

# Паттерны (шаблони) проектирования

Фундаментальные паттерны



Eugeny Berkunsky, Computer Science dept.,  
National University of Shipbuilding  
eugeny.berkunsky@gmail.com  
<http://www.berkut.mk.ua>



# О чем вообще речь?

- **Шаблон проектирования или паттерн** в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.



# Какие бывают паттерны?

- **Основные шаблоны (Fundamental)**
- **Порождающие шаблоны (Creational)**
- **Структурные шаблоны (Structural)**
- **Поведенческие шаблоны (Behavioral)**
- **Частные**
  - **Шаблоны параллельного программирования (Concurrency)**
  - **MVC**
  - **Enterprise**

- Immutable
- Interface
- Abstract Superclass
- Marker interface
- Functional design
- Delegation pattern

# Immutable / Неизменяемый

- Объект может быть неизменяемым как полностью, так и частично.
- В некоторых случаях объект считается неизменяемым с точки зрения пользователя класса, даже если изменяются его внутренние поля.
- Как правило, неизменяемый объект получает все внутренние значения во время инициализации, либо значения устанавливаются в несколько этапов, но до того, как объект будет использован.

# Immutable / Неизменяемый

- Часто, неизменяемые объекты могут быть полезными потому, что они позволяют избежать некоторых дорогостоящих операций копирования и сравнения.
- Таким образом упрощается исходный код программы, и ускоряется ее работа.
- Однако, в некоторых случаях, неизменяемость объекта может мешать, например, если объект содержит большое количество изменяемых.
- Многие языки программирования имеют возможности работы как с изменяемыми, так и с неизменяемыми объектами.

- Неизменяемые в Java:
  - Строки: String
  - Оболочки: Integer, Double, Character и т.д.
  - ... еще?



- **Неизменяемые в Java:**
  - Строки: `String`
  - Оболочки: `Integer`, `Double`, `Character` и т.д.
  - `BigInteger`, `BigDecimal`
  - `java.io.File`
  - `java.util.Locale`
  - `java.net.URL`, `java.net.URI`
  - и много других



- Неизменяемые в Java:
  - Строки: String
- Изменяемые “строки”:
  - StringBuffer
  - StringBuilder



# Interface

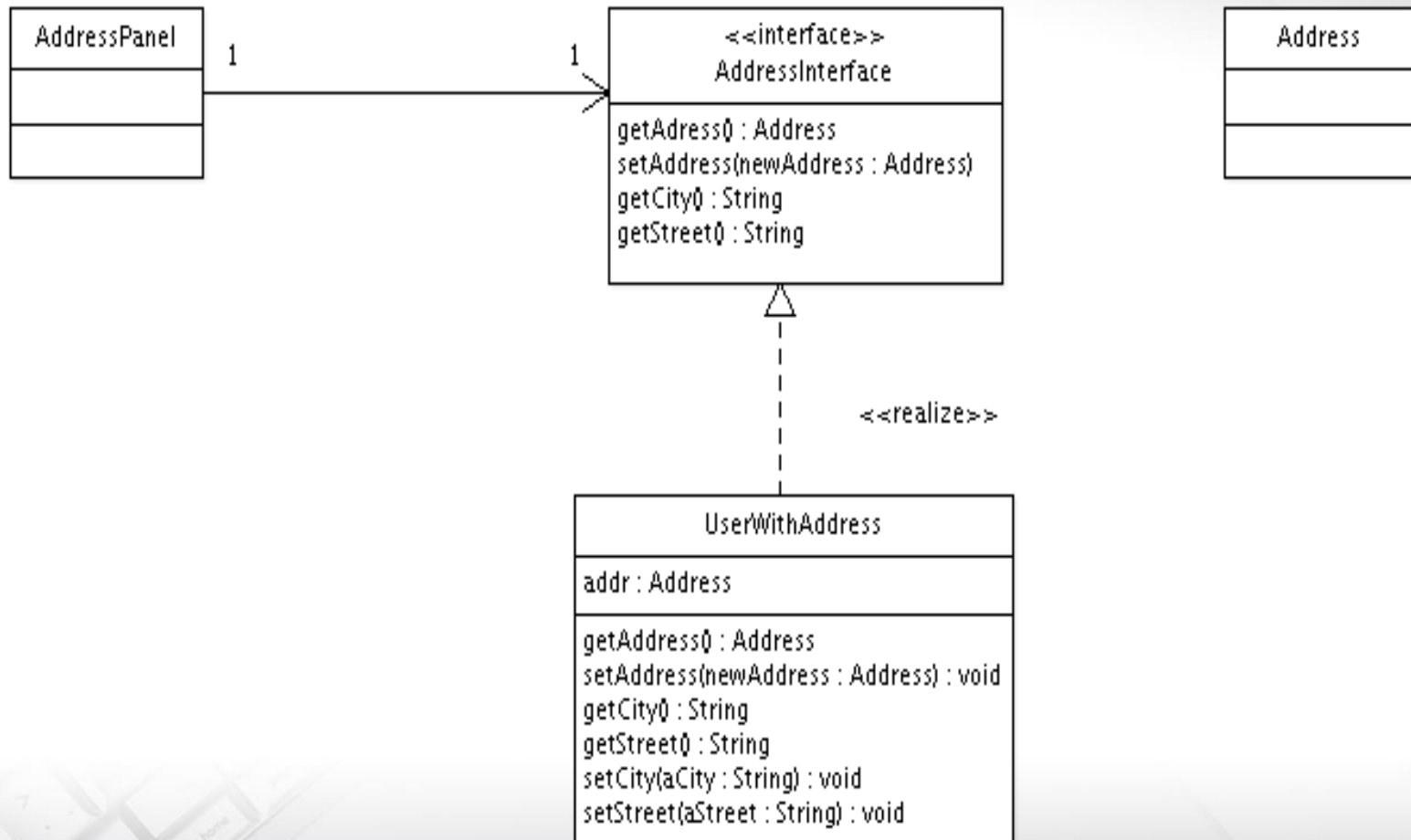
- **Интерфейс** — основной шаблон проектирования, являющийся общим способом структурирования программ для того, чтобы их было проще понять.
- В общем, **интерфейс** — это контракт класса, который обеспечивает программисту простой или более специфический способ доступа к классу из других классов.

