

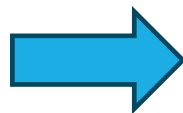
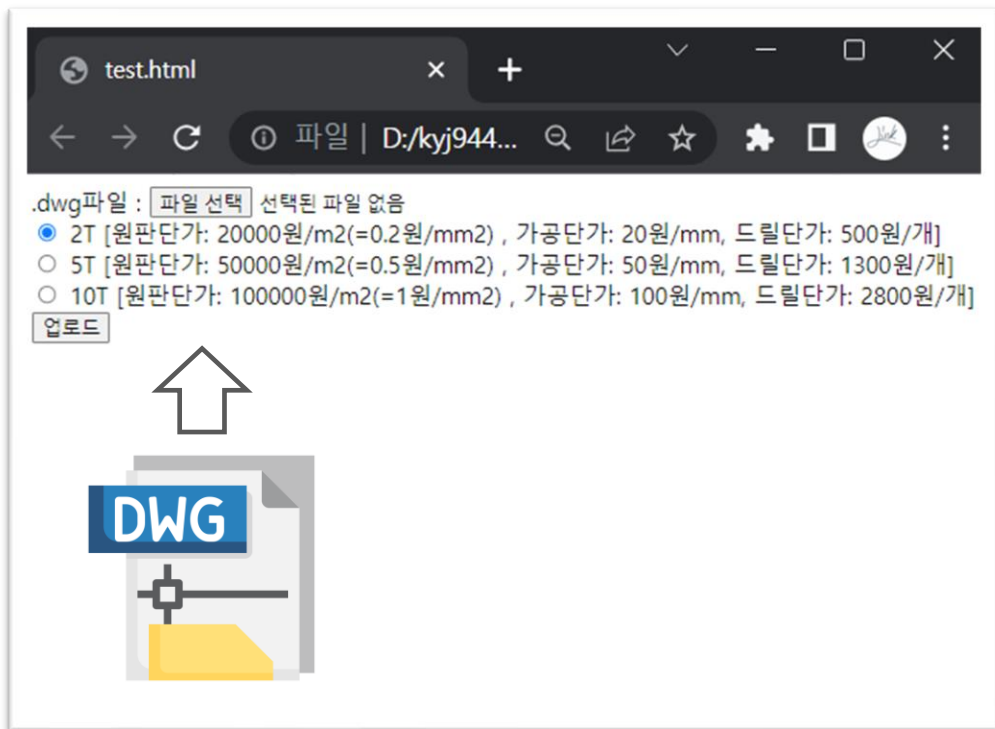


# Java Portfolio

고영진










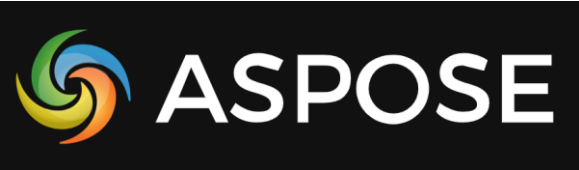
# Java 포트폴리오 기능 요약



1. .dwg파일과 재료 철판의 두께를 선택해 submit하면

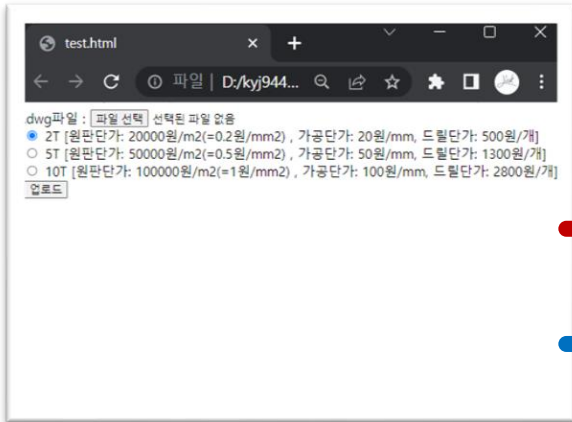
2. 도면을 분석해 가공 비용을 산출하여 그림과 함께 출력

## 사용 Tools & Libraries

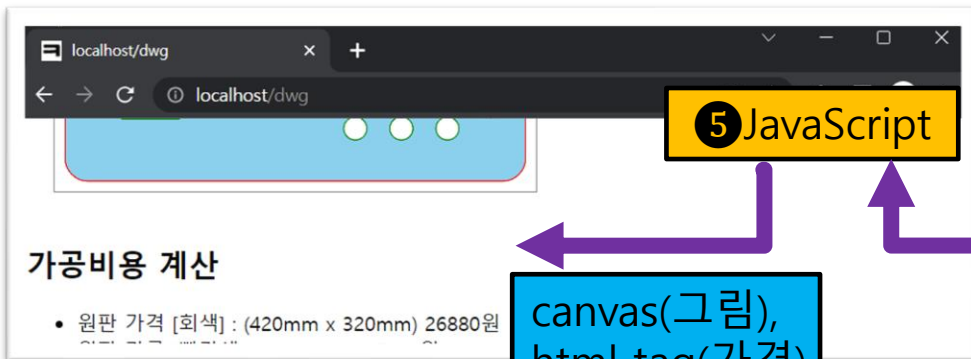
코드 에디터	 Visual Studio Code  IntelliJ IDEA Community Edition
웹 프레임워크	 Spring Boot  Thymeleaf (스크립트 엔진)
개발 언어	 Java  HTML5  JavaScript
외부 라이브러리 (.dwg파일 읽기)	 Aspose.CAD for Java

# 동작 흐름 개요

## 1 시작 (/dwg/input.html)



## 완료 (/dwg/output.html)



POST (/dwg)

.dwg MultipartFile

attribute "thick" value

2 DwgController  
(POST, /dwg)

3 if문{  
thick에 맞는  
원판단가,  
가공단가,  
드릴링단가}

attribute  
"plateprice",  
"cutprice",  
"drillprice"

Model

4 DwgReader  
.parseDwg(MultipartFile)

(Java) List<FinalObject>

Gson().toJson(javaList)

attribute  
"finalObjectList" :  
JSON (List<FinalObject>)

5 JavaScript

Response  
return  
"/dwg/output"

canvas(그림),  
html tag(가격)  
출력



Spring

## 동작 상세 ① 시작 (/dwg/input.html)

```
<form action="/dwg" enctype="multipart/form-data" method="post" onsubmit="showLoading()">
  <div>
    .dwg파일 : <input accept=".dwg" name="dwgFile" type="file">
  </div>
  <input type="radio" name="thick" value="2" checked> 2T [...생략...]<br>
  <input type="radio" name="thick" value="5"> 5T [...생략...]<br>
  <input type="radio" name="thick" value="10"> 10T [...생략...]<br>
  <input th:name="${_csrf.parameterName}" th:value="${_csrf.token}" type="hidden">
  <button type="submit">업로드</button>
</form>
```

submit 클릭시  
"dwgFile" = 사용자가 입력한 파일  
"thick" = 사용자가 선택한 값  
을 포함하여 /dwg로 POST 요청을 보낸다

## 동작 상세 ② DwgController

```
@PostMapping(value = "/dwg")
public String uploadDwg(MultipartFile dwgFile, Model model, @RequestParam("thick") int thick) throws IOException {

    String json = new Gson().toJson(dwgReader.parseDwg(dwgFile));
    System.out.println("thick : "+thick);
    if (thick == 2){
        model.addAttribute("plateprice", 0.2);
        model.addAttribute("cutprice", 20);
        model.addAttribute("drillprice", 500);
    }
    else if (thick == 5){...}
    else {...}

    model.addAttribute("finalObjectList", json);
    return "/dwg/output";
}
```

## 동작 상세 ② DwgController

`@PostMapping(value = "/dwg")`

`public String uploadDwg(MultipartFile dwgFile, Model model, @RequestParam("thick") int thick) throws IOException {`

`String json = new Gson().toJson(dwgReader.parseDwg(dwgFile));`

List<FinalObject>를 Gson 라이브러리를 통해 JSON형태의 String으로 변환한다.

