24.4.26.금_JavaDoc

#JavaDoc이란..
    구조
    JavaDoc 용어
    JavaDoc 생성하는 법
#빌드
    인텔리제이 IDEA 빌드 과정
    터미널을 통한 빌드
    JAR 파일 구동
#JavaDoc
    Polygon [JavaDoc 활용 01]
    Human [JavaDoc 활용 02]
    Student [JavaDoc 활용 03]
    JavaDoc [JavaDoc 확인]
#연습문제
    📄 Ex00 []

#JavaDoc이란..

구조

- ```
  /**
   * 요약된 간단한 설명
   *
   * 디테일한 설명...
   * @param 파라미터에 대한 설명
   * @return 반환 값에 대한 설명
   * @throws 예외에 대한 설명
   */
  ```

JavaDoc 용어
  ```

- *@return* : 반환 값
- *@param* : 파라미터
- *@throws 예외* : 예외 처리에 대한 설명
- *@author 이름* : 작성자
- *@version 버전* : 버전
- *@deprecated* : 더 이상 사용하지 않는 메서드에 대한 표시
- *@see #메서드()* : 대신 # 표기한 메서드를 확인하라는 표시
- *@since 버전* : 해당 버전 이후 사용되고 있다는 표시
- *{@inheritDoc}* : 부모 클래스의 메서드를 오버라이딩 했을 경우의 표시

JavaDoc 생성하는 법

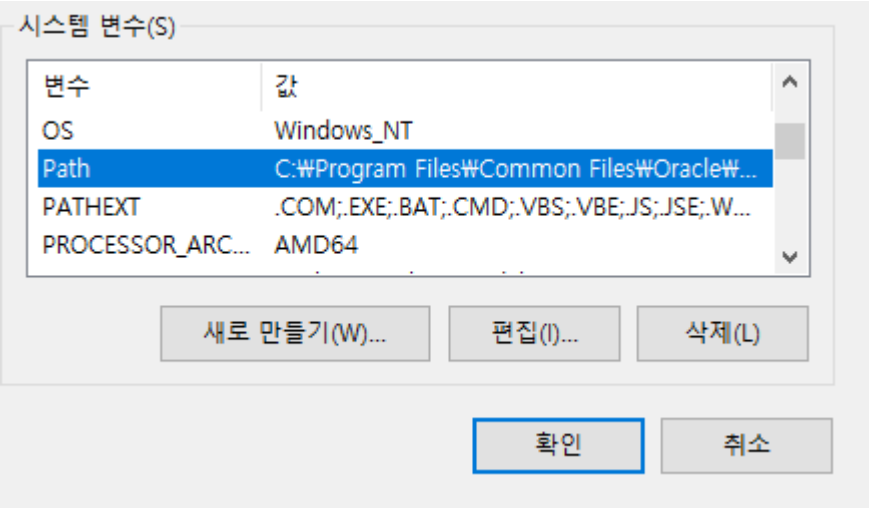
1. <Terminal> 클릭 → > `javadoc` 입력
    - <error: No modules, packages or classes specified.
      1 error> 대신 빨간 줄이 떴을 경우

      a. C:\Program Files\Java\jdk-21\bin

      b. 시작 메뉴에 <시스템 환경 변수 편집> 클릭

      c. 하단에 <환경 변수> 클릭

      d. 하단에 <시스템 변수>에서 <Path> 클릭

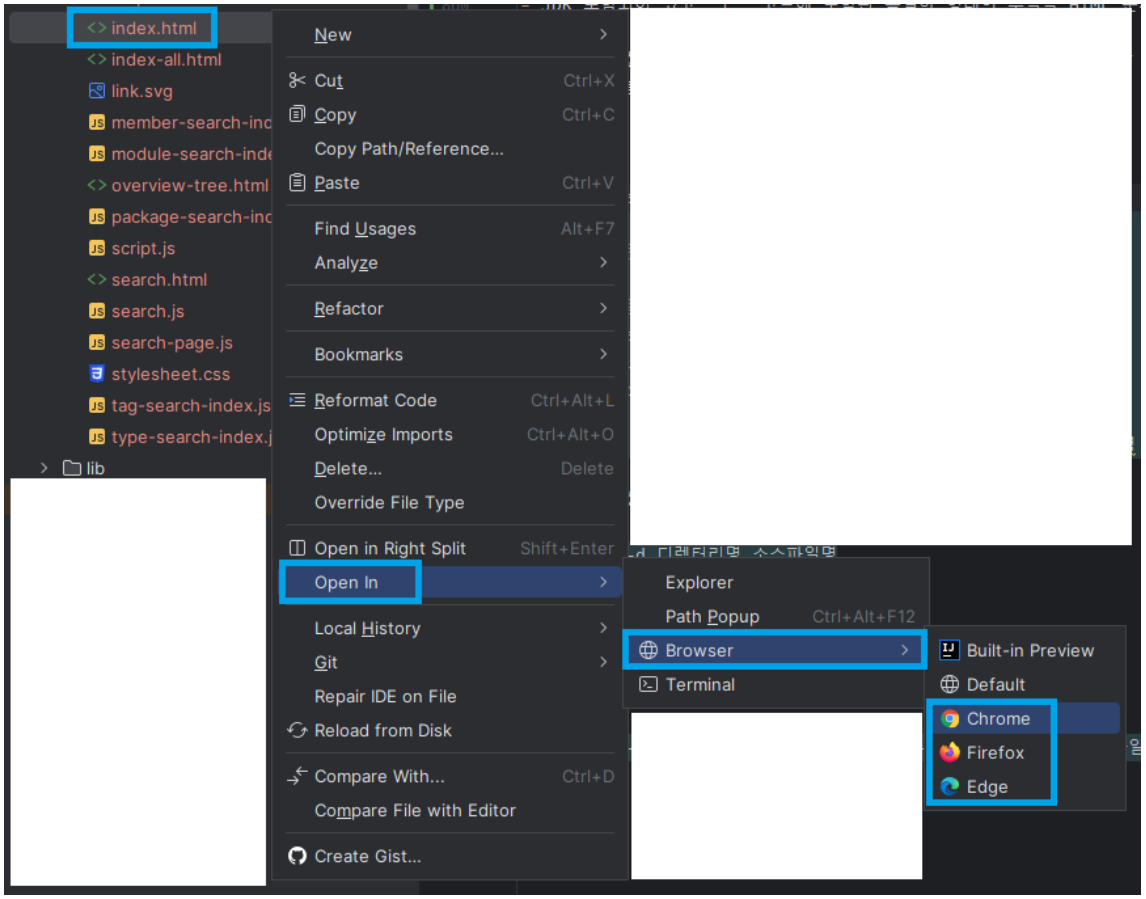


      e. C:\Program Files\Java\jdk-21\bin 추가

f. <확인> 클릭

2. > `javadoc -encoding UTF-8 -charset UTF-8 -d 디렉터리명 소스파일명` 입력
   - 예시) > `javadoc -encoding UTF-8 -charset UTF-8 -d docs src/javadoc/Polygon.java`