# Interface

- Интерфейс является основой для построения более сложных шаблонов:
- Фасад - может содержать набор объектов и обеспечивать простую, высокоуровневую функциональность для программиста
- Адаптер - может использоваться в качестве «клея» между двумя различными API
- и для многих других целей.

- Мотивы использования:
  - Некоторый объект использует другой объект для получения от него данных или сервисов. Если наш объект для получения доступа должен явно указать, к какому классу принадлежит этот объект, то усложняется возможность многократного использования нашего класса из-за сильной связанности.
  - Нужно изменить объект, используемый другими объектами, и при этом нельзя, чтобы эти изменения затронули какой-либо класс, кроме класса изменяемого объекта.
  - *К сожалению, **конструкторы** не могут быть доступны через интерфейс, т.к. интерфейсы в Java не имеют конструкторов.*

# Interface

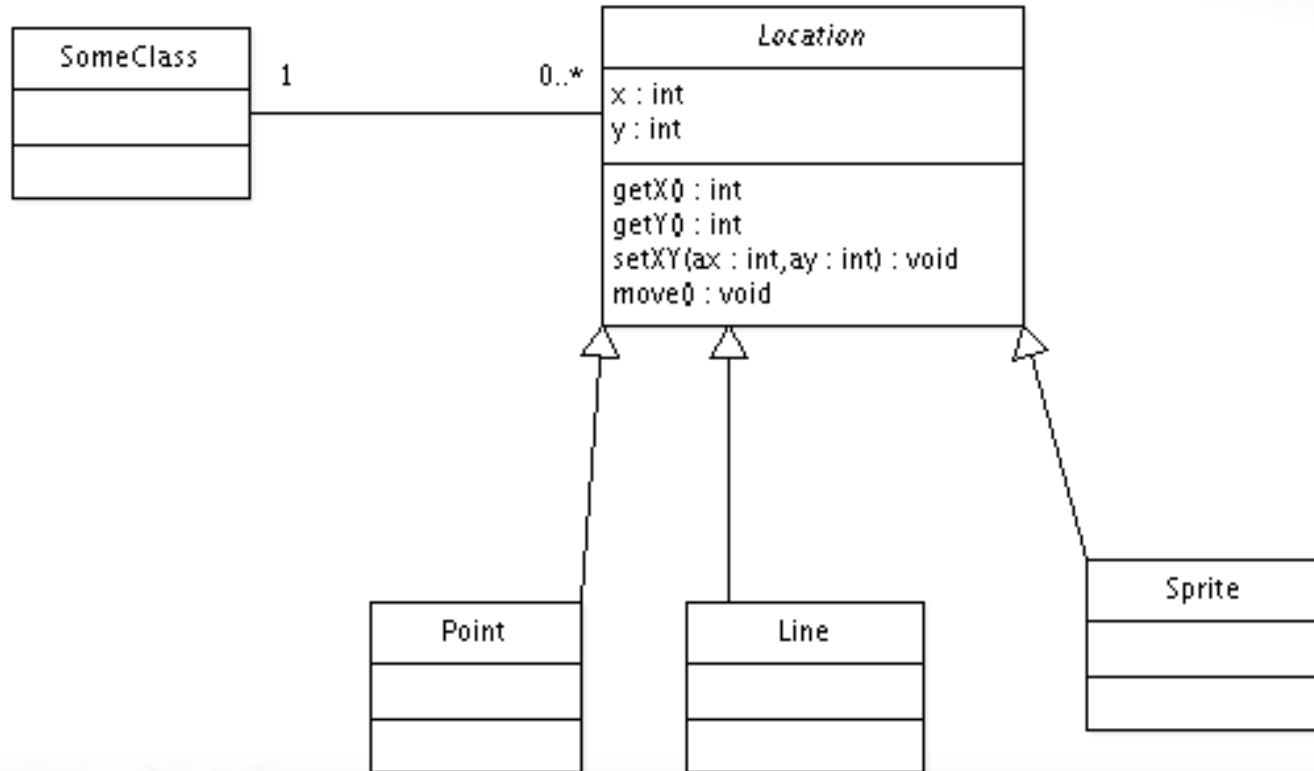


# Abstract Superclass

- **Мотивы**

- Нужно гарантировать, чтобы общая логика для связанных классов реализовывалась одинаково для каждого класса.
- Нужно избежать издержек, связанных со временем разработки и поддержкой излишнего кода.
- Нужно упростить написание связанных классов.
- *Нужно задать общее поведение, хотя во многих случаях наследование не самый подходящий способ его реализации (а, например, Delegation)*

# Abstract Superclass



# Marker interface

- Этот шаблон используется с языками, которые обеспечивают сохранение информации о типах объектов во время выполнения программы.
- Он предоставляет возможность связать метаданные с классом, если язык не имеет явной поддержки для таких метаданных.
- *В современных языках программирования вместо этого могут применяться аннотации.*



- Marker Interface в Java
  - Cloneable
  - Serializable
  - `java.util.EventListener`