POST를 통해 받은 MultipartFile을 매개변수로 하여 ④DwgReader.parseDwg() 진행 -> List<FinalObject>가 return 된다

```
model.addAttribute("plateprice", 0.2);
model.addAttribute("cutprice", 20);
model.addAttribute("drillprice", 500);
}
else if (thick == 5){...}
else {...}
```

```
model.addAttribute("finalObjectList", json);
return "/dwg/output";
}
```

변환된 json값을 Model의 "finalObjectList"라는 attribute의 값으로 추가한다

## 동작 상세 ③ if문

```
@PostMapping(value = "/dwg")
```

```
public String uploadDwg(MultipartFile dwgFile, Model model, @RequestParam("thick") int thick) throws IOException {
```

"thick"이라는 이름의 parameter의 값을  
thick이라는 이름의 정수형 변수로 사용

```
String json = new Gson().toJson(dwgReader.parseDwg(dwgFile));
```

```
System.out.println("thick : "+thick);
```

```
if (thick == 2){
```

```
    model.addAttribute("plateprice", 0.2);
```

```
    model.addAttribute("cutprice", 20);
```

```
    model.addAttribute("drillprice", 500);
```

```
}
```

```
else if (thick == 5){...}
```

```
else {...}
```

```
model.addAttribute("dwg", json);
```

```
return "/dwg";
```

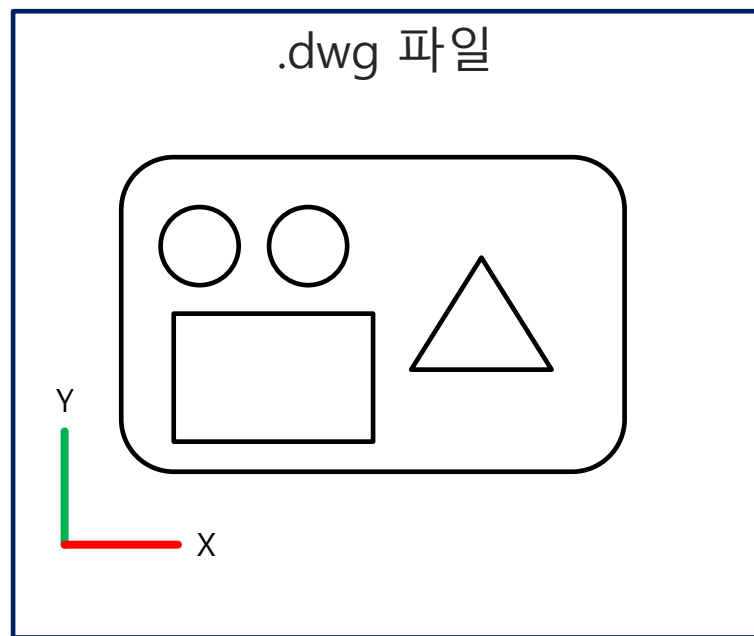
```
}
```

thick의 값에 따라 정해진 가격 3종류를  
Model의 attribute에 추가

(※ 실제 단위는 원₩ 입니다. / 각 단가정보는 임의로 지정했습니다.)



## 동작 상세 ④ DwgReader - 개요



parseDwg(file.dwg)



```
List<FinalObject> {  
    FinalObject1 {  
        partList<CadObject> {  
            Line{시작점, 끝점, 길이 ...},  
            Circle{중심점, 반지름 ...},  
            Arc{중심점, 반지름, 각도 ...}  
            ...  
        }  
        면적;  
        길이;  
        ...  
    }  
    FinalObject2 {...}  
    FinalObject3 {...}  
    FinalObject4 {...}  
    FinalObject5 {...}  
    ...  
}
```

## 동작 상세 ④ DwgReader - 클래스 설명 (Aspose.CAD for Java)

Aspose.CAD 제공 Class

```
import com.aspose.cad
└ .Image;
└ .fileformats.cad
    └ .CadImage;
    └ .cadobjects
        └ .CadBaseEntity;
            └ .CadCircle;
                └ .CadArc;
            └ .CadLine;
                └ .vertices
            └ .Cad2DVertex;
                └ .polylines
            └ .CadPolyline;
```



### Image

파일 혹은 스트림을 받아 읽는 클래스

### CadImage

데이터를 CAD파일로 해석하는 클래스

### CadBaseEntity

CAD파일 내 요소들의 부모객체

### CadCircle

중심점, 반지름 정보

### CadArc

시작각도, 끝각도 정보

CadLine 시작점, 끝점 정보

Cad2DVertex 단일 점 정보

### CadPolyline

Cad2DVertex를 List로 가진 연속Line

## 동작 상세 ④ DwgReader - 클래스 설명 (직접 작성 Class)

### DwgReader

.dwg파일을 읽고 처리하는 main객체

```
moveAll(List<FinalObject> A, double X,  
double Y)
```

모든 FinalObject의 좌표값을  
X, Y만큼 이동시키는 메서드

```
parseDwg(MultipartFile DWG)  
파일을 읽어 List<FinalObject>로  
return하는 main 메서드
```

### TAC (Total Area Comparator)

FinalObject의 totalarea를 비교하기 위한  
Comparator 객체

```
compare(FinalObject A, FinalObject B)  
비교 실행 메서드
```

### CadObject

CAD파일 내 요소들의 부모객체

```
double startX;  
...  
double length;  
double area;  
객체정보, 계산정보 값 멤버들
```

### Line

시작점, 끝점 정보

### Circle

중심점, 반지름 정보

### Arc

중심점, 반지름, 시작각, 끝각 정보

```
roundcut(double input)  
반올림용 메서드(소수 2자리까지)
```

### FinalObject

1개의 정상적으로 완성된 도형 객체

```
List<CadObject> objectlist  
도형을 이루는 CadObject의 리스트
```

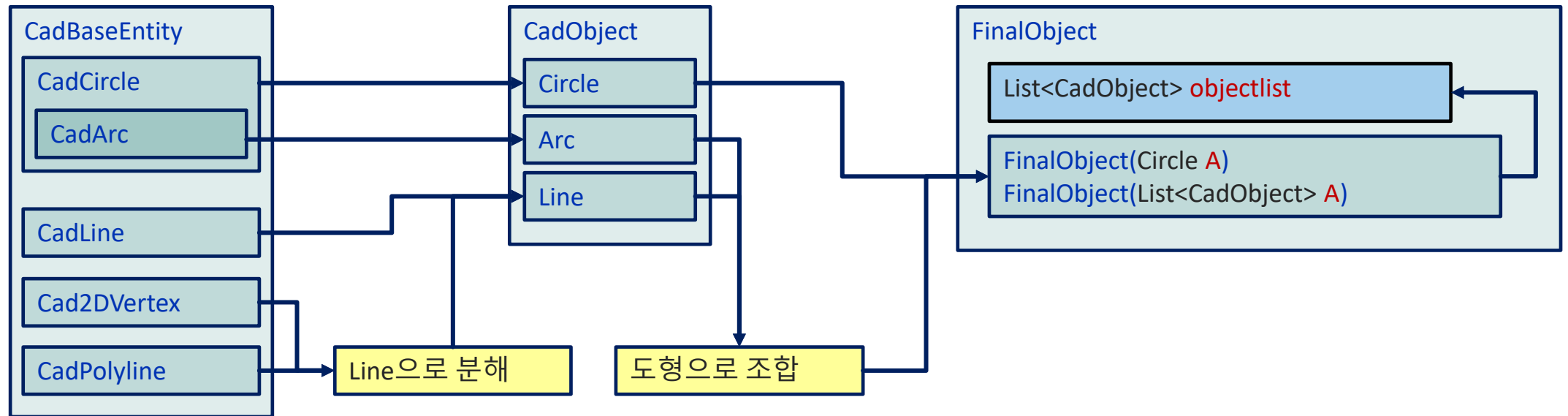
```
boolean circle  
해당 도형이 Circle인지 표시
```

```
double totalarea  
double totallength  
계산정보 값 멤버들
```

```
FinalObject(Circle A)  
FinalObject(List<CadObject> A)  
매개변수 Type별 생성자
```

```
calcLength()  
calcArea()  
생성자에서 호출되는 계산 메서드
```

