Hanbit eBook

Realtime 22

BACK TO THE BASIC

JAVA 핵심 요약 노트

빠르게 훑어보는 자바 프로그래밍

김흥래 지음

한빛미디어 Hanbit Media, Inc.

BACK TO THE BASIC

JAVA 핵심 요약 노트

빠르게 훑어보는 자바 프로그래밍

지은이_ **김흥래**

자바카페 커뮤니티에서 운영진으로 활동하고 있다. 심심할 틈 없이 신기술들이 쏟아져 나오는 IT를 매우 사랑하는 개발자로, 개발자들 간의 소통과 교류를 즐긴다. 현재 NHN INS에서 그룹웨어를 개발하고 있다.

• hrkim3468@gmail.com

BACK TO THE BASIC JAVA 핵심 요약 노트 빠르게 훑어보는 자바 프로그래밍

초판발행 2013년 3월 27일

지은이 김흥래 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-609-8 15000 / **정가** 6,000원

책임편집 배용석 / **기획** 김병희 / **편집** 안선화
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
마케팅 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

Published by HANBIT Media, Inc. Printed in Korea

저자 서문

2000년 초부터 자바Java를 접하고 있지만 자바라는 언어는 여전히 쉬운 것 같으면서도 어려운, 뭔가 아리송한 언어인 것 같습니다. C 언어를 다루다 처음 접한 자바는 메모리 관리에서 프로그래머들을 해방시켜준 매우 혁신적인 언어였습니다.

웹이 활성화되면서 자바는 폭발적으로 성장했습니다. 태생적으로 네트워크와 밀접한 관계를 맺고 있다 보니 자바 또한 인터넷 열풍을 타고 급속도로 발전했습니다. 지금은 대부분의 프로그래머가 자바를 이용해 프로그램을 작성할 정도입니다. 자바는 새로운 버전이 릴리즈Release 될 때마다 많은 기능을 추가해 오고 있는데, 지금은 다루지 못하는 영역이 있을까라는 의문이 들 정도로 다양한 분야들을 지원하고 있습니다.

최근 자바의 일곱 번째 버전을 발표하면서 언어 차원의 큰 변화를 예고했습니다. 곧 출시될 여덟 번째 버전에서는 JVM용 자바스크립트 엔진이 추가되고 멀티코어 활용을 위한 Lambda 표현식이 가능해지는 등 한층 더 강력한 기능들을 제공할 것입니다.

이 책은 새롭게 변화하는 자바를 접하기 전에 지금의 자바를 먼저 정리해보자는 생각으로 기획했습니다. Java 6를 기준으로 자바의 기본적인 부분을 훑어보고 자바로 프로그래밍하는 데 필요한 것들을 하나씩 짚어나갑니다. 많은 주제를 부담 없이 읽을 수 있도록, 간단히 가이드하는 형태로 요약해서 설명했습니다.

이 책이 독자들에게 조금이나마 도움이 되었으면 하는 바람입니다.

집필을 마치며,

지은이 **김흥래**

대상 독자 및 도서 구성

초급 | 초중급 | 중급 | 중고급 | 고급

이 책은 Java SE 플랫폼의 구성을 간략하게 요약·정리한 책으로, 자바 입문서를 한 권 이상 보았거나 실무를 막 시작한 새내기 개발자를 대상 독자로 한다.

방대한 자바 API 문서에서 필요한 정보를 찾고자 할 때, 어떻게 해야 할지 몰라 난감할 때가 종종 있을 것이다. 이 책을 참고하여 어떤 클래스가 어떤 시점에 어떻게 활용되는지를 파악하기 바란다. 또한 더 자세한 내용을 쉽게 찾아볼 수 있도록 각 장이 끝날 때마다 각종 레퍼런스들의 정보를 정리해 두었으니 찾아볼 것을 권한다.

프로그래밍하다 보면, 하나의 용어를 영어와 한글로 혼용하여 표기할 때가 많다. 이 책은 용어 표기에 익숙해질 수 있도록 영문 표기와 한글 표기를 적절히 사용하였다. 장별로 용어 표기를 통일하였으니, 이해하는 데 어려움은 없을 것이다.

책의 구성

이 책은 크게 3부분으로 나눠져 있으며, 각 부의 내용은 다음과 같다.

1부 : 알짜배기 자바 이야기

1부에서는 자바로 프로그래밍하기 위해 필요한 일반적인 내용을 간략하게 정리했다. 자바 문법을 간단히 훑어보고 기본적인 객체 지향 개념도 정리해본다. 또한, 자바 프로그램이 실행되는 JVM의 구조도 간단히 살펴본다. 이를 통해 자바 언어 자체가 가지는 특징을 전체적으로 정리해볼 수 있을 것이다.

2부 : Java SE 플랫폼

2부에서는 실제 자바 플랫폼에 대해 알아본다. 자바 플랫폼을 구성하고 있는 JRE 환경과 더불어, 개발자들을 위한 JDK 도구를 살펴보고 자바 플랫폼이 어떻게 구성되어 있는지를 간단히 정리한다. 또한 자바 API를 구성하는 중요한 패키지인 'java.lang 패키지'와 'java.util 패키지'를 간략하게 훑어볼 것이다.

