

JAVA 훑어보기

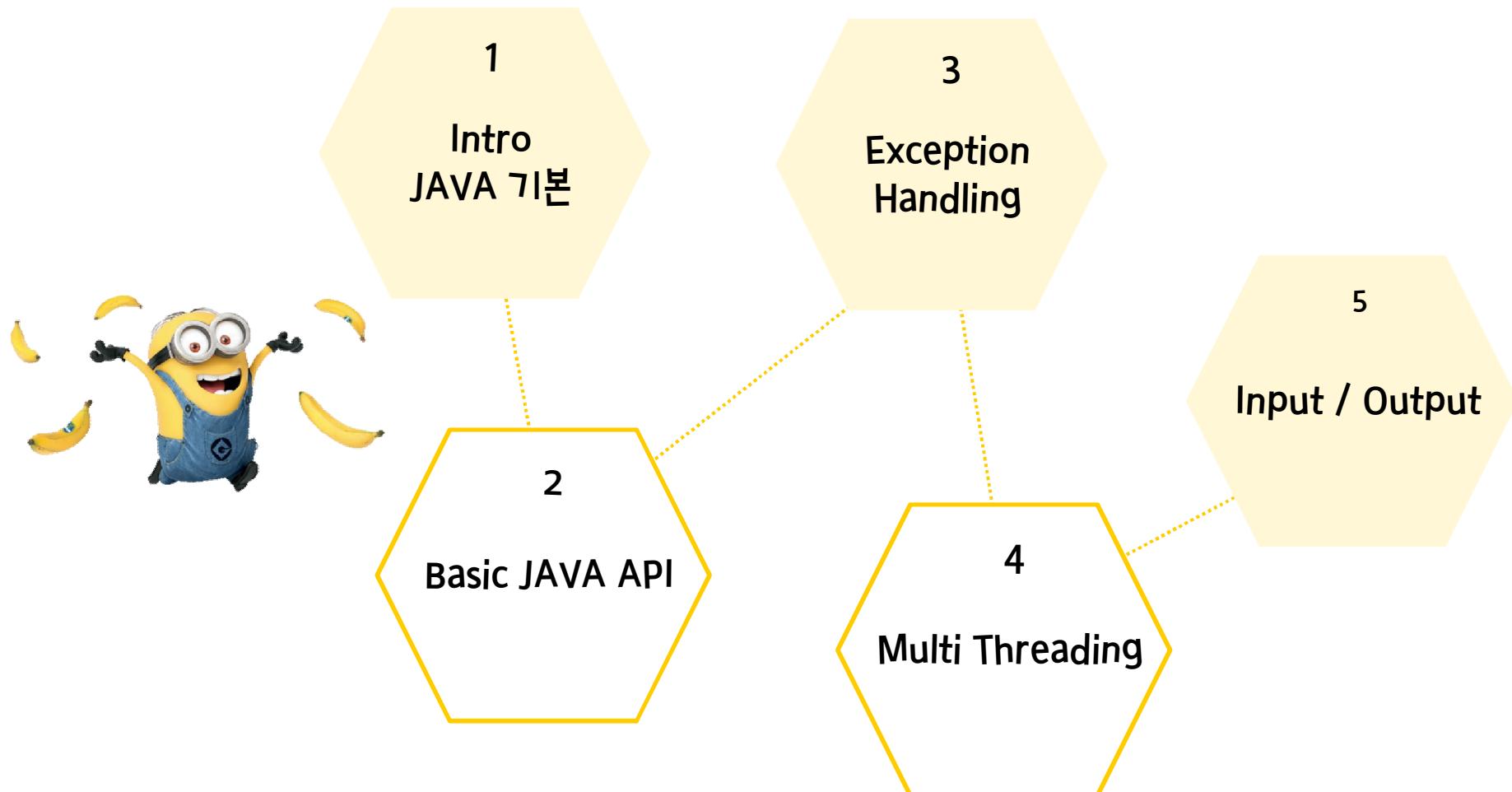
copyright© 2015 Nkak All Rights reserved blog.naver.com/jiwon6315

