Эффективность Java

Код: JVA-017

Длительность: 30 ч.

Описание:

Данный курс ориентирован на углубленное понимание языка программирования Java, которое позволит программистам писать более чистый и надежный код с меньшим количеством ошибок и с расширенными возможностями для повторного использования. В курсе подробно рассматриваются паттерны проектирования и языковые идиомы, а также демонстрируется, как максимально эффективно использовать различные возможности от обобщенных типов до перечислений, от аннотаций до автоупаковки.

Каждый раздел этого курса состоит из нескольких пунктов, каждый из которых представлен в форме короткого отдельного обзора, в котором даются конкретные советы и приводятся примеры отличного кода. Полное описание и разъяснения по каждому пункту помогут слушателям понять, что нужно и чего не нужно делать, и почему.

Цели:

Улучшить навыки разработки на языке Java на основе обсуждения и анализ практических примеров.

Разбираемые темы:

1. Создание и уничтожение объектов.

- Выбор статических фабричных методов вместо конструкторов
- Шаблон Builder в случае большого количества параметров конструктора
- Применение семантики шаблона Singleton с приватным конструктором или enum
- Применение приватного конструктора
- Как избежать создания ненужных объектов
- Удаление устаревших ссылок на объекты
- Отказ от использования финализаторов

2. Методы, применяемые ко всем объектам.

- Выполнение общего контракта при переопределении метода equals
- Обязательное переопределение hashCode при переопределении метода equals
- Обязательное переопределение toString
- Осмысленное переопределение клонов

- Выбор реализации Comparable

3. Классы и интерфейсы.

- Минимизация доступности классов и компонентов
- Использование методов доступа вместо public полей
- Минимизация изменчивости
- Предпочтение композиции наследованию
- Проектирование с учётом дальнейшего наследования
- Предпочтение интерфейсов абстрактным классам
- Использование только интерфейсов для определения типов
- Используйте иерархии классов вместо "тег-классов"
- Использование объектов-функций для представления стратегий
- Отдавайте предпочтение статическим внутренним классам над не статическими

4. Обобщенные типы.

- В новом коде всегда используйте generic типы
- Исключение непроверенных предупреждений
- Предпочтение списков массивам
- Предпочтение generic типов
- Предпочтение generic методов
- Использование ограниченных обобщений для повышения гибкости API
- Выбор однородных контейнеров typesafe

5. Enums и аннотации.

- Использование enums вместо констант int
- Использование полей экземпляра вместо ordinals
- Использование EnumSet вместо битовых полей
- Использование EnumMap вместо порядкового индексирования
- Эмуляция расширяемых enums с интерфейсами
- Предпочтение аннотаций паттернам именования
- Всегда используйте аннотацию Override
- Использование интерфейсов-маркеров для определения типов

6. Методы.

- Проверка параметров на валидность
- Использование защитного копирования при необходимости
- Тщательное проектирование сигнатур методов
- Осмысленное использование перегрузки
- Осмысленное использование varargs
- Возврат пустых массивов или коллекций, вместо null
- Написание документации для всех public элементов API

7. Общее программирование.

- Минимизация области видимости локальных переменных
- Предпочтение циклов for-each традиционным циклам for
- Знание и использование библиотек
- Отказ от использования float и double, если требуются точные ответы
- Предпочтение примитивных типов "классам обвёрткам"
- Избегайте строк, где более подходящими являются другие типы
- Учитывайте производительность во время выполнения конкатенации строк
- Ссылки на объекты по их интерфейсам
- Предпочтение интерфейсов к reflection
- Осмысленное использование нативных методов
- Осмысленная оптимизация
- Соблюдение общепринятых норм именования

8. Исключения.

- Использование исключений только для исключительных условий
- Использование checked exceptions для проверки условий и unchecked exceptions для ошибок программирования
- Отказ от ненужного использования checked exceptions
- Предпочтительное использование стандартных исключений
- Выброс исключений соответствующих абстракции
- Документирование всех исключений, выданных каждым методом
- Включение информации о failure-capture в подробные сообщения
- Стремление к атомарности сбоев
- Не допускайте игнорирования исключений

9. Параллелизм.

- Синхронизация доступа к общим изменяемым данным
- Исключение чрезмерной синхронизации
- Предпочтение использованию Executor-ов
- Предпочтение утилит параллелизма методам wait и notify
- Документирование безопасности потоков
- Осмысленное использование отложенной инициализации
- Независимость от планировщика потоков
- Исключение групп потоков

10. Сериализация.

- Осмысленная реализация Serializable
- Использование настраиваемой сериализированной формы
- Безопасное написание методов readObject
- Для контроля экземпляров лучше выбрать типы enum вместо readResolve
- Выбор прокси сериализации вместо сериализированных экземпляров

Целевая аудитория:

Java-разработчики начального и среднего уровня.

Рекомендуемые дополнительные материалы, источники:

Joshua Bloch, "Effective Java"