3부 : 필수 라이브러리

3부에서는 자바 개발을 위해 필수적으로 알아야 하는 컬렉션 프레임워크인 JCF를 살펴본다. 이와 함께, 개발의 편의성을 위해 다양한 유틸리티들을 제공하는 Apache Commons 라이브러리 중 하나인 Commons Lang 컴포넌트를 소개한다.

자바카페 커뮤니티

자바카페는 오프라인 중심의 개발자 커뮤니티로, 1999년에 처음 설립된 후 10여 년이 넘는 기간 동안 지속적으로 활동하고 있습니다.

그동안 많은 개발자 커뮤니티가 대한민국 소프트웨어 발전에 이바지해왔습니다. 수많은 커뮤니티가 생성되고 사라지길 반복한 가운데, 지금은 커뮤니티 대부분이 사라졌습니다. 커뮤니티로 10년이 넘는 시간 동안 활동하기란 결코 쉬운 일이 아닙니다. 자바카페는 오프라인 특유의 자생력과 구성원들 사이의 끈끈한 유대 관계에 기반을 두고 활동하고 있습니다.

자바카페는 매년 두 번씩 오프라인 스터디를 진행합니다. 독자 여러분도 자바카페 스터디에 참여하여 다양한 분야의 개발자들을 만나보기 바랍니다. 열린 공간을 제공하는 자바카페 커뮤니티에 많은 관심과 참여 부탁합니다.

- 매년 상반기와 하반기에 스터디 멤버를 모집하고 있습니다.
- 자세한 사항은 홈페이지를 참고하세요.

  http://www.javacafe.or.kr/

  http://www.facebook.com/groups/javacafe/

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook 입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1부 알짜배기 자바 이야기

1부

알짜배기 자바 이야기

1부에서는 자바로 프로그래밍하는 데 필요한 일반적인 내용을 간략하게 살펴본다. 프로그래밍하기 위해 익혀야 할 API 문서 활용법이나 코드 컨벤션에 대해 알아보고, 'Hello World' 프로그램을 예로 들어 주요 문법에 대해 간단히 훑어볼 것이다.

문법을 살펴본 후에는 객체 지향 개념에 대해서도 짚고 넘어간다. 그리고 간단한 코드를 통해 객체 지향 프로그래밍의 주요 특징인 상속성, 캡슐화, 추상화, 다형성의 개념을 정리한다. 마지막으로 자바 프로그램이 실행되는 JVM의 구조와 함께, 자바 프로그램이 어떤 식으로 동작하는지도 알아본다.

이상을 통해 자바 프로그래밍에 필요한 여러 가지 내용을 정리할 수 있을 것이다. 또한 각 단원이 끝날 때마다 레퍼런스 문서 정보를 정리해두었으니 참고하기 바란다. 살펴본 내용이 자바에서 어떤 식으로 활용되고 있는지 정리했으므로 자세한 내용을 찾아보는 데 도움이 될 것이다.

1 | INTRO

자바는 프로그래밍 패러다임이 패키지 기반 소프트웨어에서 웹 기반 소프트웨어로 변하면서 폭발적으로 성장했다. 이에 따라 프로그래밍에 대해 배울 때 C 언어 대신 자바로 시작하는 일도 많아졌다. 지난 수년간 C 언어로 프로그래밍에 입문했던 것을 생각해보면 이는 상당한 변화다. 그만큼 자바의 위상이 높아졌고 이용하는 사람도 많아지고 있다.

현재 기업용 프로그램의 대다수를 자바 기반 기술로 개발하고 있다. 때문에 기업용 프로그램을 비롯한 프로그램을 개발하려면 복잡하고 다양한 자바 기반 기술을 익혀야 한다. 모든 분야가 그렇겠지만, 기술을 지탱하는 기본적인 내용을 충분히 이해하지 못하고서는 고급 기술 역시 이해할 수 없다. 즉, 자바의 기본을 이해하는 것은 자바 고급 개발자가 되기 위한 밑거름이자 필수적인 요건이다.

고급 기술을 학습하기 전에 자바 플랫폼을 전반적으로 학습하고 이해하는 것이 좋다. 자바의 가장 기본적인 플랫폼은 'Java SE'이다. Java SE 플랫폼은 자바의 실행 환경인 JVM(Java Virtual Machine)과 JVM을 구성하고 있는 API와 JVM에 직접 명령을 내릴 수 있는 Java Language로 구성되어 있다.

이번 장에서는 자바 언어의 역사를 간단하게 알아보고 프로그래밍에 필요한 전반적인 내용을 살펴볼 것이다.

