객체지향 프로그래밍 언어

1.객체지향이란 현실 세계의 개체 모델을 바탕으로 프로그램을 구조화하고 개발하는 프로그래밍 기법을 말한다.
2.함수와 메서드(Method)가 비슷하며, 기본적으로는 객체 간의 메시지 교환을 이용한 프로그래밍 모델이다.
3.상속, 추상화, 캡슐화 등 특징을 활요한 구조적이고 재활용 가능한 모듈화로 생산성과 유지보수 효율성을 높인 프로그래밍 언어이다. 대표적으로 스몰토크(SmallTalk), C++, C#, Net, 자바, 파이썬(Python) 등이 있다.

객체

객체를 뜻하는 'Object'는 기본적으로는 '사물'이라고 해석된다. 사물은 우리 눈에 보이는 모든 것을 말하며, 그 사물이 가진 속성(Attribute)과 행위(Behavior)로 설명할 수 있는 대상이다.

클래스와 인스턴스

1.속성이 같은 객체들을 대표할 수 있는 대상을 클래스(Class)라고 한다. 예를 들어 소나무는 클래스가 되고, 주변에 보이는 각 소나무들은 모두 소나무 클래스의 인스턴스(객체)가 되는 것이다.
2.클래스는 객체를 정의하는 틀이며, 필드(속성)와 메서드(행위)로 구성한다. 클래스는 추상화를 이용하여 슈퍼 클래스(상위 클래스, 부모 클래스)와 서브 클래스(하위 클래스, 자식 클래스)로 구분한다.
3.인스턴스는 클래스에서 생성한 객체로, 고유한 상태가 있다. 동일한 클래스에서 생성된 객체라 할지라도 필드 값과 메서드의 내용은 다를 수 있다.

상속

1.상속은 어떤 클래스에서 좀 더 내용이 구체적인 새로운 클래스가 필요할 때 기존 클래스를 물려받아 새로운 부분만 추가하거나 수정하려고 만든 개념이다.
2.상속을 이용하면 슈퍼 클래스의 기본 구성 요소(필드, 메서드)를 물려받으며(상속), 자신만의 필드나 메서드를 추가하여 구체화하는 것이 가능하다. 물론 물려받은 메서드의 내용을 수정하는 것도 가능하다.
3.상속은 클래스를 선언할 때 extends 키워드를 사용하여 정의한다.

객체지향 프로그래밍의 주요 특징

1.캡슐화: 생성한 객체를 어떤 메서드와 필드로 어떻게 일을 수행할지 외부에 숨기는 특성을 말한다.
2.상속: 클래스는 추상화된 슈퍼 클래스와 구체화된 서브 클래스로 구성한다. 그리고 슈퍼 클래스와 서브 클래스의 고나계를 상속이라고 한다. 자바에서는 다른 객체지향 언어와 달리 다중 상속을 지원하지 않는다.
3.다형성: 클래스의 상속 관계를 이용하여 슈퍼 클래스가 같은 서버 클래스들이 동일한 요청을 다르게 처리할 수 있는 특징을 말한다. 실제 프로그램에서는 메서드를 오버라이딩(재정의)하여 구현한다.

자바 클래스의 기본 구조

1. 패키지: 자바 클래스들을 같은 성격으로 묶어서 관리하는 일종의 디렉터리 개념이다. 역 도메인(Reverse Domain) 방식으로 이름을 부여한다.
2. 클래스: 자바 프로그램은 기본적으로 클래스를 만드는 것에서부터 시작한다. 클래스는 자바 프로그램의 기본 단위이며, 프로그램 실행을 위한 main() 메서드를 반드시 클래스에 포함해야 하는 것으 아니다.
3. 인스턴스: 객체지향의 개념에 따라 클래스는 바로 사용할 수 없고, 인스턴스로 사용해야 한다. main() 메서드에서는 클래스의 인스턴스를 만들고, 인스턴스를 이용하여 메서드 호출 등 필요한 작업을 처리한다.
4. 참조 변수: 자바 클래스의 인스턴스 참조를 위한 변수이다. 구조는 일반적인 변수와 유사하지만, 참조 변수에는 데이터가 없고 클래스 인스턴스를 가리킨다.