빠르게 훑어보는 JAVA

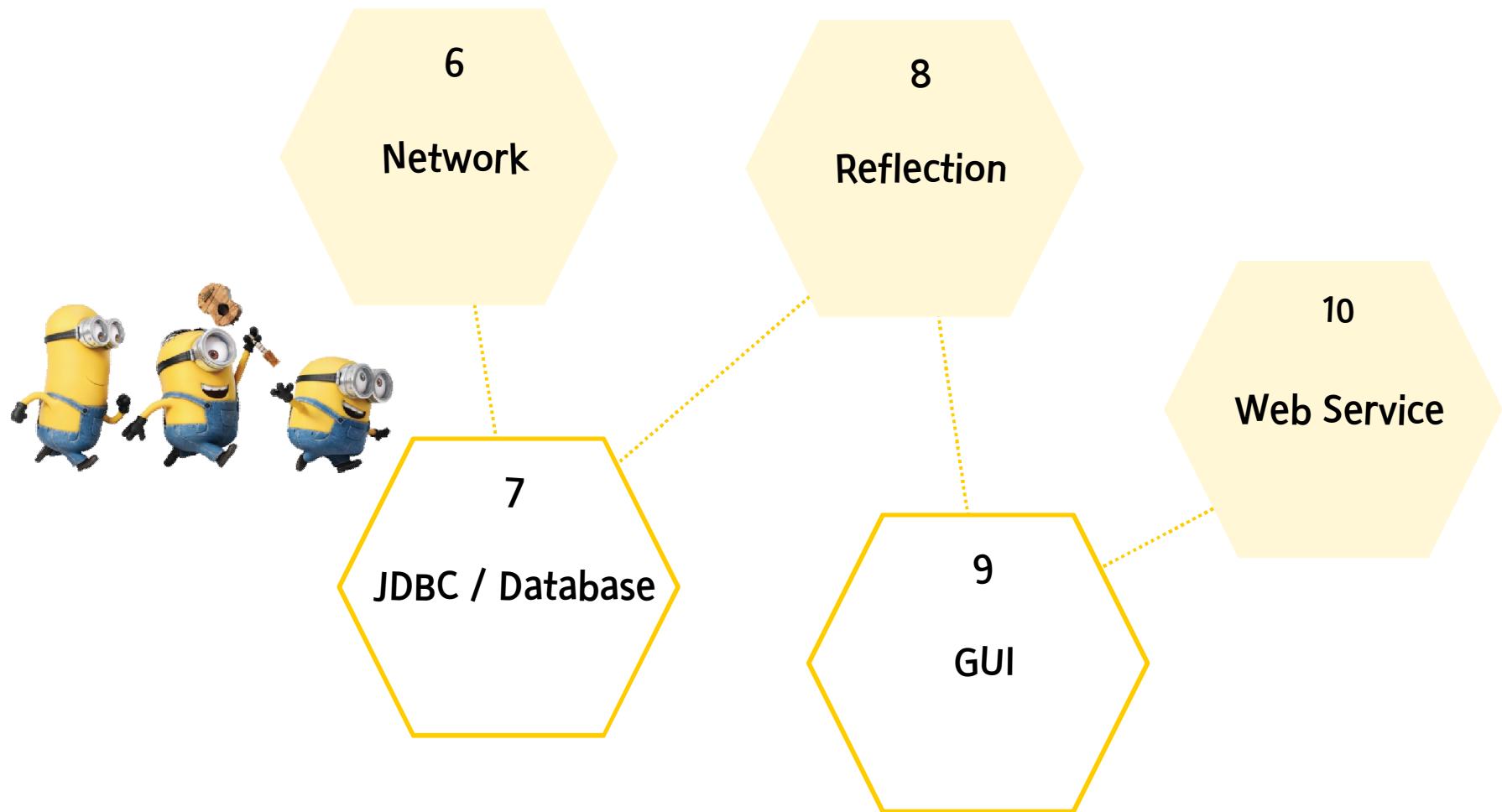


안경훈

Index - 1



Index – 2



Chapter

1

Intro





JAVA

- 객체 지향 프로그래밍 언어의 대표 격.
- JVM(Java Virtual Machine)에서 동작
- **플랫폼 독립적**
- 사용분야
 - Mobile (Android, Java ME)
 - Web / Server(Java EE 등)
 - GUI (Swing, JAVA FX)
 - Embedded (Java ME)



JAVA

- 객체 지향 프로그래밍이란?
 - OOP: Object-Oriented Programming
 - 이전까지 작성되어 왔던 절차적인 명령어 목록으로 보는 시각에서 벗어나 여러 개의 독립된 객체들의 모임으로 파악하고자 하는 것.
 - Class와 Object, Method/Message들로 이루어진다.
 - 추상화 개념과 상속, 다형성 및 동적 바인딩을 지원한다.