# Functional design

- Используется для упрощения проектирования программного обеспечения.
- Функциональный дизайн гарантирует, что каждый модуль программы имеет только одну обязанность и исполняет её с минимумом побочных эффектов на другие части программы.
- Функционально разработанные модули имеют предельно низкую связанность.

# Functional design

- В Java обычно означает, что каждый метод должен выполнять только одно действие.
- Кроме того – каждый класс проектируется так, чтобы выполнять связанные задачи.
- Классы группируются по пакетам по функциональному назначению: `java.util`, `java.io`, `java.sql` и т.п.

# Delegation pattern

## Мотивы:

- Наследование — это статическое отношение, которое не меняется со временем.
- Если оказалось, что в разные моменты времени объект должен быть представлен разными подклассами одного и того же класса, то данный объект нельзя представить подклассом этого общего класса.

# Delegation pattern

## Мотивы:

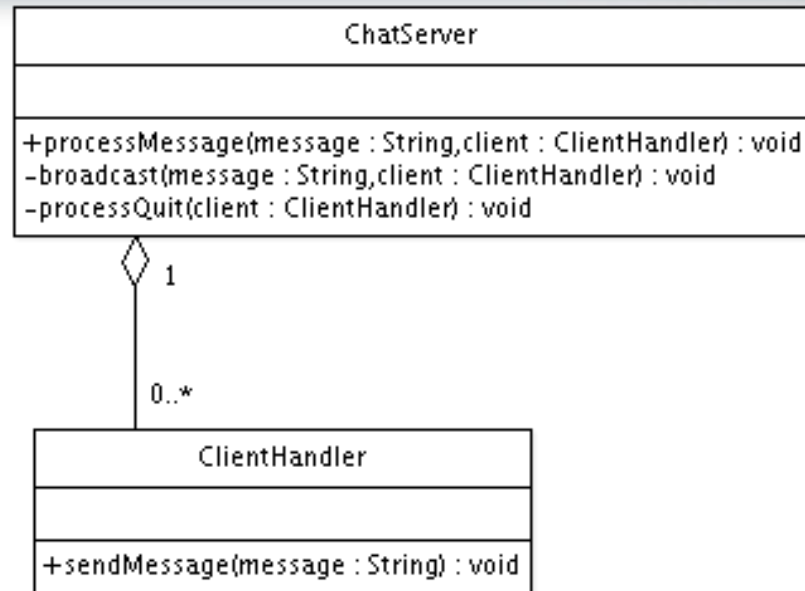
- Если класс пытается скрыть от других классов метод или переменную, унаследованную им от суперкласса, то этот класс не должен наследоваться от такого суперкласса.
- Не существует способа эффективного сокрытия методов и переменных, унаследованных от суперкласса.

# Delegation pattern

## Мотивы:

- «Функциональный» класс (класс, имеющий отношение к функциональности программы) не должен быть подклассом вспомогательного класса.
- Почему?

# Delegation pattern



Для реализации делегирования необходимо, чтобы делегирующий класс содержал ссылку (список ссылок) на класс, которому делегируется выполнение метода.

# Вопросы?





# Паттерны (шаблони) проектирования

Фундаментальные паттерны



Eugeny Berkunsky, Computer Science dept.,  
National University of Shipbuilding  
eugeny.berkunsky@gmail.com  
<http://www.berkut.mk.ua>

