# ABOUT UNIFIED MODELING LANGUAGE

# 통합 모델링 언어 (UML)에 관하여


**Nathan Cho** *조나단 [20425]*
Department of Web Operation
Sunrin Internet High School
Seoul, Korea
dev.bedrock@gmail.com

# 목차

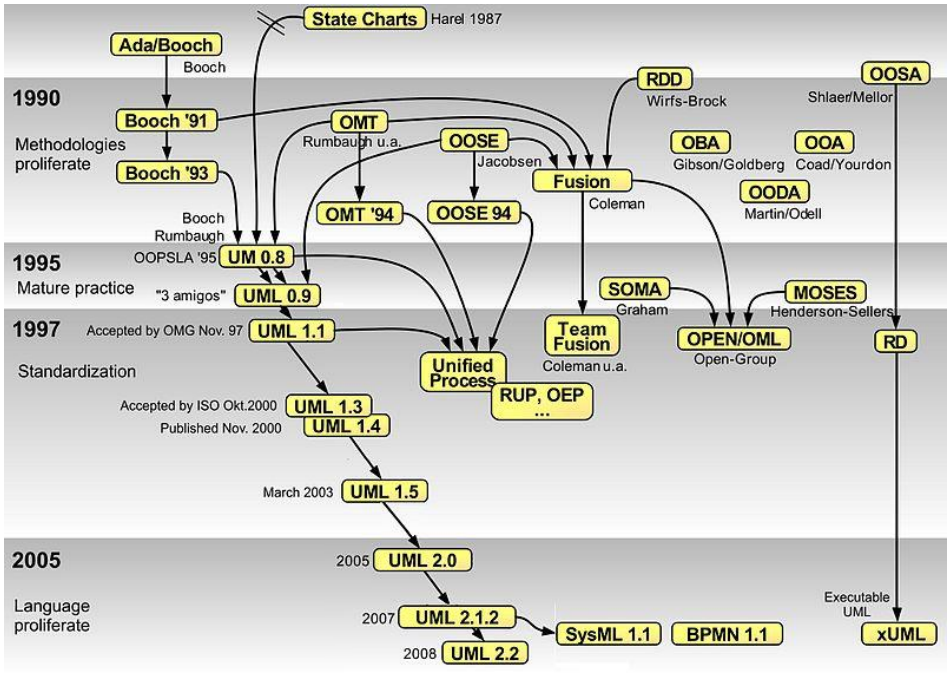
# 1. UML

## 1) UML 이란

UML (Unified Modeling Language, 통합 모델링 언어)은 일반적 사용이 가능한 모델링 언어이다. 시스템을 시각화하는데 표준을 제공하기 위해 제작되었다. UML 은 소프트웨어 공학에서 사용하기 위해 만들어 졌지만 기타 구현 기술의 모든 프로세스에서 사용할 수 있다.

UML 은 1994-1995 년에 소프트웨어 디자인의 표기법 시스템의 표준화를 위해 그래디 부치*(Grady Booch)*, 제임스 럼버*(James Rumbaugh)*, 이바 야콥슨*(Ivar Jacobson)*에 의해 개발되었다. 1997 년에 객체 관리 그룹 *(Object Management Group)*의 표준으로 채택되었고, 그 이후로 객체 관리 그룹의 관리 아래에 있다. 또한 2005 년에 ISO 표준으로 정해 졌다.

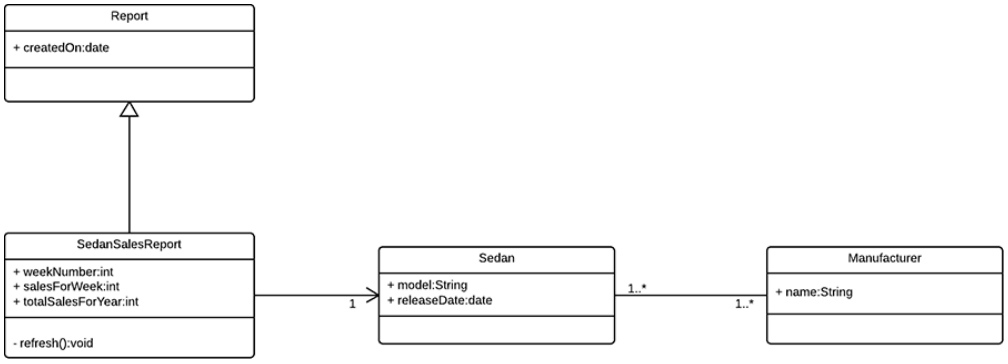
UML 은 1980 년대 후반의 객체 지향 프로그래밍에 뿌리를 두고 1990 년대 후반부터 개선되어 오고 있다. Booch 표기법과 OMT*(object-modeling technique)* 표기법, OOSE*(object-oriented software engineering)* 표기법이 하나의 언어로 합쳐 져서 UML 이 탄생했다. 제작자 3 명의 기술적 관리 아래에서 UML 1.0 버전이 발표되었고 1997 년에 객체 관리 그룹에게 채택되었다. 2005 년에는 UML 2.0 버전이 발표되어 2017 년 6 월 현재 최신 버전은 UML 2.5 이다.



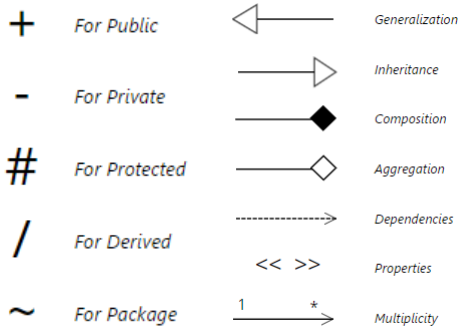

*객체 지향 표기법의 역사(출처: https://en.wikipedia.org/wiki/File:OO_Modeling_languages_history.jpg)*

### 2) UML 의 구조

UML 은 시스템 구조 개요를 다이어그램으로 시각화 할 수 있는 방법을 제공한다. 작업, 시스템 부품, 객체간의 상호 작용, 외부 유저 인터페이스 등의 여러 가지 객체들로 시각화 할 수 있는 표준을 제공한다.