3. > `docs` 파일 생성 확인

4. 아래와 같이 실행



#빌드

인텔리제이 IDEA 빌드 과정

- **기존 프로젝트에서 ~**
  1. 상단 <Build> → <Build Project> → <<src>> 위에 <<out>> 디렉터리 생성 확인
  2. 상단 <File> → <Project Structure> → <Artifacts> → <+> → <JAR> → <From modules with ~> → <Apply> → <OK>
  3. 다시 상단 <Build> → <Build Artifacts> → <Build Artifact : ~.jar> → <Action : Build>
  4. <<out>> 디렉터리 안에 <<artifacts>> 디렉터리 생성 및 ~.jar 파일 생성 확인
  5. ~.jar 파일 복사
- **다른 프로젝트로 ~**
  1. <<lib>> 디렉터리 생성 → 그 안에 ~.jar 파일 붙여넣기
  2. ~.jar 파일(혹은 <<lib>> 디렉터리) 우 클릭 → <Add as Library>
- **만약! 다른 프로젝트로 복사할 때 디렉터리를 생성 안 하고 사용하고 싶다면…**
  1. <File> → <Project Structure> → <Library> → <+> → <java> → 앞서 ~.jar 파일을 찾아 클릭 → <Apply> → <OK>

터미널을 통한 빌드

- **기존 프로젝트에서 ~**
  1. 상단 <Build> → <Build Project> → <<src>> 위에 <<out>> 디렉터리 생성 확인
  2. <Terminal>
     - > `jar cf pet.jar -C .\out\production\JAR_Practice\ com/pet/`
     - > `jar tf pet.jar`
     - > `jar xf pet.jar`
- **다른 프로젝트로 ~**
  1. <File> → <Project Structure> → <Library> → <+> → <java> → 앞서 ~.jar 파일을 찾아 클릭 → <Apply> → <OK>

- ※ 접근 제한자 확인! public이 아닐 경우 접근 불가

JAR 파일 구동

- <Terminal>
  1. > `jar cfe pet.jar com.pet.Main -C .\out\production\JAR_Practice\ com/pet/`
  2. > `java -jar pet.jar`

#JavaDoc

Polygon [JavaDoc 활용 01]