1.1 자바 언어의 역사

자바는 1996년 썬 마이크로시스템즈(Sun Microsytems, Inc.)가 처음 선보인 이후, 주인이 오라클(Oracle)로 바뀌는 등 많은 우여곡절을 겪으면서 현재에 이르렀다. 처음에는 임베디드(Embedded) 장비에 탑재할 목적으로 개발되었는데, 때마침 등장한 웹(World

Wide Web 열풍에 따라 웹 프로그래밍에 강점을 지닌 플랫폼으로 진화했다.

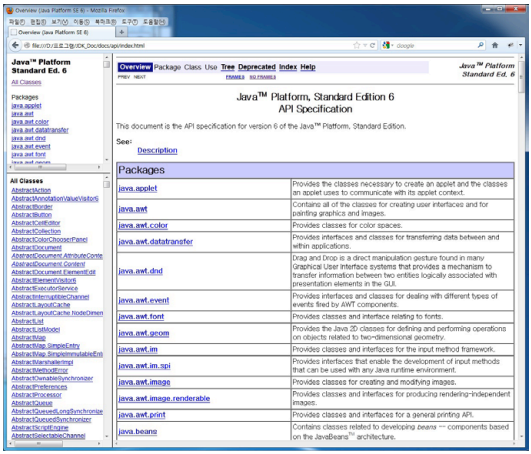
대한민국은 현재 자바 열풍이라 해도 과언이 아니다. 대부분의 기업용 프로그램이 자바 기반의 기술로 구현되어 있으며, 최근 새롭게 등장한 스마트폰 환경 역시 자바 기반의 기술이 대부분 장악하고 있다.

이런 상황을 고려한다면 자바의 앞날은 매우 밝다. 시장의 요구에 따라 현존하는 대부분의 기술과 융합을 시도하고 있으며, 지금도 다양한 개념을 받아들이며 진화하는 중이다. 앞으로도 자바는 IT 산업 전반에 걸쳐 중요한 위치를 차지해 나가리라 생각한다.

1.2 API 문서 가이드

프로그래밍 언어를 학습하고 나면, 가장 많은 시간을 할애하여 살펴보는 것이 관련 API 문서다. API[Application Programming Interface]는 플랫폼에서 제공하는 라이브러리의 명세로, 대부분의 프로그래밍 언어는 API의 사용법을 쉽게 찾아볼 수 있도록 도움말을 제공한다.

그림 1-1 Java API 문서



자바도 공식적인 API 문서[01]를 제공하는데, 워낙 다양한 분야를 지원하다 보니 관련 API 문서의 양이 엄청나다. 처음 API 문서를 접하는 사람이라면 수많은 라이브러리에 주눅부터 들 것이다. 하지만 API 문서가 많다고 걱정할 것은 없다. 자바를 학습하기 위해 반드시 모든 API를 알아야 하는 것은 아니기 때문이다. 우선 빈번하게 사용하는 기본 API를 익힌 후, 필요할 때마다 API를 찾아보고 익히도록 한다.

자바는 또한 매우 강력한 API 문서 생성 툴을 제공한다. 일반적으로 API 문서를 작성하면 문서를 업데이트하는 데 비용이 많이 든다. 실제 소스 코드가 변경되면 API 문서를 같이 변경해야 하는데, 이 작업이 만만치 않기 때문이다. 자바에서는 이러한 문제점을 해결하기 위해 매우 효율적인 방식으로 API 문서를 관리한다. 자바가 API 문서를 생성하고 관리하는 방식은 다음과 같다.

① 소스 코드에 작성된 주석을 일정한 포맷에 맞춰 작성한다.
② 자바에서 제공하는 API 문서 생성 툴을 실행한다.
③ 주석 자체가 API 문서로 완벽하게 변환된다.

자바에서는 API에 대한 모든 설명이 소스 코드에 담겨 있다. 소스 코드가 변경되면 변경된 소스 코드를 이용해 API 문서를 새롭게 생성해 제공한다.

Quick_ 레퍼런스 가이드

- Java SE API 다운로드
  http://www.oracle.com/technetwork/java/javase/documentation/index.html
- Java SE API 온라인 문서
  http://docs.oracle.com/javase/6/docs/api/index.html
- Java 튜토리얼Tutorial
  http://docs.oracle.com/javase/tutorial/index.html

01 • 공식 자바 API 문서(영문) : http://docs.oracle.com/javase/6/docs/api/
• 자바 API 문서(한글) : http://docs.xrath.com/java/se/6/docs/ko/api/

1.3 코드 컨벤션

프로그래밍 언어를 익힌 다음에는 여러 가지 소스 코드를 작성해보면서 자신만의 코딩 스타일을 결정하게 된다. 하지만 이 때문에 코드의 가독성이 떨어지고 다른 사람이 소스 코드를 수정하기가 어려워진다. 이러한 문제점들을 해결하기 위해 다양한 코드 컨벤션Code Convention 방법을 사용한다.

코드 컨벤션이란 소스 코드를 작성하는 방식에 대한 가이드라인을 모아둔 집합이다. 처음에는 물론 코드 컨벤션에 따라 코딩하는 것이 불편하겠지만, 익숙해지면 코드 가독성과 생산성이 높아진다. 따라서 자신만의 코딩 스타일을 유지하는 것보다는 잘 정리된 코드 컨벤션을 따르는 것이 좋다.