*UML 사용 예시*

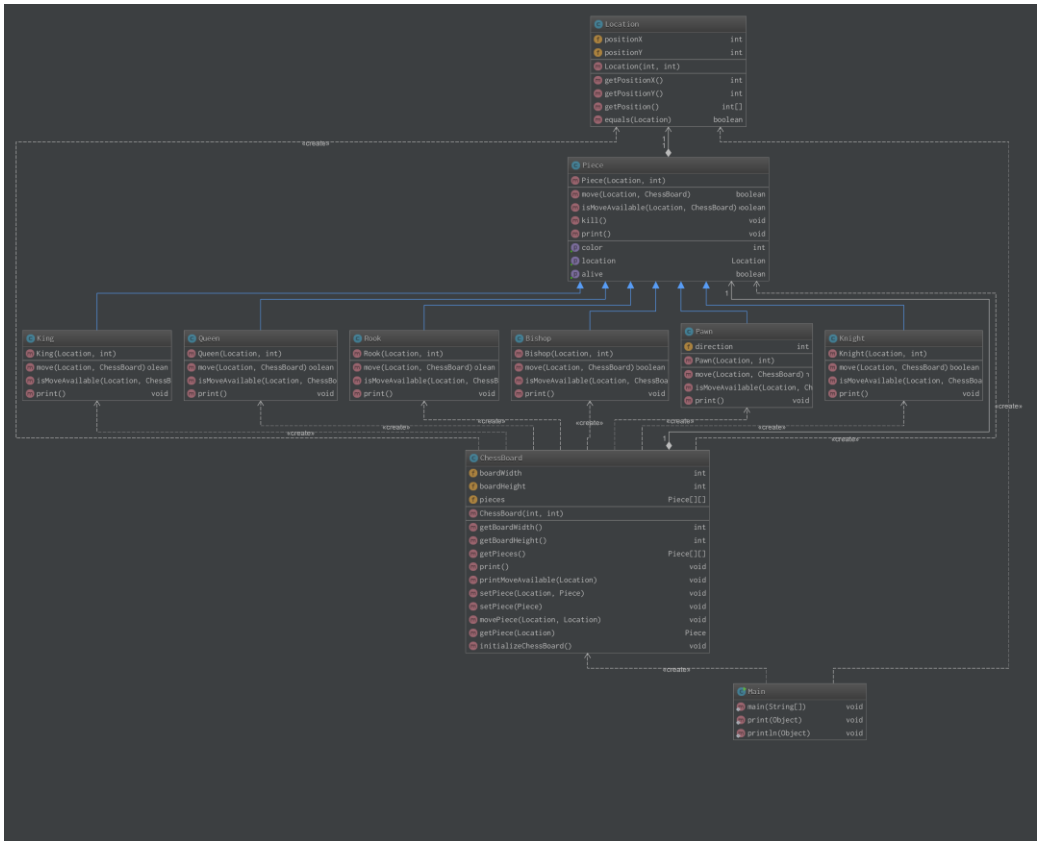



*UML 에서 사용되는 기호들*

## 2. UML 사용 예시

모든 소스는 MIT 라이선스로 GitHub 에서 볼 수 있습니다.

간단한 체스 게임으로 Piece (체스 말) 이라는 객체를 King, Queen, Bishop, Rook, Pawn, Knight (체스 말 종류)가 상속하고 있는 구조입니다.




*UML 은 IntelliJ 플러그 인을 사용하여 출력하였습니다.*

## Piece.java

```java
public class Piece {
    protected Location location;

    private int color;

    private boolean isAlive;

    public Piece(Location location, int color) {
        this.location = location;
        this.color = color;
        this.isAlive = true;
    }

    boolean move(Location targetLocation, ChessBoard chessBoard) {
        if (isMoveAvailable(targetLocation, chessBoard)) {
            chessBoard.movePiece(this.location, targetLocation);
            return true;
        } else {
            return false;
        }
    }

    public Location getLocation() {
        return location;
    }

    public int getColor() {
        return color;
    }

    public boolean isAlive() {
        return isAlive;
    }

    boolean isMoveAvailable(Location targetLocation, ChessBoard chessBoard) {
        return chessBoard.getBoardWidth() >= targetLocation.getPositionX()
                && chessBoard.getBoardHeight() >= targetLocation.getPositionY()
                && !location.equals(targetLocation);
    }

    void kill() { this.isAlive = false; }

    void print() {
        System.out.print("0");
    }
}
```

King.java (Piece 를 상속받음)

```java
public class King extends Piece {
    public King(Location location, int color) { super(location, color); }

    @Override
    boolean move(Location targetLocation, ChessBoard chessBoard) {
        return isMoveAvailable(targetLocation, chessBoard) && super.move(targetLocation, chessBoard);
    }

    @Override
    boolean isMoveAvailable(Location targetLocation, ChessBoard chessBoard) {
        // 1 square to any direction
        int differenceX = Math.abs(targetLocation.getPositionX() - location.getPositionX());
        int differenceY = Math.abs(targetLocation.getPositionY() - location.getPositionY());
        return super.isMoveAvailable(targetLocation, chessBoard)
                && differenceX <= 1
                && differenceY <= 1;
    }

    @Override
    void print() { System.out.print("K"); }
}
```