```
package javadoc;

/**
 * 다각형을 나타내는 클래스입니다.
 * 다각형의 각 변의 길이는 동일합니다.
 */
public class Polygon {

    /**
     * 변의 수 입니다.
     */
    int sides;

    /**
     * 변의 개수를 인자로 받는 생성자입니다.
     * @param sides
     */
    public Polygon(int sides) {
        this.sides = sides;
    }

    /**
```

```
     * 내각의 크기를 반환하는 메서드입니다.
     * @return 내각의 크기 (double 타입)
     */
    public double getInnerAngle() {
        return (sides % 2) * 180 / sides;
    }

    /**
     * 각 변의 길이를 입력받아 둘레를 반환하는 메서드입니다.
     * @param sideLength 각 변의 길이
     * @return 둘레
     */
    public int getPerimeter(int sideLength) {
        return sideLength * sides;
    }
}
```

Human [JavaDoc 활용 02]

```
package javadoc;

/**
 * 다각형을 나타내는 클래스입니다.
 * 다각형의 각 변의 길이는 동일합니다.
 */
public class Polygon {

    /**
     * 변의 수 입니다.
     */
    int sides;

    /**
     * 변의 개수를 인자로 받는 생성자입니다.
     * @param sides
     */
    public Polygon(int sides) {
```

```
        this.sides = sides;
    }

    /**
     * 내각의 크기를 반환하는 메서드입니다.
     * @return 내각의 크기 (double 타입)
     */
    public double getInnerAngle() {
        return (sides % 2) * 180 / sides;
    }

    /**
     * 각 변의 길이를 입력받아 둘레를 반환하는 메서드입니다.
     * @param sideLength 각 변의 길이
     * @return 둘레
     */
    public int getPerimeter(int sideLength) {
        return sideLength * sides;
    }
}
```

Student [JavaDoc 활용 03]

```
package javadoc;

/**
 * 학생을 나타내는 클래스
 */
public class Student extends Human {

    /**
     * 학교 이름
     */
    String schoolName;

    /**
     * 사람의 이름과 학교 이름을 추가로 인자로 받는 생성자
```

```
     *
     * @param name 이름
     * @param age 나이
     * @param schoolName 학교명
     */
    public Student(String name, int age, String schoolName) {
        super(name, age);
        this.schoolName = schoolName;
    }

    /**
     * {@inheritDoc} 학교 정보도 추가하여 소개합니다.
     */
    @Override
    public void intro() {
        super.intro();
        System.out.printf("%s 학생입니다.\n", schoolName);
    }

    /**
     * 주어진 시간에 대한 과목 반환 메서드
     *
     * @param period 교과목 시간 (1, 2, 3교시)
     * @return 과목명을 반환
     * @throws ArrayIndexOutOfBoundsException 주어진 시간이 1 ~
     */
    public String getClassFromPeriod(int period) throws Array
        return new String[] {"국어", "영어", "수학"}[period - 1]
    }
}
```

JavaDoc [JavaDoc 확인]

```
package javadoc;

public class JavaDoc {
    public static void main(String[] args) {
```

```
//      JavaDoc 으로 추가한 경우 사용할 때 클래스, 생성자, 메서드 등에서
        Polygon polygon = new Polygon(3);