## 동작 상세 ④ DwgReader - 클래스 설명 (변환 과정)



## 동작 상세 ④ DwgReader - 클래스 설명 (생성자 - Line)

```
Line(double X, double Y, double x, double y) {
```

```
    startX = roundcut(X);  
    startY = roundcut(Y);  
    endX = roundcut(x);  
    endY = roundcut(y);
```

```
    length = roundcut(Math.sqrt((X - x) * (X - x) + (Y - y) * (Y - y)));
```

// 생성시

```
System.out.println("start " + startX + ", " + startY);  
System.out.println("end " + endX + ", " + endY);
```

```
}
```

좌표값은 매개변수를 반올림 후 그대로 대입하고  
길이는 계산 후 반올림하여 length에 대입

## 동작 상세 ④ DwgReader - 클래스 설명 (생성자 - Circle [1])

```
Circle(double X, double Y, double R) {
```

```
    centerX = roundcut(X);  
    centerY = roundcut(Y);  
    radius = roundcut(R);
```

좌표값, 반지름 값은 매개변수를 반올림 후 그대로  
대입하고 길이, 면적은 계산 후 반올림하여 length,  
area에 대입

```
    if (radius*2 == 2 || radius*2 == 3 || radius*2 == 4 || radius*2 == 5 || radius*2 == 6 || radius*2 == 8 || radius*2 == 10  
        || radius*2 == 12) { // 사용 가능한 드릴 직경들  
        drillable = true;  
    } else {  
        drillable = false;  
    }
```

```
    area = roundcut(R * R * Math.PI);  
    length = roundcut((R + R) * Math.PI);
```

```
    // 생성시 정보출력  
    ...생략...
```

```
}
```

## 동작 상세 ④ DwgReader - 클래스 설명 (생성자 - Circle [2])

```
Circle(double X, double Y, double R) {
```

```
    centerX = roundcut(X);  
    centerY = roundcut(Y);  
    radius = roundcut(R);
```

```
    if (radius*2 == 2 || radius*2 == 3 || radius*2 == 4 || radius*2 == 5 || radius*2 == 6 || radius*2 == 8 || radius*2 == 10  
        || radius*2 == 12) { // 사용 가능한 드릴 직경들  
        drillable = true;  
    } else {  
        drillable = false;  
    }
```

```
    area = roundcut(R * R * Math.PI);  
    length = roundcut((R + R) * Math.PI);
```

```
    // 생성시 정보출력  
    ...생략...
```

```
}
```

Circle 직경이 드릴로 뚫을 수 있는 직경이면 drillable = true;  
(현재 4mm, 6mm, 8mm, 10mm, 12mm, 16mm, 20mm로 임의로 지정한 값입니다.)

## 동작 상세 ④ DwgReader - 클래스 설명 (생성자 - Arc [1])

```
Arc(double X, double Y, double R, double startA, double endA) {
```

```
    centerX = roundcut(X);  
    centerY = roundcut(Y);  
    radius = roundcut(R);  
    startangle = roundcut(startA);  
    endangle = roundcut(endA);
```

좌표값, 반지름, 각도 값은  
매개변수를 반올림 후 그대로 대입

```
    startY = roundcut(Y + R * Math.sin(Math.toRadians(startA)));  
    endX = roundcut(X + R * Math.cos(Math.toRadians(endA)));  
    endY = roundcut(Y + R * Math.sin(Math.toRadians(endA)));
```

```
    length = roundcut(((R + R) * Math.PI) * Math.abs(startA - endA) / 360); // 둘레 * (일부각도 / 전체각도)  
    area = roundcut((R * R * Math.PI) * Math.abs(startA - endA) / 360); // 면적 * (일부각도 / 전체각도)
```

```
    // 생성시 정보출력  
    ...생략...
```

```
}
```



## 동작 상세 ④ DwgReader - 클래스 설명 (생성자 - Arc [2])

```
Arc(double X, double Y, double R, double startA, double endA){
```

```
centerX = roundcut(X);  
centerY = roundcut(Y);  
radius = roundcut(R);  
startangle = roundcut(startA);  
endangle = roundcut(endA);
```

기존에 없던 호의 시작, 끝점을  
삼각함수로 계산하여 대입

```
startX = roundcut(X + R * Math.cos(Math.toRadians(startA)));  
startY = roundcut(Y + R * Math.sin(Math.toRadians(startA)));  
endX = roundcut(X + R * Math.cos(Math.toRadians(endA)));  
endY = roundcut(Y + R * Math.sin(Math.toRadians(endA)));
```

```
length = roundcut(((R + R) * Math.PI) * Math.abs(startA - endA) / 360); // 둘레 * (일부각도 / 전체각도)  
area = roundcut((R * R * Math.PI) * Math.abs(startA - endA) / 360); // 면적 * (일부각도 / 전체각도)
```

길이(호 길이), 면적(부채꼴면적)  
계산하여 대입

```
// 생성시 정보출력  
...생략...
```

```
}
```

## 동작 상세 ④ DwgReader - 클래스 설명 (생성자 - FinalObject)

FinalObject(Circle A) { // 원 들어올 경우

```
objectlist = new ArrayList<>();  
objectlist.add(A); // 그냥 추가  
totalarea = A.area;  
totallength = A.length;  
circle = true;
```

Circle의 경우 객체 하나로 도형이 완성되므로 멤버값을 그대로 넣음  
(circle은 true로 대입하고 objectlist는 길이가 1이 됨)

FinalObject(List<CadObject> A) { // 완성된 닫힌 리스트 들어올 경우

```
objectlist = new ArrayList<>();  
for (int i = 0; i < A.size(); i++) {  
    objectlist.add(A.get(i));  
}
```

```
totallength = calcLength();  
totalarea = calcArea();
```

새 List 생성 후  
매개변수로 들어온 List의 내용을 복사  
계산처리용 임시리스트인 A는 FinalObject 생성 후 clear()할 예정.  
objectlist가 A를 참조하게 되면 (대입연산자 사용시)  
A clear()시 objectlist도 clear()된다.

```
} 총 길이 계산, 총 면적 계산 메서드 호출
```

## 동작 상세 ④ DwgReader - 메서드 설명 ( FinalObject calcLength() )

```
double calcLength() { // 총 길이 계산
```

```
    double result = 0;
```

```
    for (int i = 0; i < objectlist.size(); i++) {
```

```
        result += objectlist.get(i).length;
```

```
    }  
    return result;
```