다양한 프로그래밍 언어가 있는 것처럼 코드 컨벤션을 구성하는 명명법도 다양하다. 명명법에 따라 장단점이 다르니 함께 일하는 팀원들과 협의하여 코드 컨벤션을 정의하기 바란다. 변수명을 정의할 경우 전통적으로 유닉스Unix/Linux 기반에서는 '언더스코어 표기법'을, 윈도우Windows 기반에서는 '헝가리언 표기법'을 선호했다. 하지만 최근 자바나 C# 같은 객체 지향 언어로 개발하면서는 이러한 구분이 희석되는 추세다.

기본적인 컨벤션 정의를 위해 자바에서도 공식적으로 코드 컨벤션 문서[02]를 제공하고 있다.

표 1-1 자바에서 사용하고 있는 대표적인 표기법

표기법	특 징
파스칼 표기법 **(Pascal Notation)**	모든 단어의 첫 번째 문자를 대문자로 표기하는 방법이다. • 각 단어의 첫 번째 문자를 대문자로 표기하고 나머지를 소문자로 표기한다. • 대표적인 사용처 : 클래스class 이름, 인터페이스interface 이름 • 사용례 : class Car

02 http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

카멜 표기법 **(Camel Notation)**	단어와 단어 사이를 대문자로 구분하는 방법으로, 낙타의 혹을 닮았다고 해서 붙여진 이름이다. • 각 단어의 첫 번째 문자를 소문자로, 나머지를 대문자로 표기한다. • 대표적인 사용처 : 메소드Method 이름, 변수명 • 사용례 : void speedUp( ), int speed
헝가리언 표기법 **(Hungarian Notation)**	특별한 형식의 접두어를 사용하여 표기하는 방법이다. • 타입 + 이름 형식으로 작성한다. • 대표적인 사용처 : AWT, SWING 등에서 GUI 적용 시 • 사용례 : txtName, lblAge
전체 대문자 **(Upper)**	모든 단어를 대문자와 언더스코어("_")로 표기하는 방법이다. • 대표적인 사용처 : 상수명 • 사용례: float PI = 3.14
전체 소문자 **(Lower)**	모든 단어를 소문자로 표기하는 방법이다. • 대표적 사용처 : 패키지Package 이름, 예약어, 키워드 • 사용례 : class, private, static

Quick_ 레퍼런스 가이드

- Java Code Conventions
  http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html

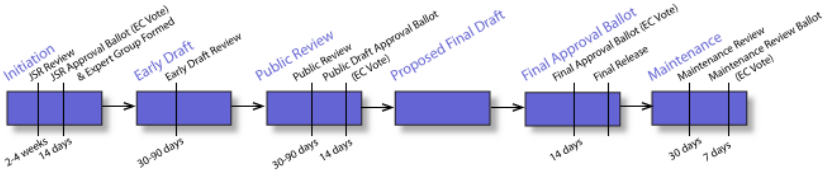
1.4 JSR(Java Specification Request)

자바의 최신 버전은 2011년 출시된 '자바 7'이다. 새로운 버전으로 출시될 때마다 자바는 항상 많은 기능이 추가된다. 현재 자바는 단순한 언어를 뛰어넘어 플랫폼으로 진화했다. 우리가 배운 자바라는 프로그래밍 언어는 자바 플랫폼의 일부일 뿐이다. 자바 플랫폼은 다양한 기술의 집합체며, 자바 언어는 이러한 자바 플랫폼에 명령을 내리기 위한 훌륭한 도구다.

새로운 기술이 자바 플랫폼에 포함될 때마다 JCP[Java Community Process]라는 독특한 과정을 거치는데, 이를 통해 기술이 성숙되면 새로운 자바 표준으로 결정된다. JCP[03]는 1998년에 최초로 설립된 표준 지정 방식으로, 자바 기술에 관련한 각 분야의 전문가들[Export Group]이 자바 플랫폼의 향후 버전과 기능을 정의하고 표준화하는 과정을 일컫는 말이다. 자바 표준에 관심 있는 사람이라면 누구든 JCP 과정을 통해 자바에 필요한 기능을 표준으로 제안할 수 있다.

JCP에는 JSR[Java Specification Request]이라는 표준 문서가 있다. 자바 플랫폼을 구성하는 기술들의 표준을 정리한 공식 문서로서, 표준으로 제안할 내용이 있는 사람이라면 누구든 JSR 문서를 제출하여 새로운 표준을 제안할 수 있다. 새로운 JSR 문서가 등록되면 JCP 과정을 통해 전문가 그룹이 문서를 검토하고, 제안한 내용이 최종 승인을 통과하면 공식적인 표준으로 인정받는다.

그림 1-2 JSR Release 과정(http://jcp.org/en/procedures/overview)



현재 JCP 과정을 거쳐 생성된 JSR 문서는 300여 개가 넘으며, 다양한 분야의 표준이 승인되었거나 진행 중에 있다. 자바가 어떻게 발전할지 궁금한 사람은 JSR 문서를 관심 있게 지켜보기 바란다.