Chapter

2

Basic JAVA API





Basic JAVA API

- **java.awt**
- **java.io / java.nio**
- **java.lang**
- **java.net**
- **java.security**
- **java.util**
- **etc...**

Chapter

3

Exception Handling





Exception Handling

- **try / catch / finally**
- **throws Throwable**

```
public static void copy(InputStream inputStream, OutputStream outputStream) throws IOException {  
    try {  
        byte[] buffer = new byte[DEFAULT_BUFFER_SIZE];  
        int n;  
        while (EOF != (n = inputStream.read(buffer))) {  
            outputStream.write(buffer, 0, n);  
        }  
        outputStream.flush();  
    }  
    finally {  
        outputStream.close();  
        inputStream.close();  
    }  
}
```



Exception Handling

- Error
- Exception
 - RuntimeException
 - 그 외의 checked Exception
- Throwable

Chapter

4

Multi Threading





Multi Threading

- Thread란?
 - “실”이라는 의미.
 - 프로세스 내에서 실행되고 있는 명령의 흐름을 뜻함
- Multi Thread란?
 - 한 프로세스 내에서 여러 개의 작업 흐름을 갖는 것.
- 유의할 점
 - 공유 메모리 접근 문제



Multi Threading

- JAVA의 스레드 관련 클래스
 - Thread
 - Executor
 - Future
- 동시성 제어
 - Volatile / Synchronized 키워드
- 관련 패키지
 - `java.concurrent` 패키지

Chapter

5

Input / Output





Input / Output

- Input / Output
 - Java.io 패키지 참조
 - 입력과 출력에 관계된 패키지이다.
 - 입력은 InputStream계열, 출력은 OutputStream계열을 사용
 - 입출력 매체는 File, Console 등이 있다.



Input / Output

- 기존 IO의 문제점
 - 입출력 시 Process Block 발생
Storage > 커널 > JVM 순으로 복사됨 → 느리다.
 - JVM으로 복사해온 메모리는 결국 GC 되어야 한다.
→ 느리다
 - 요약하면 느려서 느리고 느리기 때문에 느리다…
 - 커널 데이터를 직접 사용할 수 없기 때문. 직접 사용 할 수만 있다면 많이 빨라진다.



Input / Output

- NIO (Non-blocking IO)
 - 커널 버퍼를 그대로 사용할 수 있는 Buffer 관련 클래스를 제공(DirectByteBuffer 한정)
 - Channel 도입
 - 읽기/쓰기/읽고쓰기 가능(기존은 단방향 IO만 가능)
 - 다양한 네이티브 IO 서비스를 사용
 - 각 채널을 선택하기 위해 Selector 도입
 - 운영체제에서 관리하는 IO서비스를 그대로 이용할 수 있어 훨씬 빠르다!

Chapter

6

Network





Network

- Java.net 패키지 사용
- 내부적으로 사용되는 클래스들
 - Socket / InetAddress
 - URL / URLConnection
- 많이 사용되는 라이브러리
 - Apache HttpClient
 - Netty/Apache MINA

Chapter

7

JDBC / Database





JDBC / Database

- JDBC(Java Database Connectivity)란?
 - JAVA에서 데이터베이스에 접속할 수 있도록 하는 JAVA API
 - 각 DB마다 JDBC 드라이버를 제공
- java.sql 사용

Chapter

8

Reflection





Reflection

- Reflection?
 - Java의 커다란 특징 중 하나
 - 객체를 통해 클래스의 정보를 분석해 내는 프로그램 기법
 - Framework에서 특히 많이 쓰이고 있으며, Annotation을 사용하기 위해서는 반드시 Reflection을 이용하여야 한다.