5. 생성자: 클래스를 생성할 때 제일 먼저 실행되는 특수한 형태의 메서드로, 리턴값이 없으면 메서드 이름은 반드시 클래스 이름과 일치해야 한다.?리턴값이 없다. 따라서 void를 비롯한 리턴값의 데이터형을 입력하면 안 된다.
-파라미터가 있는 다른 여러 생성자를 정의할 수 있다(메서드 오버로딩).
-파라미터가 없는 기본 생성자(Default Constructor)는 특별히 생성자에서 처리해야 하는 일이 없다면 굳이 프로그램에서 구현할 필요가 없다. 다만 생성자를 여러 개 정의할 때는 반드시 기본 생성자를 명시해야 기본 생성자로 객체 생성이 가능하다.

접근 한정자의 종류와 접근 범위

한정자	클래스 내부	동일 패키지	하위 클래스	그 외의 영역
public	O	O	O	O
protected	O	O	O	X
default	O	O	X	X
private	O	X	X	X

일반 한정자의 종류

static: 클래스 메서드와 클래스 변수를 선언하는 데 사용한다. static은 자바의 정적 영역에 할당되는 리소스를 선언하는 데 사용하며, 동일 가상머신상에서 실행 중인 모든 클래스에서 공유한다. 인스턴스를 생성하지 않고도 클래스의 메서드나 멤버에 접근할 수 있다.
final: 더 이상 변경할 수 없도록 선언하는 한정자. 클래스에 사용할 때는 서브 클래스를 만들 수 없고, 메서드에 사용할 때는 오버라이딩을 할 수 없다. 변수에 사용하면 변수에 저장된 값이 변할 수 없으므로 변수가 아닌 상수의 역할을 한다.
abstract: 추상 클래스를 선언하는 데 사용하는 한정자.
synchronized/volatile: 스레드 프로그래밍에서 여러 스레드가 동시에 자원에 접근할 때 발생하는 데이터 동기화를 처리하는 한정자.

인스턴스 변수와 클래스 변수

1.인스턴스 변수는 클래스의 인스턴스로만 접근 가능한 변수를 말하며, 일반적인 멤버 변수가 여기에 속한다. 클래스 외부에서 접근을 차단하려고 private 키워드를 사용하기도 한다.

2.동일 클래스의 인스턴스라 해도 각 인스턴스 변수는 값이 서로 다르고 서로에 영향을 주지 않는다.

3.클래스 변수는 모든 클래스의 인스턴스로 공유되는 변수로, static 키워드를 사용하여 선언한다. 클래스가 인스턴스되기 전(main() 메서드 시작 부분)에는 인스턴스 변수에 접근할 수 없으므로 main()에서 멤버 변수에 바로 접근하는 것은 불가능하다.

자바 가상머신의 메모리 구조

Method 영역	Heap 영역	Stack 영역
메서드 바이트코드	자바 객체	메서드 파라미터
클래스(static) 변수	인스턴스 변수	지역 변수

추상 클래스와 인터페이스

1.추상 클래스와 인터페이스는 객체지향 개념을 실제 프로그램 개발에 쉽게 적용하고 유연한 설계를 지원하는 요소이다.
2.일반 클래스보다 추상적인 관점에서 접근한다. 구체적인 내용보다는 기본적인 속성과 필요한 메서드의 형태(프로토타입)만 기술하고 세부적인 구현은 구현하는 클래스에서 담당하도록 하는 형태이다.

추상 클래스

추상 메서드(구현되지 않고 정의만 한 메서드)를 하나 이상 포함한다. 추상 메서드가 포함된 클래스는 반드시 추상 클래스로 정의해야 한다.
-추상 클래스는 일반 클래스와 같이 멤버 변수 및 메서드를 포함할 수 있다.
-추상 클래스는 그 자체를 인스턴스화, 즉 객체 생성에 사용할 수 없으며, 반드시 추상 클래스를 상속받는 클래스를 만든 후 추상 클래스에 선언된 모든 추상 메서드를 오버라이딩해서 구현해야 한다.