> **NOTE_** 가장 최근에 출시된 Java SE 7은 'JSR 336'으로 지정되어 있다.
>
> • JSR 336: Java SE 7 Release Contents (http://jcp.org/en/jsr/detail?id=336)

03 http://jcp.org/en/home/index

2 | Hello World 완전 분석

자바 프로그래밍을 처음 시작할 때 반드시 거치는 과정이 있다. 바로 자바 언어의 문법을 익히는 과정이다. 문법을 익히는 과정은 대부분 지루하고 이해하기 어려운 난관의 연속이다. 만약 C 언어를 어느 정도 마스터한 후 자바 언어를 익히는 경우라면, C 언어와 자바 언어의 차이점 및 장단점을 비교하면서 학습하는 것이 많은 도움이 될 것이다. 하지만 요즘은 학부에서조차 C 언어 대신 자바를 바로 가르치는 경우가 많아서 이같이 하기도 쉽지 않다. 결국 각개격파하는 방법밖에는 없다.

이 책은 자바 문법의 기본을 한 번쯤 공부한 독자를 대상으로 하는 만큼, 기본적인 문법을 논하기보다는 자바 언어의 주요 문법을 빠르게 리뷰해보고 객체 지향의 핵심 개념을 정리하는 데 중점을 두고 살펴볼 것이다.

누군가 세상에서 가장 유명한 프로그램이 뭐냐고 묻는다면 단연 'Hello World'라고 말할 것이다. Hello World 프로그램은 프로그램 언어를 배울 때 가장 먼저 만나는 프로그램으로, 통합 개발 환경(Compiler, Debugger, Linker 등)을 테스트할 목적으로 자주 사용한다.

이 프로그램은 1978년 브라이언 커니건Brian Kernighan과 데니스 리치Dennis Ritchie가 집필한 『The C Programming Language』(Prentice Hall, 1978년 2월)라는 책에 등장하며서 유명해졌다. 지금은 거의 모든 언어의 샘플 코드로 HelloWorld 프로그램이 제공되고 있다.

그런 만큼 자바 언어를 알아보기에 앞서 Hello World 프로그램에 대해 먼저 알아보자.

소스 코드 2-1 HelloWorld.java

```
package kr.or.javacafe.hello;
```

```
public class HelloWorld {
   public static void main(String args[]) {
      System.out.println("Hello World!");
   }
}
```

아마 소스 코드를 외울 정도로 많이 본 프로그램일 것이다. 이 프로그램을 실행하면, 단순히 콘솔 화면에 'Hello World!'라는 문자열이 한 줄 출력된 후 종료된다.

실행 결과

```
Hello World!
```

단순한 프로그램이다. 하지만 자세히 살펴보면 이 프로그램 안에 자바의 주요 문법과 개념이 다수 포함되어 있다. 지금부터 Hello World 프로그램을 자세히 분석해 보자.

2.1 클래스

클래스Class는 자바 프로그램을 만들기 위한 기본 단위다. 자바를 한마디로 표현하면 '클래스의 묶음'이라고 해도 과언이 아니다. 자바의 모든 API는 클래스 형태로 제공되며 우리가 작성하는 프로그램 또한 클래스 형태로 작성해야 한다. 이처럼 클래스는 자바에서 매우 중요한 개념이다.

클래스를 간단히 표현하면 "객체Object를 정의하기 위한 강력한 데이터 형식"이라고 할 수 있다. 클래스는 객체를 만들기 위한 '데이터(변수)'와 이를 처리하기 위한 '기능(메소드)'을 하나로 묶어서 표현한다.

표 2-1 객체 모델Object Model

일반적인 물체	객체(Object)
물체(자동차)	클래스(Class)
속성(색상, 이름)	변수(Variable)
동작(가속, 감속)	메소드(Method)

프로그래머는 특정 사물을 클래스로 정의한다. 하지만 클래스 그 자체를 직접 사용할 수는 없다. 이 클래스를 프로그램에서 사용하려면 어떻게 해야 할까?

클래스를 사용하려면, 클래스를 객체로 만들어야 한다. 즉, 정의된 클래스를 컴파일한 후 실행Runtime 시 메모리상에서 인스턴스화 되어야만 비로소 사용할 수 있다.

Hello World 클래스 또한 컴파일되어 메모리상에서 객체가 된다. Hello World 프로그램은 표면상으로는 하나의 main( ) 메소드만 있는 간단한 형태의 단일 클래스 프로그램이다. 하지만 컴파일 과정을 거치면서 컴파일에 필요한 다양한 클래스와 함께 컴파일되고 복잡한 형태의 바이트 코드로 생성되어 메모리에서 객체화된다.