Chapter

9

GUI





GUI

- AWT
 - Java.awt 패키지를 이용
- SWING
 - Javax.swing 패키지 이용
- JAVA FX
 - Swing을 대체하기 위해 만들어짐.
 - JDK7버전부터 통합됨
 - JAVA GUI 최후의 보루.

Chapter

10

Web Service





Web Service

- Java EE (Java Enterprise Edition)
 - 서버측 개발을 위한 플랫폼 스펙.
 - Java EE 스펙에 따른 구현체
 - Web Application Server (이하 WAS)
 - 대표적인 WAS
 - JEUS – 티맥스 소프트
 - JBOSS – Redhat
 - WebLogic – Oracle
 - GlassFish – Oracle



Web Service

- **Servlet / JSP**
 - JDK 1.1부터 포함된 Java 진영의 웹 개발 표준
 - JSP는 PHP, ASP(닷넷말고)와 함께 당시 웹개발의 3대장
 - Perl이나 CGI와 달리 멀티스레드 기반의 가벼운 시스템으로 보다 많은 응답을 처리
 - Servlet Container에서 동작



Web Service

- EJB(Enterprise Java Beans)
 - 기업환경에 맞는 대규모 서비스를 지원하는 자바 표준 규약.
 - 엔티티빈, 메시지 드리븐 빈, 세션 빈 등으로 이루어져 있다.
 - 대량 트랜잭션의 안정성, 분산 트랜잭션 지원, 원격 서비스 지원.



Web Service

- EJB의 단점
 - EJB 컨테이너에 종속적이며 불필요한 작업이 많다.
 - 개발이 힘들다.
 - 기능이 막강한 만큼 무겁고 구현이 까다롭다.
 - 개발이 힘들다.
 - 상속구조가 복잡하다.
 - 개발이 힘들다.
 - 등골이 15도 훈다.



Web Service

- Struts
 - Jakarta 서브 프로젝트
 - Java EE 웹 개발을 위한 오픈소스 프레임워크
 - MVC 아키텍쳐 지원
 - Web개발 한정



Web Service

- Spring framework
 - Rod Johnson이라는 아저씨가 만든 자바진영의 짱짱맨 프레임워크(안티도 많다)
 - EJB를 까면서 나타남
 - 엔터프라이즈 시스템의 복잡함을 해결
 - POJO 방식의 프로그래밍
 - 특정 환경(EE)에 구애받지 않음
 - 사실상 대한민국의 웹 개발 표준::::::

Chapter
Bonus

JAVA Evolution History





Java Evolution History

- JAVA 1.4 > 5(JDK 1.5)
 - 네이밍 변경 (JDK 1.x → JDK 5)
 - Generics 추가
 - foreach 지원
 - Enum 타입 추가
 - static import 추가
 - Annotation 추가
 - concurrent api 추가



Java Evolution History

- **JAVA 5 > 6**
 - JAX-WS (Web Services Client)
 - Adding javax.swing.GroupLayout
 - Password prompting
 - Free disk-space API
 - ClassPath wildcards
 - Annotation processing done by javac
 - Solaris Dynamic Tracing (DTrace) Support
 - jhat QQL (Jmap for heap dump)
 - JConsole plugin API
 - Monitoring and Management for the Java Platform
 - OOME Handling
 - Script Langauge Support
 - JDBC 4.0 API



Java Evolution History

- JAVA 6 > 7
 - GC 개선
 - 언어 개선
 - Binary Literals
 - Switch문 String 지원
 - Multiple Catch Exceptions
 - 숫자 리터럴에 언더스코어 지원 (20_000_000)
 - 제네릭 다이아몬드 지원
 - JAVA FX 통합



Java Evolution History

- JAVA 7 > 8
 - 람다 지원
 - 스트림 지원
 - 제네릭 인터페이스 개선
 - 시간/날짜 관련 클래스 추가
 - Collections API 확장
 - Interface에서의 기본 메서드 지원
 - 멀티코어 환경의 병렬 처리 기능 강화





Thank you
Good bye