인터페이스

인터페이스는 일반 메서드를 포함할 수 없으며, 모두 추상 메서드(public abstract)로만 구성해야 한다.
일반 멤버 필드는 없고, public, static, final로 선언한 상수만 있다.
추상 클래스와 마찬가지로 직접 객체를 생성하는 것은 불가능하고(Anonymous Inner Class 생성 형태로는 가능), 다른 클래스로 구현할 때는 implements 키워드로 구현을 선언해야 한다.
클래스 상속과는 별도로 동작하므로 문법상 다중 상속을 지원하지 않는 자바에서 다중 상속의 개념을 지원하는 형태로 사용하는 것이 가능하다.

메서드 오버로딩과 메서드 오버라이딩

1.메서드 오버로딩과 메서드 오버라이딩은 자바에서 다형성을 지원하는 구현 형태 중 하나이다.
2.메서드 오버로딩은 메서드 이름은 동일하지만 파라미터가 다른 여러 메서드를 만드는 것을 말한다.
3.메서드 오버로딩의 조건은 다음과 같다.?메서드 이름이 같아야 한다.
-파라미터의 개수 또는 타입이 달라야 한다.
-파라미터는 같고 리턴 타입이 다르면 오버로딩이 성립되지 않는다.
4.메서드 오버라이딩은 슈퍼 클래스에서 정의한 메서드를 서브 클래스에서 재정의하는 것을 말한다.
5.메서드 오버라이딩의 조건은 다음과 같다.?오버라이딩하는 메서드는 슈퍼 클래스의 메서드와 메서드 구성 요소 모두가 동일해야 한다.
이때 이름, 파라미터, 리턴 타입이 모두 같아야 한다.

6. 자바 기본 라이브러리 활용

자바 API

1.API는 Application Programming Interface의 약어로, 응용 프로그램 개발에 사용할 수 있는 라이브러리 규격을 말한다.

2.일반적인 자바 프로그램에서는 Java SE API를 주로 사용하며, 개발에 주로 활용하는 유용한 패키지들은 다음 표와 같다.

주요 기능	설명	주요 패키지/클래스
lang and util	자바 언어의 주요 구성 요소와 관련된 패키지	java.lang, java.util
Math	각종 수학 계산과 관련된 클래스 및 유틸리티	java.math, java.lang.Math
Collections	다양한 자료구조를 동일한 방법으로 처리할 수 있도록 하는 라이브러리	java.util
Input/Output	키보드, 네트워크, 파일 등 입출력 관련 주요 기능을 제공하는 라이브러리	java.io, java.nio, java.net
Date and Time	시간 및 날짜 처리 관련 라이브러리	java.util, java.text
Networking	네트워크 통신 프로그래밍 관련 라이브러리	java.net
Swing/AWT	GUI 구현 관련 라이브러리	java.swing, java.awt, javax.swing
JDBC	데이터베이스 프로그래밍 관련 라이브러리	java.sql

java.lang 패키지의 주요 클래스

1.java.lang은 자바 언어의 가장 기본이 되는 클래스를 모아 둔 패키지이다.

2.String 클래스는 문자열 처리를 위한 클래스로, 자바에서 문자열은 기본 자료형이 아니라 클래스를 사용하는 객체 타입이다.

3.문자열을 비교할 때는 '=='이 아닌 equals() 메서드를 사용한다.

4.더하기 연산을 수행하면 문자열 결합이 가능하다. 그러나 자바 문자열 특성상 기존 데이터를 대체하는 것이 아니라 계속 새로운 인스턴스를 생성하기 때문에 성능에 문제가 발생할 수 있으므로 주의해야 한다.

5.StringBuffer 클래스는 성능에 영향을 받지 않고 문자열 처리를 하는 클래스이다. 긴 문자열 조합은 문자열 더하기 대신 StringBuffer를 사용하는 것이 좋다.

6.Wrapper 클래스는 자바의 기본 자료형을 객체 타입으로 처리할 수 있도록 만든 클래스이다. 기본 자료형에 대응하는 클래스들이 모두 준비되어 있다. 각 Wrapper 클래스에는 해당 데이터 타입과 관련된 유용한 기능들이 제공된다.

7.System 클래스는 자바 프로그램을 실행하는 것과 관련된 유용한 기능들을 제공한다.

java.util 패키지의 주요 클래스

1.java.util은 프로그램을 개발할 때 유용한 기능을들 모아 놓은 유틸리티 클래스 패키지이다.