자신의 objectlist를 불러 모든 CadObject의  
length를 더한 뒤 return

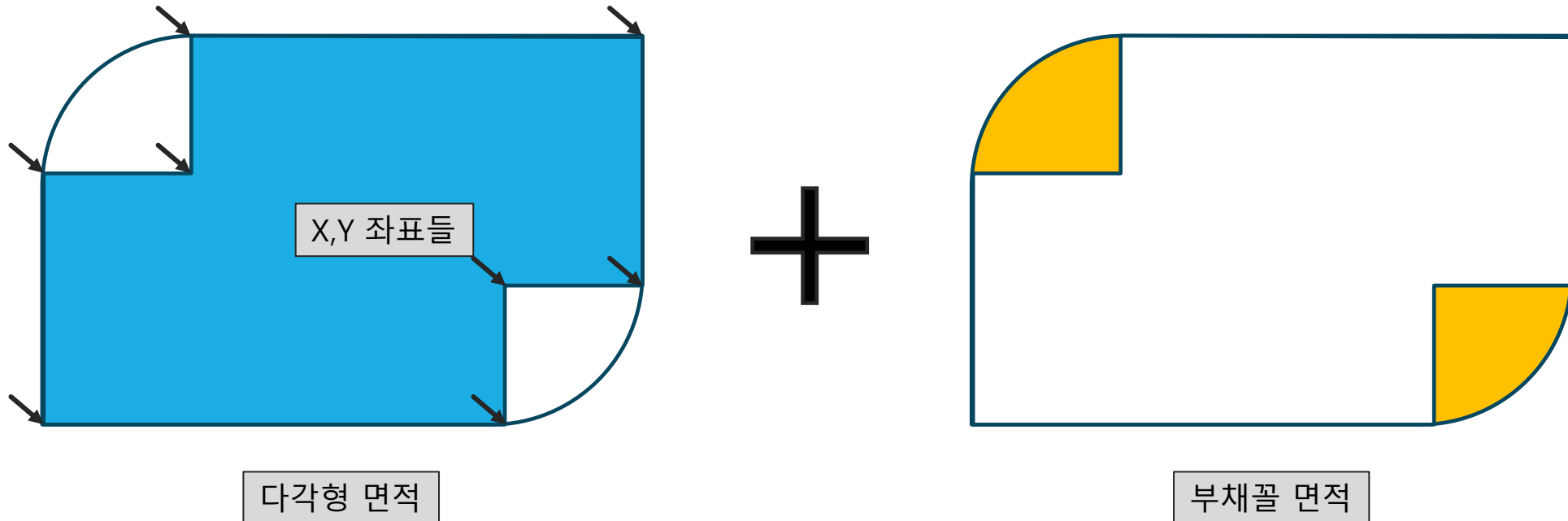
## 동작 상세 ④ DwgReader - 메서드 설명 ( FinalObject calcArea() [1])

### 슈레이스 공식

사선 공식으로도 알려져있으며, 여러 개의 좌표로 이루어진 다각형의 면적을 계산하는 식이다.

$$\frac{1}{2} \begin{vmatrix} x_1 & x_2 & x_3 & \cdots & x_n & x_1 \\ y_1 & y_2 & y_3 & \cdots & y_n & y_1 \end{vmatrix}$$

calcArea() 메서드는 이를 이용해 다각형의 면적을 계산한 뒤 구성List에 Arc가 있으면 해당 Arc의 부채꼴 면적을 더하는 식으로 FinalObject의 면적을 계산한다



## 동작 상세 ④ DwgReader - 메서드 설명 ( FinalObject calcArea() [2])

```
double calcArea() {
```

슈레이스 공식은 필요 배열의 길이가  
총 좌표의 개수 +1이므로 1부터 시작함

```
int count = 1;
```

```
for (int i = 0; i < objectlist.size(), i++) {
```

```
if (objectlist.get(i) instanceof Arc) {
```

```
count += 2;
```

```
} else {
```

```
count += 1;
```

```
}
```

```
}
```

```
double[] shoeX = new double[count];
```

```
double[] shoeY = new double[count];
```

자신의 objectlist를 불러  
Line이면 +1(끝점), Arc면 +2(중심점, 끝점)

계산한 count로 X용 배열, Y용 배열 생성

--계속--

## 동작 상세 ④ DwgReader - 메서드 설명 ( FinalObject calcArea() [3])

```
int j = 0;
```

objectlist 순번, 좌표가 들어갈 배열의 index 별도 계산

```
for (int i = 0; i < objectlist.size(); i++) {
```

```
    if (objectlist.get(i) instanceof Arc) {
```

Arc일 경우 중심점, 끝점 대입 후 j += 2

```
        shoeX[j] = objectlist.get(i).centerX;
```

```
        shoeY[j] = objectlist.get(i).centerY;
```

```
        shoeX[j + 1] = objectlist.get(i).endX;
```

```
        shoeY[j + 1] = objectlist.get(i).endY;
```

```
        j += 2;
```

```
    } else {
```

```
        shoeX[j] = objectlist.get(i).endX;
```

Line일 경우 끝점 대입 후 j += 1

```
        shoeY[j] = objectlist.get(i).endY;
```

```
        j += 1;
```

```
    }
```

```
}
```

```
shoeX[count-1] = shoeX[0];
```

첫번째 값을 배열 마지막에 복사  
(슈레이스 공식 이미지 참고)

```
shoeY[count-1] = shoeY[0];
```

--계속--

## 동작 상세 ④ DwgReader - 메서드 설명 ( FinalObject calcArea() [4])

```
double sumA = 0;
double sumB = 0;
for (int i = 0 ; i < shoeX.length-1; i++) {
    sumA += shoeX[i]*shoeY[i+1];
    sumB += shoeY[i]*shoeX[i+1];
}

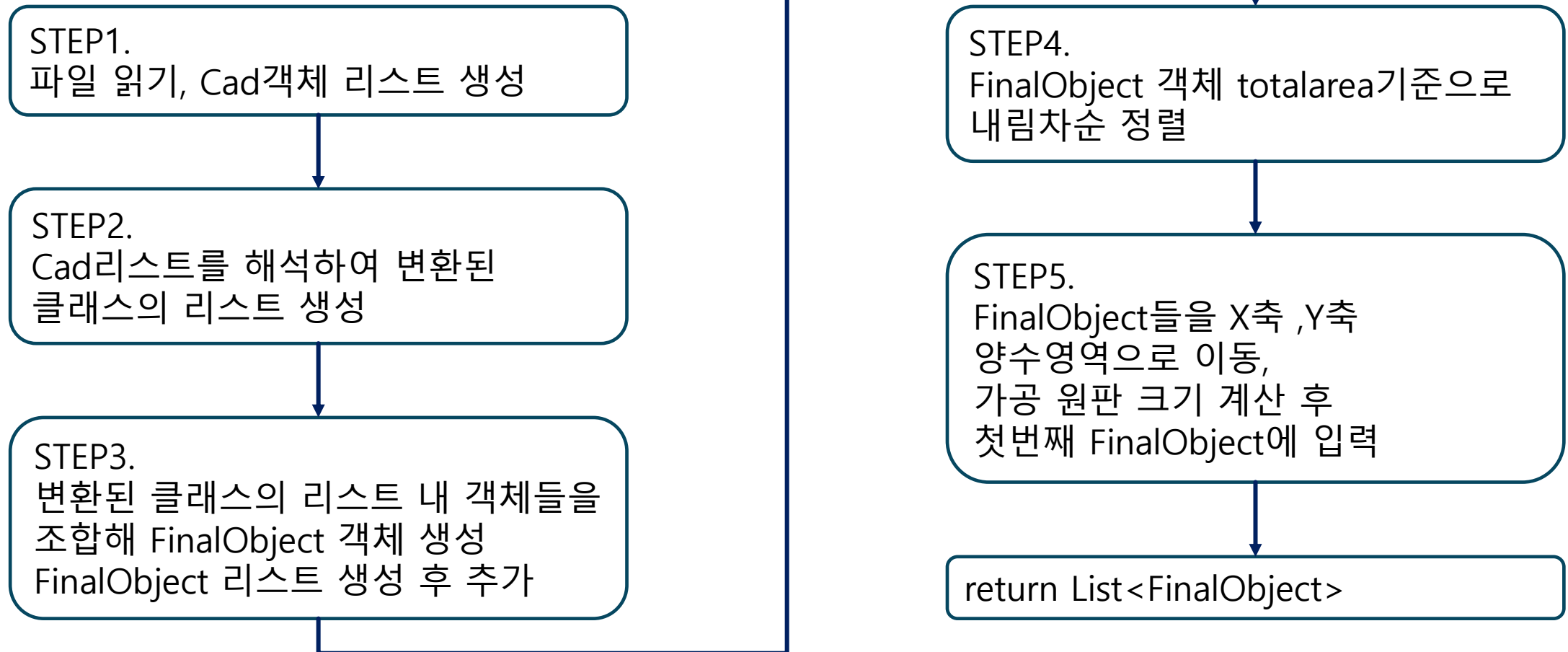
result += Math.abs(sumA-sumB)/2
```

```
for (int i = 0 ; i < objectlist.size() ; i++) {
    if (objectlist.get(i) instanceof Arc){
        result += objectlist.get(i).area;
    }
}
```

```
return roundcut(result);
```

