Паттерн Strategy

Есилевич Александр

2 декабря 2011 г.

Инструмент соединения фигур

- Пользователь выбирает инструмент соединения
- Каждая фигура имеет набор соединительных точек
- При наведении на соединительную точку она подсвечивается
- После клика на точке пользователь тянет линию к другой соединительной точке
- Соединительные линии перемещаются при изменении/перемещении фигур

Инструмент соединения фигур

Существенное требование: доступно несколько вариантов соединительных линий, пользователь может выбирать и изменять тип соединительной линии:

- Прямые линии
- Ломаные линии, идущие по вертикали и горизонтали
- Кривые линии
- Линии, огибающие фигуры

Реализация

- Для идентификации соединительных точек создадим класс `ConnectPoint`, по аналогии с классом `SelectPoint`
- Для определения точек соединения для каждой фигуры используем паттерн Visitor
- Каждая соединительная линия - тоже фигура, которую можно выделять и перемещать, поэтому создадим класс `ConnectorShape`, унаследованный от класса `Shape`

Возникающие проблемы

- Как реализовать изменение типа соединительной линии?
- Для разных типов соединительных линий меняется алгоритм отрисовки и выделения, но все типы соединительных линий содержат ссылки на два объекта класса `ConnectPoint`, и алгоритм перемещения точек соединения одинаковый. Как избежать дублирования кода?
- Алгоритм отрисовки используется не только в нарисованной соединительной линии, но и при её рисовании. Как избежать дублирования кода?

Вариант реализации

Сделаем класс `ConnectorShape` абстрактным, и унаследуем от него отдельный класс для каждой соединительной линии

Недостатки

- Как изменять стратегию отрисовки у существующей соединительной линии? Это не удобно – инструменту изменения свойств требуется удалить линию из списка фигур, и создать другую
- Как переиспользовать алгоритм отрисовки при рисовании линии?

Решение

Паттерн Strategy – отдедяет семейство алгоритмов, инкапсулирует каждый из них, и делает из взаимозаменяемыми

