

## Определение паттерна Итератор

Вы уже видели, как паттерн Итератор реализуется в самостоятельно написанных итераторах. Также было показано, как итераторы поддерживаются в некоторых классах коллекций языка Java (таких, как `ArrayList`). Пора ознакомиться с формальным определением паттерна:

**Паттерн Итератор** предоставляет механизм последовательного перебора элементов коллекции без раскрытия ее внутреннего представления.

Итак, паттерн позволяет перебирать элементы коллекции, не зная, как реализована коллекция. Мы уже рассмотрели пример с двумя реализациями меню. Однако применение итераторов в ваших собственных архитектурах приводит и к другим, не менее важным последствиям: при наличии универсального механизма перебора элементов можно написать полиморфный код, который работает с *любыми* коллекциями — как метод `printMenu()`, который работает с элементами, хранящимися в `Array`, `ArrayList` или в любой другой коллекции, способной создать `Iterator`.

Применение паттерна Итератор имеет и другое важное последствие для архитектуры системы: ответственность за перебор элементов передается от объекта коллекции объекту итератора. Это обстоятельство не только упрощает интерфейс и реализацию коллекции, но и избавляет коллекцию от посторонних обязанностей (ее главной задачей является управление объектами, а не перебор).

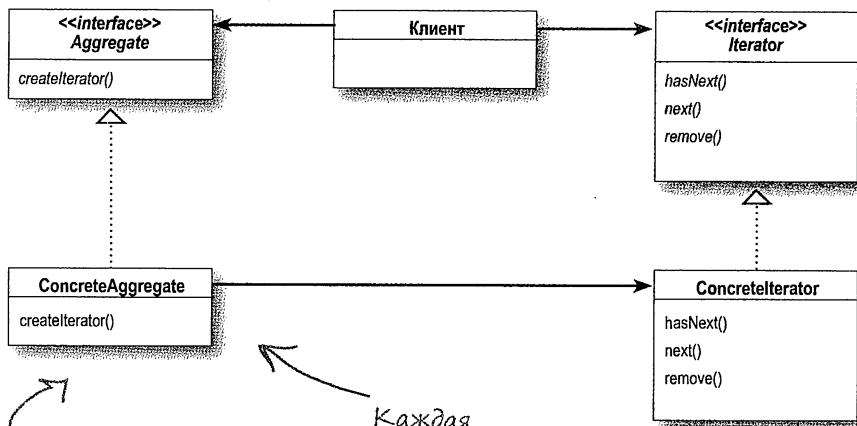
Следующая диаграмма классов поможет лучше понять суть итератора...

**Паттерн Итератор**  
обеспечивает перебор  
элементов коллекции без  
раскрытия реализации.

Кроме того, перебор  
элементов выполняется  
объектом итератора, а не  
самой коллекцией. Это  
упрощает интерфейс  
и реализацию коллекции,  
а также способствует более  
логичному распределению  
обязанностей.

Наличие общего интерфейса удобно для клиента, поскольку клиент отделяется от реализации коллекции объектов.

Интерфейс Iterator должен быть реализован всеми итераторами (как и входящие в него методы перебора элементов). В данном случае мы используем java.util.Iterator. Если вы не хотите использовать интерфейс Iterator языка Java, ничто не мешает вам создать собственный интерфейс.



ConcreteAggregate содержит коллекцию объектов и реализует метод, который возвращает итератор для этой коллекции.

Каждая разновидность ConcreteAggregate отвечает за создание экземпляра ConcreteIterator, который может использоваться для перебора своей коллекции объектов.

ConcreteIterator отвечает за управление текущей позицией перебора.



Диаграмма классов паттерна Итератор очень похожа на диаграмму классов другого паттерна, описанного ранее. Что это за паттерн? Подсказка: создаваемый экземпляр выбирается субклассом.