슈레이스 공식으로 구한 다각형 면적  
+ Arc의 부채꼴 면적 합산 후 반올림된 값 return

## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() )





## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP1 [1])

```
public static List<FinalObject> parseDwg(MultipartFile DWG) throws IOException {
```

```
    CadImage cadImage = (CadImage) Image.load(DWG.getInputStream());
```

```
    CadBaseEntity[] objectlist = cadImage.getEntities();
```

```
    List<CadObject> partList = new ArrayList<>();  
    List<CadObject> partList2 = new ArrayList<>();  
    List<FinalObject> finalList = new ArrayList<>();
```

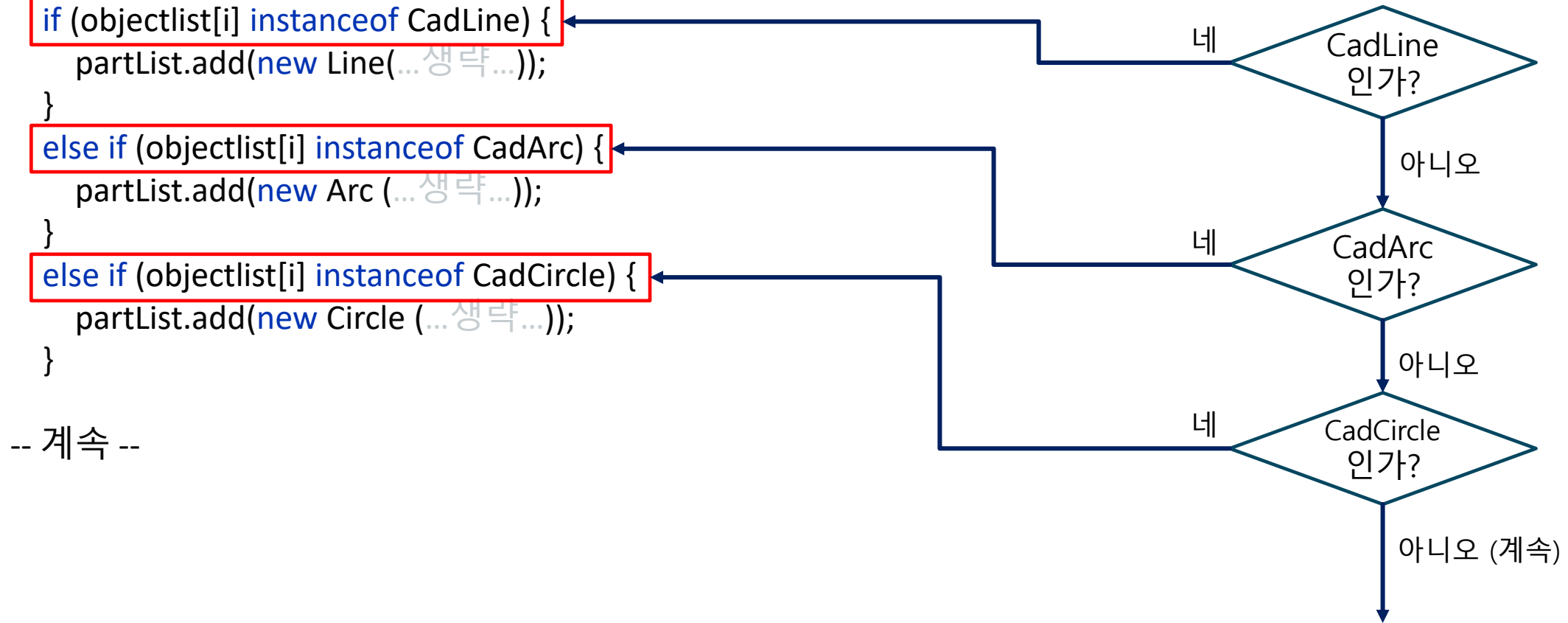
partList : 모든 CadObject를 담을 List  
partList2 : partList에서 CadObject를 꺼내 FinalObject를 만들기 위한 임시 List  
finalList : 완성된 FinalObject 리스트

Controller로 들어온 파일의  
InputStream을 입력,  
파일 내 Cad객체들을 배열로 저장한다

## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP2 [1])

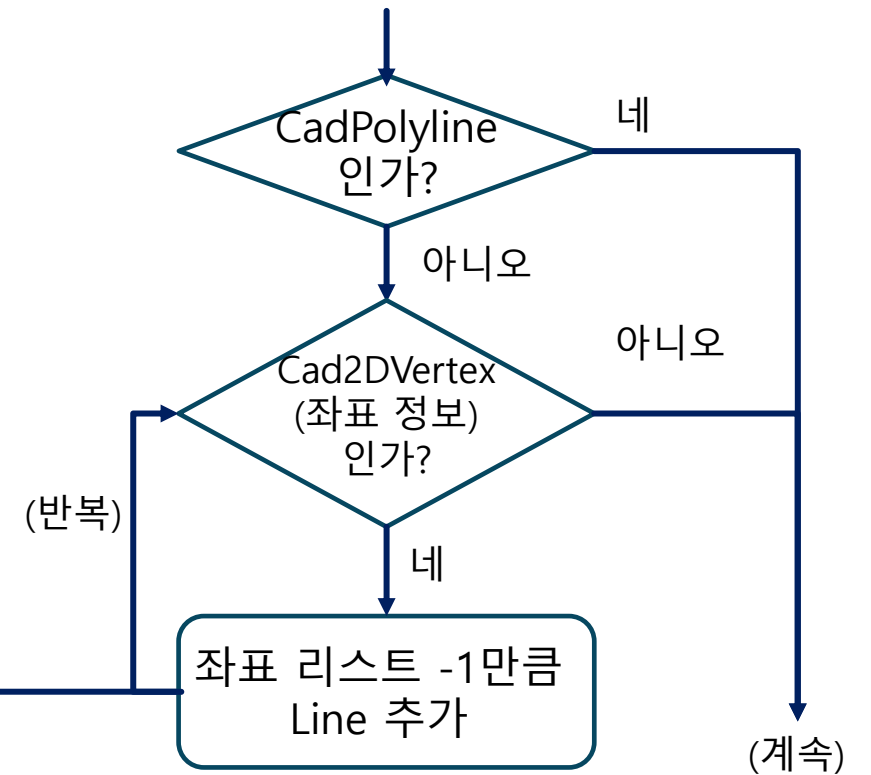
```
for (int i = 0; i < objectlist.length; i++) {  
    if (objectlist[i] instanceof CadLine) {  
        partList.add(new Line(...생략...));  
    }  
    else if (objectlist[i] instanceof CadArc) {  
        partList.add(new Arc (...생략...));  
    }  
    else if (objectlist[i] instanceof CadCircle) {  
        partList.add(new Circle (...생략...));  
    }  
}
```