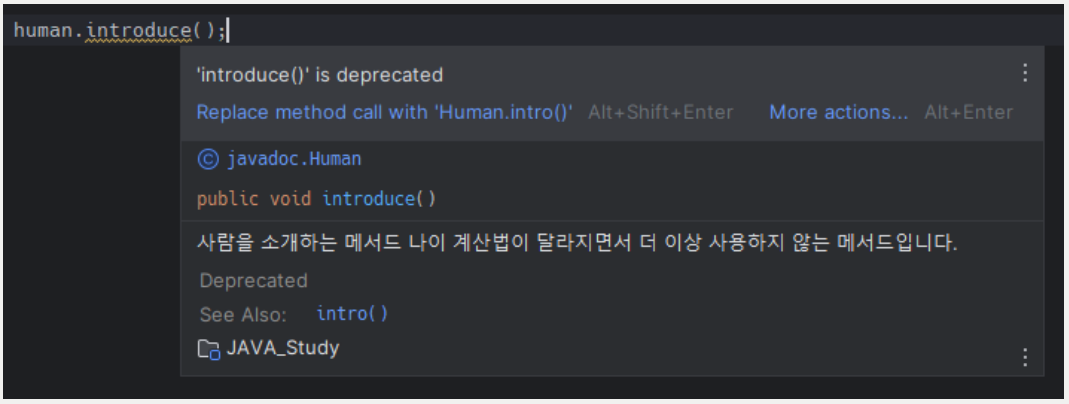
        double angle = polygon.getInnerAngle();
        int perimeter = polygon.getPerimeter(10);

        System.out.println(String.format("삼각형의 내각은 %.0f도
        System.out.println(String.format("삼각형의 둘레는 %d 입니

        Human human = new Human("홍길동", 30);

        human.introduce();  // deprecate 된 메서드
        human.intro();
    }
}
```

출력화면




```
삼각형의 내각은 60도 입니다.
삼각형의 둘레는 30 입니다.
안녕하세요! 저는 홍길동이고, 30세 입니다.
안녕하세요! 저는 홍길동이고, 만 29세 입니다.
```

#연습문제

📄 Ex00 []

```
package javadoc;

public class Ex00 {
    public static void main(String[] args) {

/*
        연습 문제 1. Java 수학 라이브러리 빌드하기

        - 작성한 수학 유틸리티 클래스 2개를 빌드하고 패키징하여 외부에서
            - static0.ex.MathUtil
            - static0.ex.MathArrayUtil

        1. 새로운 프로젝트를 생성하세요.(MathLib)
        2. 위의 소스파일을 해당 프로젝트에 준비하세요.
        3. JavaDoc으로 해당 클래스에 대한 설명을 태그와 함께 작성해 보세
        4. JavaDoc을 사용해 API 문서를 만들어주세요.
        5. 빌드 단계를 거쳐 JAR 파일 아티팩트를 생성해 보세요.
        6. 다른 프로젝트에서 수학유틸리티 라이브러리 클래스를 불러와 사용해

        ------------------------------------------------------

        연습 문제 2. 실행 가능 컬렉션 주소록 관리 앱 빌드하기

        - 컬렉션 프레임워크 주소록 관리 자바 프로그램을 빌드하여 외부에서
            - collection.list.ex.ex3.AddressBook;

        1. 새로운 프로젝트를 생성하세요. (AddressBook)
        2. 위 실행 가능 Java 파일과 관련된 클래스을 해당 프로젝트에 준비
        3. JavaDoc으로 해당 클래스에 대한 설명을 태그와 함께 작성해 보세
        4. JavaDoc을 사용해 API 문서를 만들어주세요.
        5. 빌드 단계를 거쳐 실행가능 JAR 파일 아티팩트를 생성해 보세요.
        6. 생성된 JAR 파일을 다른 경로로 옮겨 실행해 보세요.

        ------------------------------------------------------
```

```
        연습 문제 3. DB 도서 관리 프로그램 앱 빌드하기

        - JDBC로 DB에 접근하여 CRUD를 수행하는 도서관리 프로그램을 빌드
            - package jdbc.ex.book;

        1. 새로운 프로젝트를 생성하세요.(BookManagement)
        2. 위 패키지와 관련된 클래스들을 해당 프로젝트에 준비하세요.
        3. JDBC 드라이버를 외부 라이브러리로 연결하세요.
        4. JavaDoc으로 해당 클래스에 대한 설명을 태그와 함께 작성해 보세
        5. JavaDoc을 사용해 API 문서를 만들어주세요.
        6. 빌드 단계를 거쳐 실행가능 JAR 파일 아티팩트를 생성해 보세요.
        7. 생성된 JAR 파일을 다른 경로로 옮겨 실행해 보세요.
*/
    }
}
```