Диаграмма классов

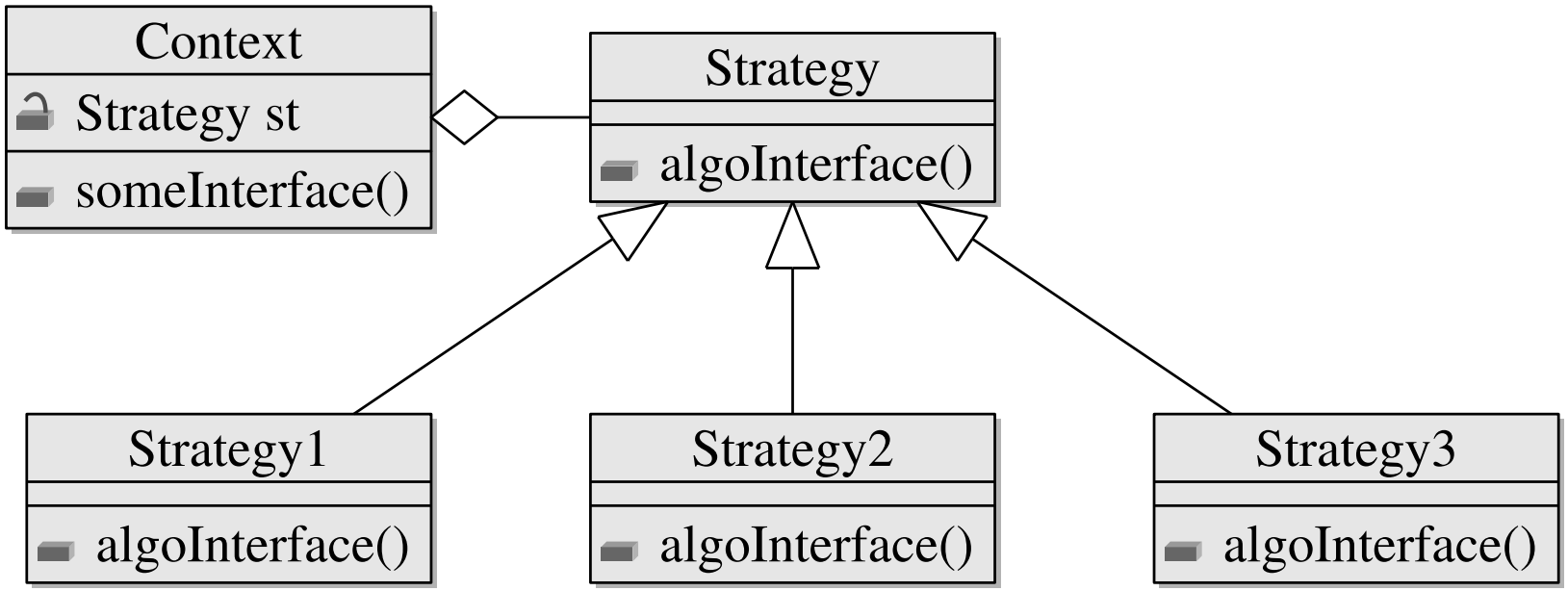
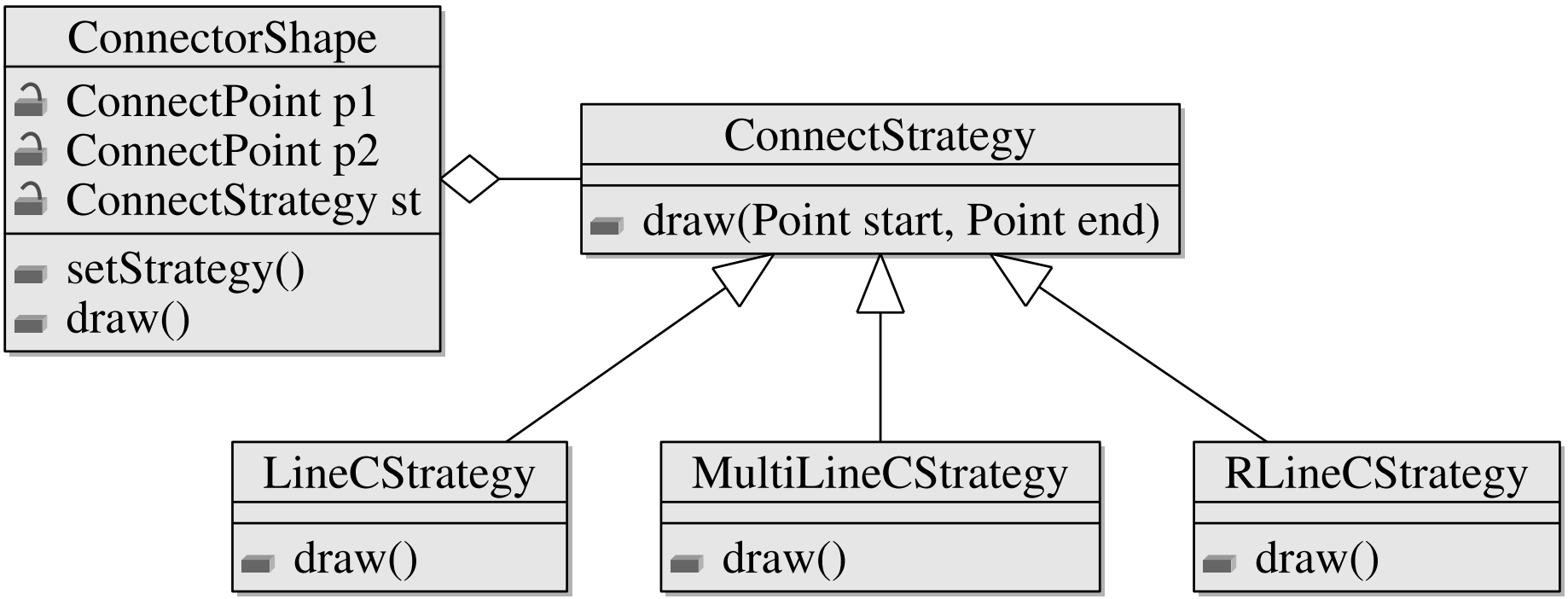


Диаграмма классов для ConnectStrategy



Результаты

- Нет необходимости в громоздских условных операторах
- Нет необходимости порождать новые классы для каждой пары Context + алгоритм
- Алогоритм может быть переиспользован в разных контекстах