-- 계속 --

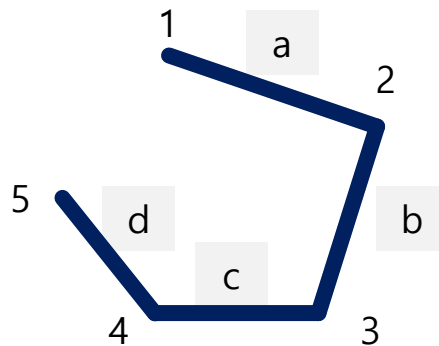


## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP2 [2])

```
else if (objectlist[i] instanceof CadPolyline) {  
    CadPolyline x = (CadPolyline) objectlist[i];  
    for (int j = 1; j < objectlist[i].getChildObjects().size(); j++) {  
        if (objectlist[i].getChildObjects().get(j) instanceof Cad2DVertex) {  
            ...생략...  
            partList.add(new Line(startX, startY, endX, endY));  
        }  
    }  
}
```

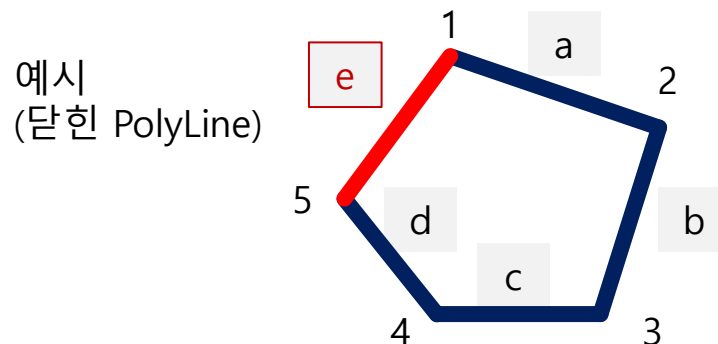
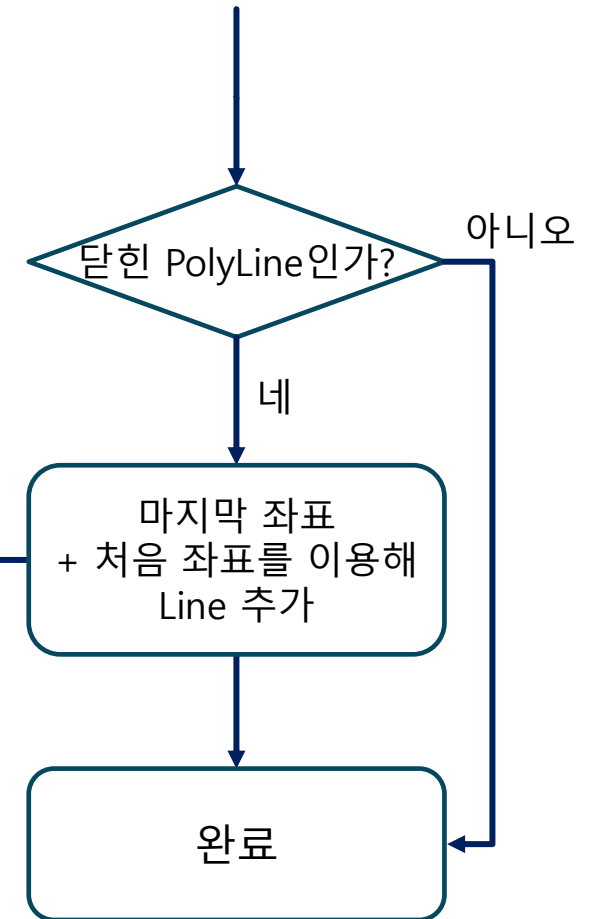


예시  
(열린 PolyLine)



## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP2 [3])

```
if (x.getFlag() == 1) {  
    PolyLine.getFlag()가 1이면 닫힌  
    폴리라인이다  
    double startX = ((Cad2DVertex) objectlist[i].getChildObjects().get(0))  
        .getLocationPoint().getX();  
    double startY = ((Cad2DVertex) objectlist[i].getChildObjects().  
        .get(objectlist[i].getChildObjects().size() - 2)).getLocationPoint().getY();  
    double endX = ((Cad2DVertex) objectlist[i].getChildObjects().get(0))  
        .getLocationPoint().getX();  
    double endY = ((Cad2DVertex) objectlist[i].getChildObjects().get(0))  
        .getLocationPoint().getY();  
    partList.add(new Line(startX, startY, endX, endY));  
}
```



## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP3 [1])

```
for (int i = 0; i < partList.size(); i++) {  
    if (partList.get(i) instanceof Circle) {  
        finalList.add(new FinalObject((Circle) partList.get(i)));  
        partList.remove(i);  
        i--;  
    }  
}
```

Circle은 바로 FinalObject 생성 후  
리스트에 추가



## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP3 [2])

```
while (partList.size() != 0) {
```

```
    if (partList2.size() == 0) {
```

```
        partList2.add(partList.get(0));  
        partList.remove(0);
```

partList2가 비었을경우 partList 첫객체를 넣고  
for문 시작

```
    for (int i = 0; i < partList.size(); i++) {
```

```
        double lastEndX = partList2.get(partList2.size() - 1).endX;
```

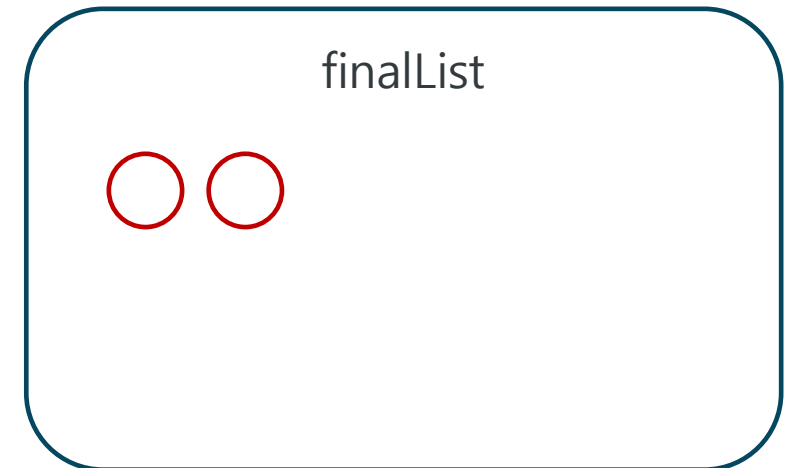
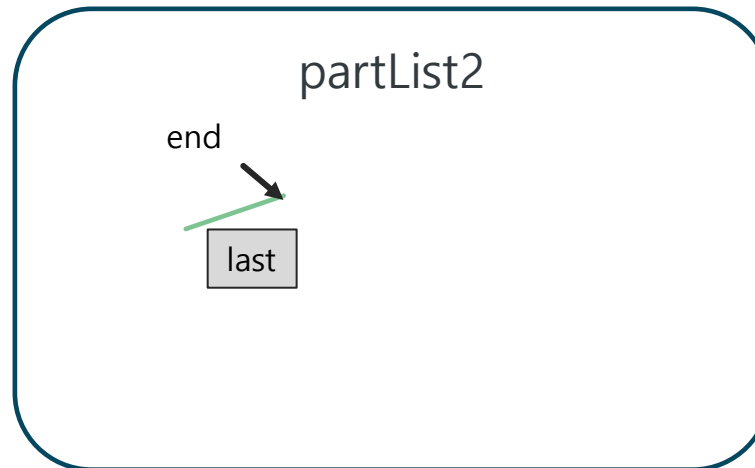
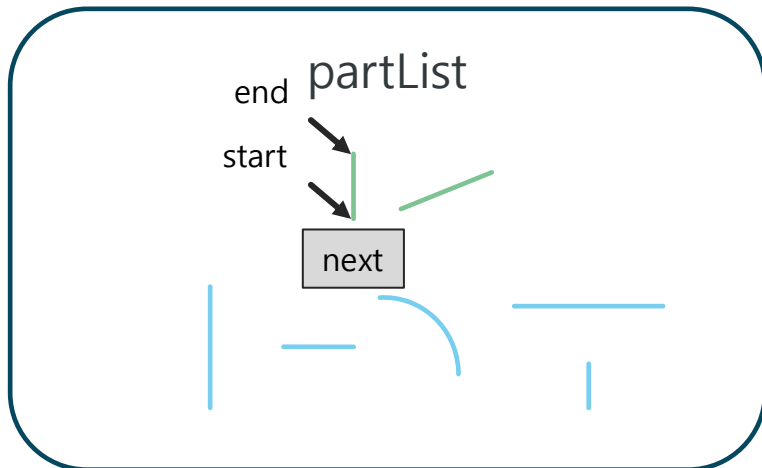
```
        double nextStartX = partList.get(i).startX;
```

```
        double nextEndX = partList.get(i).endX;
```

비교 대상의 값 저장

...생략...