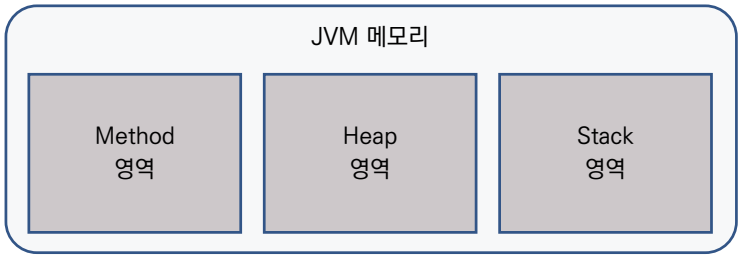
2.2 main( ) 메소드와 static 키워드

자바에서 메소드를 호출하려면 메소드를 가지고 있는 객체가 메모리상에 반드시 인스턴스화된 상태여야 한다. 하지만 예외적인 메소드가 있는데, 바로 프로그램의 시작점인 main( ) 메소드다.

프로그램이 실행되면 메모리상에서 인스턴스화된 객체들이 서로를 호출하는 방식으로 프로그램이 구성된다. 하지만 최초로 프로그램을 실행할 때는 JVMJava Virtual Machine이 객체를 호출할 진입점이 필요한데, 이 진입점 역할을 하는 것이 바로 main( ) 메소드다.

모든 클래스는 new 키워드를 이용하여 메모리상에서 인스턴스화해야 실행된다. 하지만 Hello World 프로그램을 자세히 살펴보면 클래스를 인스턴스화하는 new 키워드가 없다. Hello World 프로그램은 new 키워드로 인스턴스화하지 않아도 실행하는 데 아무런 문제가 없다. 이유가 뭘까? 비밀은 바로 main( ) 메소드 앞쪽에 선언된 static 키워드다. 왜 main( ) 메소드에는 static 키워드가 붙어 있는 것일까?

그림 2-1 JVM 메모리 구조



static으로 선언된 메소드는 메모리에 올라갈 때 일반적인 'Heap 영역'에 생성되는 것이 아니라 메모리의 별도 영역인 'Static 영역(Method 영역)'에 생성된다. 원래 객체가 Heap 영역에 생성되려면 반드시 new 키워드를 이용하여 실행Runtime 시 별도의 공간을 할당받아야 한다. 하지만 Static 영역은 프로그램이 실행되는 동시에 메모리에 생성되므로, 별도의 공간 할당 작업을 하지 않아도 된다.

main( ) 메소드는 프로그램을 실행하기 위한 진입점으로 사용하기 위해 특별히 정의된 메소드다. JVM은 프로그램 실행과 동시에 main( ) 메소드를 실행해야 하기 때문에 main( ) 메소드는 Static 영역에 생성되어 있어야 한다. 이처럼 main( ) 메소드에 static 키워드가 붙어 있는 이유는 JVM과 메모리상의 구조적인 측면 때문이다. 만약 프로그램 실행 시 main( ) 메소드가 Static 영역에 생성되지 않는다면 JVM이 프로그램을 실행하지 못한다.

2.3 Primitive 데이터 타입

자바에서 제공하는 데이터 타입은 크게 'Primitive 데이터 타입'과 'Reference 데이터 타입'으로 구분된다. Primitive 데이터 타입은 int나 boolean처럼 객체가 아닌 순수한 데이터 타입이다. 반면 Reference 데이터 타입은 여덟 개의 Primitive 데이터 타입을 제외한 모든 객체 데이터 타입이다.

표 2-2 자바에서 제공하는 Primitive 데이터 타입

분 류		데이터 타입
논리형		boolean
문자형		char
정수	**수치형**	byte
	수치형	short
	수치형	int
	수치형	long
실수	**수치형**	float
	수치형	double

표 2-2에서 보듯 자바는 다양한 종류의 Primitive 데이터 타입을 제공한다. 자바는 모든 것이 객체로 이루어져 있어서, Primitive 데이터 타입도 객체로 변환할 수 있다. Primitive 데이터 타입을 객체로 변환하려면, 각 타입에 해당하는 Wrapper 클래스를 이용해야 한다.

표 2-3 자바에서 제공하는 Wrapper 클래스

Primitive 데이터 타입	Wrapper 클래스
boolean	java.lang.Boolean
char	java.lang.Character
byte	java.lang.Byte
short	java.lang.Short
int	java.lang.Integer
long	java.lang.Long
float	java.lang.Float
double	java.lang.Double

2.4 import와 package

자바의 모든 클래스는 패키지package라는 고유 주소를 가지고 있다. 일반적으로 클래스의 고유 주소를 나타내기 위해서 "패키지명+클래스명" 형태로 표기한다.

Hello World 프로그램의 패키지명은 'kr.or.javacafe'이고 클래스명은 'Hello World'이므로, Hello World 프로그램을 나타내는 고유 주소는 'kr.or.javacafe.Hello World'가 된다. 이렇듯 모든 클래스는 고유 패키지명을 가지고 있다.

클래스 내부에서 다른 클래스의 위치를 표현할 때는 'import' 키워드를 사용한다. 즉, import 키워드는 외부에 있는 클래스의 위치를 알려주는 역할을 한다.

