абстракции
- View - представление абстракции
- Controller - обработчик событий, возникающих в абстракции

Отношения между участниками

- Model-View – паттерн Observer
- View-Controller – паттерн Strategy
- Controller-Model – использование/ассоциация