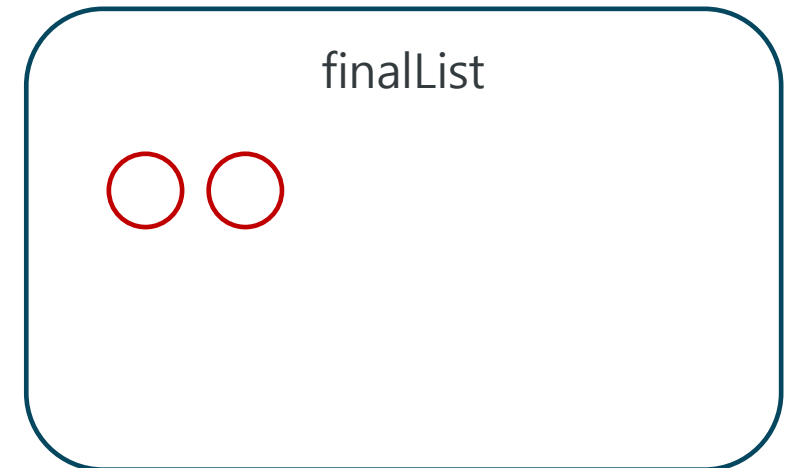
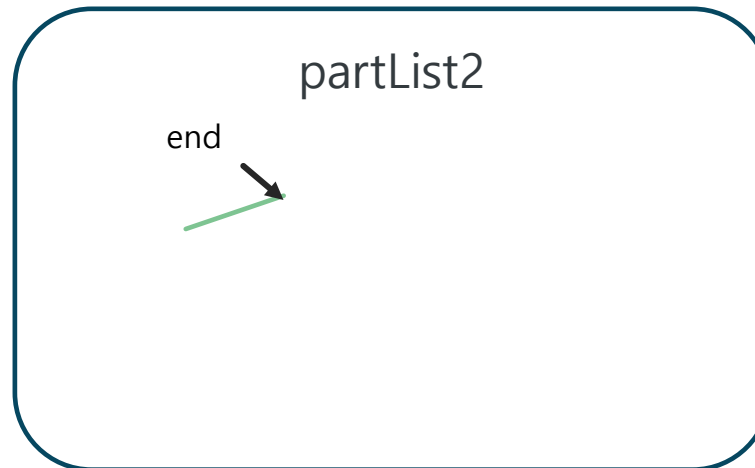
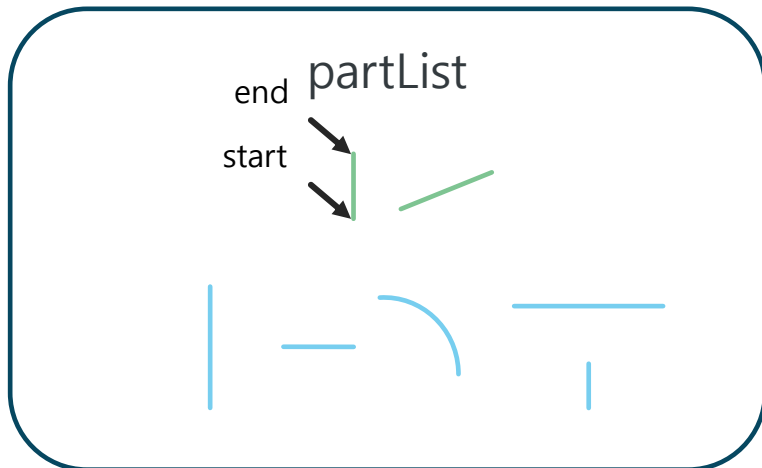
```
    }
```



## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP3 [3])

```
if (lastEndX == nextStartX && lastEndY == nextStartY) {  
    partList2.add(partList.get(i));  
    partList.remove(i);  
    i = -1;continue;}  
if (lastEndX == nextEndX && lastEndY == nextEndY) {  
    double tempstartX = partList.get(i).endX;  
    ...생략...  
    partList2.add(partList.get(i));  
    partList.remove(i);  
    i = -1;}
```

partList2의 끝점 == partList의 시작점 일치시  
해당 객체를 partList2로 이동



## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP3 [4])

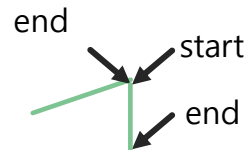
```
if (lastEndX == nextStartX && lastEndY == nextStartY) {  
    partList2.add(partList.get(i));  
    partList.remove(i);  
    i = -1;continue;}  
if (lastEndX == nextEndX && lastEndY == nextEndY) {  
    double tempstartX = partList.get(i).endX;  
    ...생략...  
    partList2.add(partList.get(i));  
    partList.remove(i);  
    i = -1;}
```

partList2의 끝점 == partList의 끝점 일치시  
해당 객체의 start, end를 바꾼 뒤 partList2로 이동

partList



partList2



finalList





## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP3 [5])

```
if (partList2.get(0).startX == partList2.get(partList2.size() - 1).endX  
    && partList2.get(0).startY == partList2.get(partList2.size() - 1).endY) {  
    finalList.add(new FinalObject(partList2));  
    partList2.clear();  
}  
else {  
    partList2.clear();  
}
```