다시 Hello World 프로그램으로 돌아가자. main( ) 메소드는 파라미터로 String 배열 형태의 인자Argument 값을 받는다. String 객체는 자바에서 제공하는 대표적인 Reference Data Type의 객체다. 이 String 객체를 사용하려면, String 클래스가

있는 곳의 위치를 클래스 내부에 알려야 한다. 하지만 Hello World 프로그램에는 String 객체가 import 키워드로 정의되어 있지 않다. 그렇다면 Hello World 프로그램은 String 객체의 패키지명을 어떻게 알고 있는 것일까?

사실 자바의 모든 클래스에는 java.lang 패키지가 기본적으로 포함되어 있다. java.lang 패키지는 자바의 기본적인 클래스를 모아둔 패키지로서, 이 패키지 안에 String 객체가 포함되어 있다. 즉, Hello World 프로그램 상단에는 "import java.lang.*;"라는 import 구문이 생략되어 있는 것이다.

2.5 생성자

new 키워드를 이용하여 클래스를 객체로 생성할 때 자동으로 호출되는 특이한 메소드가 있다. 이 특이한 메소드를 생성자Constructor라고 한다. 생성자는 일종의 메소드지만 일반적인 것과는 다른 몇 가지 제약 사항이 있다.

생성자 명은 클래스명과 동일해야 하고 리턴 타입이 없어야 한다. 또한 접근 제한자Access Modifier가 반드시 public 타입이어야 한다.

모든 클래스는 생성자를 가지고 있다. 앞서 작성한 Hello World 프로그램을 다시 한 번 살펴보자. 분명 모든 클래스는 생성자를 가지고 있다고 했는데, Hello World 프로그램을 아무리 살펴봐도 생성자가 보이지 않는다. 어떻게 된 일일까?

자바 컴파일러는 컴파일 시 문법 오류를 체크하는데, 이때 생성자 여부를 검사하여 생성자가 없는 클래스에는 Default 생성자를 생성한다. 즉, Hello World 프로그램은 별도의 생성자가 없어서 컴파일 시 컴파일러가 Default 생성자를 만들어준 것이다. 컴파일러가 생성하는 가장 기본적인 생성자는 다음과 같다.

```
public HelloWorld( ) { }
```

앞서 설명한 바와 같이 생성자 명은 클래스명과 동일하고 리턴 타입이 없으며 접근

제한자는 'public'이다. 일반적으로 생성자에서는 객체를 초기화하는 코드를 작성한다. 하지만 항상 생성자를 사용하지는 않기 때문에 매번 코드상에 Default 생성자를 작성하는 것은 굉장히 비효율적이다. 이러한 불편을 해결하기 위해 자바 컴파일러는 좀 더 프로그래머 지향적으로 설계되어 있으며, 여러 가지 기계적인 자동화를 수행한다.

Quick_ 레퍼런스 가이드

- The Java Language Specification
  http://docs.oracle.com/javase/specs/jls/se5.0/html/j3TOC.html

3 | 빠르게 살펴보는 객체 지향 프로그래밍

최근 자바가 보편화되면서 객체 지향 프로그래밍OOP, Object Oriented Programming에 대한 관심도 덩달아 높아졌다. 객체 지향 프로그래밍이란 'C 언어의 구조적인 프로그래밍 방식'을 탈피하여 사람이 생각하는 방식에 좀 더 가깝게 진화한 프로그래밍 방식이다.

자바는 처음부터 OOP를 고려하여 만들어진 언어다. 객체 지향의 기본이라 할 수 있는 클래스로 구성되어 있으며 클래스 간의 다양한 관계를 이용하여 좀 더 직관적이고 쉬운 프로그램을 작성할 수 있다. 객체 지향에 대한 개념을 한마디로 정의하기란 굉장히 어려운 일이다. OOP를 구성하고 있는 다양한 객체 지향적인 개념과 상속Inheritance에 대한 이해를 바탕으로 개념을 정리해 나가는 접근법이 필요하다. 이번 장에서는 그러한 OOP의 핵심 개념을 정리해볼 것이다.

3.1 OOP의 네 가지 주요 특징

"자바는 객체 지향 언어다"라는 말을 많이 들어보았을 것이다. 그만큼 자바라는 언어가 객체 지향 개념을 충실하게 풀어낼 수 있는 강력한 도구라는 의미다. 일반적으로 C 언어와 자바는 객체 지향적인 관점에서 자주 비교된다. C 언어를 이용할 때 주로 사용하던 함수 기반의 프로그래밍 패러다임은 자바로 넘어오면서 OOP 개념에 의해 객체 지향적으로 진화했다.

따라서 자바를 이해하려면 반드시 객체 지향 개념을 이해해야 한다. C 언어를 배울 때 C 언어의 포인터Pointer 개념을 배우며 한 번쯤 좌절하듯, 대부분은 자바의 OOP 개념을 배우면서 좌절한다. 하지만 기본부터 차근차근 밟아간다면, 결코 힘들지만은 않을 것이다. 지금부터 객체 지향 개념의 핵심이라 할 수 있는 OOP의 주요 특징 네 가지에 대해 알아보자.