2.날짜 처리는 프로그램을 개발할 때 중요하면서도 생각보다 손이 많이 가는 작업이다. java.util.Data, java.util.Calendar 클래스의 주요 메서드와 필드 등을 잘 활용해야 한다.

3.Calendar 클래스는 추상 클래스라 인스턴스 생성이 안 되며, getInstance() 메서드를 사용해야 한다.

4.날짜 형식 변환은 java.text의 클래스를 사용해야 한다.

5.Scanner 클래스는 입력 스트림으로, 데이터를 입력받는 클래스이다. 고급 입출력은 자바 I/O에서 제공하는 클래스들을 사용해야 하며, Scanner 클래스는 상대적으로 간단한 입력을 처리하는 데 사용한다. 객체를 생성할 때 입력 스트림을 변경하면 키보드 외의 파일, 네트워크 등에서 데이터 입력을 받아올 수 있다.
6.Random 클래스는 난수를 생성하는 클래스이다. 일련의 규칙에 따라 난수를 생성하는 관계로, 객체를 생성할 때 현재 시간 정보를 시드(Seed)로 하여 더 유용한 난수 생성이 가능하다. 난수 생성 범위를 지정하면 0부터 시작하므로 1부터 시작해야 할 때는 생성된 난수에 1을 더하도록 한다.
7.StringTokenizer 클래스는 구분자로 연결된 문자열을 손쉽게 분리(파싱)하는 클래스이다. 일반적으로 구분자에는 공백, 콤마, 탭 등을 많이 사용한다. 특히 콤마로 구분된 데이터는 csv 포맷이라고 하여 엑셀과도 호환된다.

java.text 패키지의 주요 클래스

1.java.text 패키지는 주로 문자 형태로 구성된 정보의 변환을 지원하는 클래스들로 구성되어 있다.
2.날짜 형식을 지정하는 클래스로 DateFormat과 SimpleDateFormat이 있으며, 자바 8에는 java.time API가 새롭게 추가되었다.
3.DateForamt은 추상 클래스로 getInstance()를 사용해야 하며, 형식 제한이 있고 다양한 형식 지정을 원할 때는 SimpleDateFormat 클래스를 사용해야 한다. 날짜 지정 형식은 yyyy.MM.dd hh:mm과 같은 형식 지정 문자를 조합해서 원하는 대로 할 수 있다.
4.숫자 형식 지정은 NumberFormat과 DecimalFormat 클래스를 사용할 수 있다. 날짜와 함께 숫자 역시 프로그램을 개발할 때 여러 변환이 필요한 데이터 중 하나이다. 화폐 단위를 비롯하여 자릿수 지정 등 숫자와 관련된 유용한 기능들을 제공한다.
5.메시지는 문자열로 구성된 의미있는 데이터를 말한다. 프로그램이나 시스템 간에 데이터를 송수신할 때 이용할 수 있고, 규칙화된 문자열 패턴에 데이터를 매핑하는 용도로도 사용한다. MessageFormat 클래스는 패턴과 데이터 조합을 쉽게 연결하여 원하는 문자열을 만들 수 있도록 도와준다.

7. 자바 I/O와 네트워크 프로그래밍

자바 I/O

1.I/O는 Input/Output, 즉 입력과 출력을 말한다. 자바에서는 java.io 패키지로 다양한 입출력 기능을 제공한다.
2.스트림(Stream)은 입출력 장치와 프로그램 간 데이터 전송 통로를 말한다. 스트림은 단방향으로만 뻗어 있으며, 연속된 데이터의 흐름으로 나타난다.
3.입출력 프로그램은 입력 및 출력 장치, 네트워크, 컴퓨터에 유/무선으로 연결된 장치 등을 모두 스트림 기반으로 프로그램하는 것이 가능하다.

바이트 스트림과 문자 스트림

1.바이트 스트림은 바이트 단위로 데이터를 전송하고, 문자 스트림은 2바이트 단위로 문자를 전송한다. 예를 들어 숫자 17을 전송할 때 문자 스트림은 숫자 17 그대로 전송하지만, 바이트 스트림은 16진수 0x11이 바이트 단위 0001 0001로 전송된다는 차이가 있다.
2.문자 스트림은 Reader와 Writer 클래스 계열을 사용하고, 바이트 스트림은 InputStream, OutputStream 클래스 계열을 사용한다.
3.바이트 스트림 중 DataInputStream과 DataOutputStream은 자바의 기본 자료형 처리에 적합한 클래스로, 주로 바이너리 데이터 전송에 많이 사용한다.
4.스트림을 사용한 후에는 항상 close() 메서드를 사용하여 닫아 줘야 다른 프로그램에서 해당 장치와 연결할 수 있다.