여기서 잠깐_ 객체 지향의 주요 특징 네 가지

① 상속성Inheritance

② 캡슐화Encapsulation

③ 추상화Abstraction

④ 다형성Polymorphism

3.2 상속성

상속Inheritance은 객체 지향의 꽃이라 불리는 개념으로, 객체 지향의 주요 특징 중에서도 단연 핵심이라 할 수 있다. 상속이라는 개념에 의해 OOP라는 패러다임이 시작되었다고 해도 과언이 아니다. 상속이란 상위 클래스(부모 클래스)의 특징을 하위 클래스(자식 클래스)가 모두 물려받는 것을 말한다. 이때 자식 클래스는 부모가 가지고 있는 멤버 변수Member Variable와 메소드를 모두 물려받아 사용할 수 있다.

상속 관계에 의해 클래스 간에 부모(Parent, Super, Base) 클래스의 개념과 자식(Child, Sub, Derived) 클래스의 개념이 생성되었다. 이들 간의 관계를 일컬어 "계층구조Hiearachy가 형성되었다" 또는 "상속 관계에 있다"라고 말한다.

간단한 예제를 살펴보자. 소스 코드 3-1은 채널 조정 기능이 있는 간단한 TV 프로그램이다.

소스 코드 3-1 TV.java

```
package kr.or.javacafe.sample1;

public class TV {

  int channel;
```

```
    // 디폴트 생성자
    public TV() {}

    // 생성자
    public TV(int intValue) {
       this.channel = intValue;
    }

    public void channelUp() {
       channel = channel + 1;
    }

    public void channelDown() {
       channel = channel - 1;
       if (channel < 0) {
          channel = 0;
       }
    }

    public static void main(String args[]) {
       TV objTV = new TV(11);
       System.out.println("현재 채널은 " + objTV.channel + "번입니다.");
    }
}
```

실행 결과

현재 채널은 11번입니다.

이 프로그램에 브랜드 이름을 추가하고 싶다면 상속을 이용해서 간단히 구현할 수 있다.

소스 코드 3-2 BrandTV.java

```
package kr.or.javacafe.sample1;

public class BrandTV extends TV {

   String brand;

   public BrandTV(int intValue, String strValue) {
      // 부모가 가지고 있는 channel 변수에 값을 할당한다.
      super.channel = intValue;
      this.brand = strValue;
   }

   public static void main(String args[]) {
      BrandTV objTV = new BrandTV(11, "삼성TV");
      objTV.channelUp();
      System.out.println(objTV.brand + " 현재 채널은 " + objTV.channel
         + "번입니다.");
   }
}
```

실행 결과

삼성TV 현재 채널은 11번입니다.

소스 코드 3-2에서 보는 바와 같이 extends 키워드를 이용하여 TV 클래스를 상속받으면, TV 클래스에 있는 channel이라는 멤버 변수와 channelUp( ), channelDown( ) 메소드를 그대로 사용할 수 있다.

정리하자면, 상속이란 다른 클래스의 멤버 변수나 메소드를 자신의 것처럼 쓸

수 있는 특성을 말한다. 상속에 의해 메소드의 오버로딩Overloading과 오버라이딩Overriding 개념, this와 super 키워드 등 OOP를 지원하기 위한 다양한 문법들이 생겨났다.

3.3 캡슐화

캡슐화Encapsulation란 '실제 기능'은 숨기고 '접근할 방법'만 노출하는 것을 말한다. 개념적인 측면으로 살펴보면 여기저기 흩어져 있는 복잡한 기능이 어떻게 구현되고 있는지는 전혀 관심을 두지 않고, 단순히 어떻게 접근하여 사용하는 것인지만 알고 싶어 한다는 개념이다.

캡슐화의 대표적인 예는 '라이브러리'다. 개발자들 대부분은 라이브러리를 어떻게 사용하는지에만 관심을 둘 뿐, 라이브러리의 내부 구조가 어떻게 되어 있는지는 관심이 거의 없다. 이처럼 세부적인 구현은 감추고 외부에서 사용할 수 있도록 최소한의 정보만 노출하는 것이 캡슐화다.

캡슐화는 또한, 프로그램의 복잡도를 줄이는 데도 사용된다. 캡슐화를 이용하여 알 필요가 없는 정보를 숨겨 프로그램의 복잡도를 제어할 수 있는 것이다. 이러한 정보의 노출 여부를 은닉성Hidden이라고 한다.

자바는 캡슐화를 위해 default, public, private, protected 등 네 가지 타입의 접근 제한자Access Modifier를 제공한다. 멤버 변수나 메소드를 선언할 때, 접근 제한자를 이용하여 해당 멤버 변수나 메소드의 접근을 제한할 수 있다.

여기서 잠깐_ 캡슐화 개념의 범위 확장

private에서 public으로 갈수록 접근 가능한 범위가 커진다.

- private → default → protected → public