자바 파일 입출력

1.파일은 컴퓨터에서 가장 기본이 되는 중요한 입출력 대상이다. 컴퓨터의 특성상 메모리에 저장된 데이터는 컴퓨터 전원이 꺼지면 사라지므로, 모든 프로그램은 어떤 형태로든 파일을 사용한다.
2.디렉터리는 파일을 체계적으로 관리하려는 것으로, 폴더라고도 한다. 디렉터리의 최상위를 루트라고 하며, 유닉스 계열은 '/', MS 윈도우 계열은 '\'를 디렉터리 구분자로 사용한다.
3.경로는 디스크상의 파일이나 디렉터리 위치를 말하는 것으로, 디렉터리 구분자와 함께 사용한다. 유닉스는 /home/user1/data/my.txt와 같은 형식을 사용하고, 윈도우는 C:\home\user1\data\my.txt와 같은 형식을 사용한다.
4.파일 입출력 프로그램은 File 클래스를 사용하여 원하는 파일 작업을 위한 객체를 생성하고, 입출력 스트림을 생성하여 프로그램하는 방식이다.
5.File 클래스는 바이트 스트림과 문자 스트립 계열의 클래스를 모두 지원하며, RandomAccessFile 클래스는 임의의 파일 위치를 읽고 쓸 수 있는 기능을 제공한다.

자바 네트워크 프로그래밍

1.네트워크는 컴퓨터와 컴퓨터를 연결한 망형 조직을 말하며, 기업 내 컴퓨터를 연결한 랜(LAN)에서부터 전 세계가 하나로 연결된 인터넷까지 모두 네트워크라고 할 수 있다.
2.네트워크의 기술적 유형에는 여러 가지가 있었지만, 지금은 TCP/IP라는 통신 프로토콜을 사용하는 컴퓨터 네트워크가 가장 일반적인 인터넷의 기반이 되었다.
3.프로토콜은 컴퓨터와 컴퓨터가 데이터를 주고받는 규약을 말하며, 물리적 레벨에서 응용 프로그램 레벨까지 다양한 프로토콜이 있다.
4.네트워크 프로그램은 데이터를 요청하는 클라이언트와 서비스를 제공하는 서버로 구분한다.
5.포트는 서버에서 네트워크 접속을 받아들이는 연결 창구를 말한다. 포트에 따라 서로 다른 여러 서비스를 제공할 수 있다.
6.소켓은 TCP/IP 네트워크에서 클라이언트와 서버가 통신하는 연결 통로를 말한다. 서버 입장에서는 클라이언트 연결 하나당 하나의 소켓 객체가 필요하다. 따라서 보통 네트워크 프로그램을 소켓 프로그래밍이라고도 한다.
7.최근에는 URL을 이용하여 웹 서버의 자원에 접근하는 네트워크 프로그램이 늘어나고 있다. 이는 웹의 프로토콜인 HTTP를 이용한 것으로, 기존 웹 프로그램과 웹 사이트에서 운영 중인 정보를 손쉽게 연동할 수 있다는 점 때문에 널리 확산되었다. 자바에서는 URL 클래스로 HTTP 연결을 지원한다.
8.ServerSocket 클래스는 클라이언트 연결을 지원하는 서버 프로그램에서 사용하고, Socket 클래스는 클라이언트 및 서버에서 클라이언트 접속을 처리하려고 사용한다.
9.Socket을 사용하여 입출력 스트림을 확보할 수 있으며, 이후의 프로그램은 스트림을 사용하는 일반 자바 입출력 프로그램과 동일하다.
10.네트워크 프로그램은 클라이언트와 서버가 상호작용을 해야 하므로 프로토콜 개념이 들어가야 하고, 다중 클라이언트 접속을 처리하려면 스레드 프로그램도 해야 하므로 정밀한 네트워크 프로그램 개발에는 많은 경험이 필요하다.