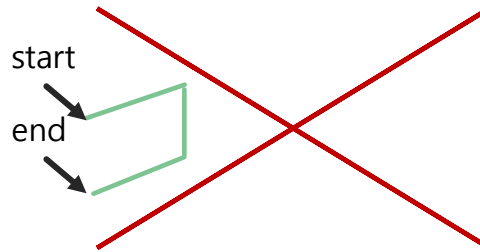
partList2의 시작점, 끝점 불일치시

FinalObject로 만들지 않고 partList2 내용 삭제

partList



partList2



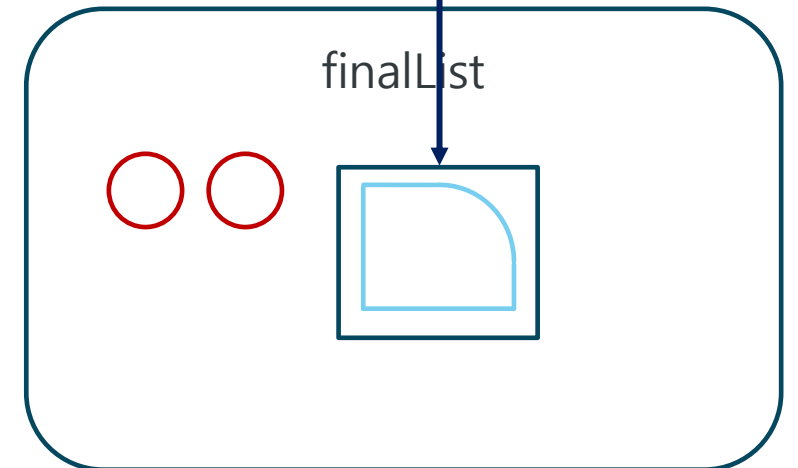
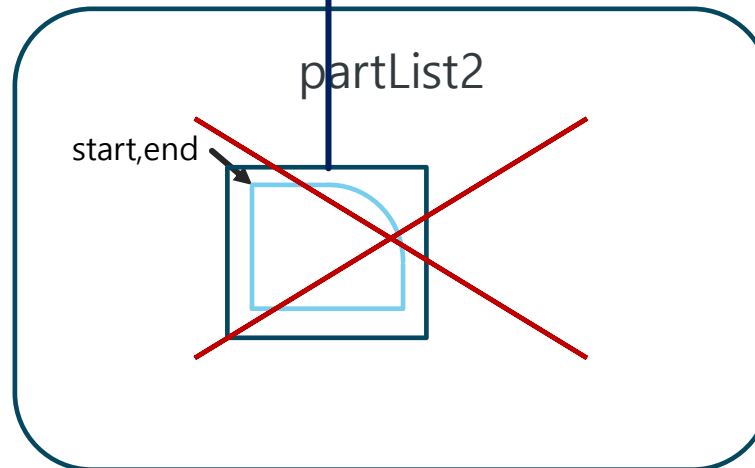
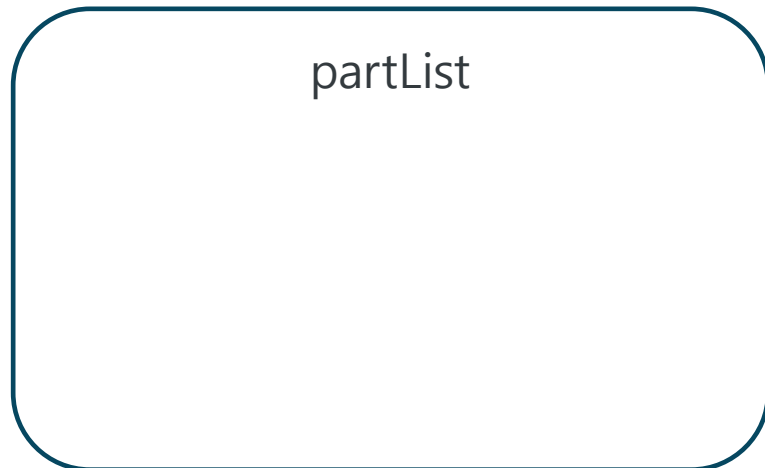
finalList



## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP3 [6])

```
if (partList2.get(0).startX == partList2.get(partList2.size() - 1).endX  
    && partList2.get(0).startY == partList2.get(partList2.size() - 1).endY) {  
    finalList.add(new FinalObject(partList2));  
    partList2.clear();  
}  
else {  
    partList2.clear();  
}
```

일치시 FinalObject객체 생성 후  
partList2 내용 삭제



## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP4 [1])

-----  
TAC.java (TotalAreaComparator)

```
public class TAC implements Comparator<FinalObject> {  
    @Override  
    public int compare(FinalObject A, FinalObject B) {  
        return Double.compare(B.totalarea, A.totalarea);  
    }  
}
```

-----

```
finalList.sort(new TAC());
```

FinalObject객체의 totalarea의 크기를 기준으로  
내림차순 정렬하는 Comparator 객체 생성 후

finalList를 내림차순 정렬함

## 동작 상세 ④ DwgReader - 메서드 설명 ( parseDwg() STEP5 [1])

```
FinalObject mainObject = finalList.get(0);
```

리스트의 첫번째 객체(=totalarea가 가장 큰 객체)를 선택

```
for (CadObject A : mainObject.objectlist) {  
    if (A instanceof Circle) {  
        tempCenterX = A.centerX;  
        tempCenterY = A.centerY;  
    } else {  
        tempCenterX += A.startX;  
        tempCenterY += A.startY;  
    }  
    pointcount++;  
}
```

선택된 객체의 objectlist의 모든 객체들의 좌표를 누적 합산

```
double mainCenterX = tempCenterX / pointcount;  
double mainCenterY = tempCenterY / pointcount;
```

평균을 계산해 중심점 좌표 1개를 저장한다

```
moveAll(finalList, -mainCenterX, -mainCenterY);
```

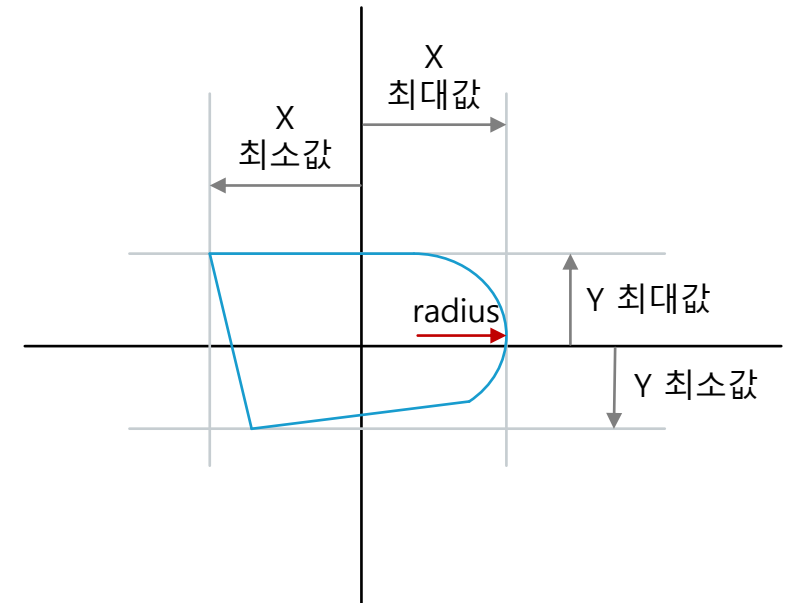
finalList의 모든 객체를 중심점 좌표값의  
음수만큼 이동한다.  
(X=0, Y=0 중심으로 이동됨)

## 동작 상세 ④ DwgReader - 메서드 설명 ( parsDwg() STEP5 [2])

```
double mainMaxX = 0;  
double mainMinX = 0;  
double Xsize = 0;  
...생략...
```

mainObject를 구성하는 모든 좌표, 반지름을 확인하여  
X,Y축별 최소, 최대값 확인

```
for (CadObject A : mainObject.objectlist) {  
    if (A instanceof Arc || A instanceof Circle) {  
        if (A.centerX+A.radius > 0 && A.centerX+A.radius > mainMaxX) {  
            mainMaxX = A.centerX+A.radius;  
        }  
        ...생략...  
    } else {  
        if (A.startX > 0 && A.startX > mainMaxX) {  
            mainMaxX = A.startX;  
        }  
        else if (A.startX < 0 && A.startX < mainMinX) {  
            mainMinX = A.startX;  
        }  
        ...생략...  
    }  
}
```



## 동작 상세 ④ DwgReader - 메서드 설명 ( parsDwg() STEP5 [3])

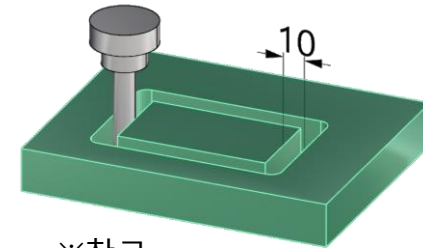
Xsize = mainMaxX - mainMinX;

Ysize = mainMaxY - mainMinY;

mainObject.canvassizeX = Xsize+20;

mainObject.canvassizeY = Ysize+20;

mainObject.canvassizeX, canvassizeY에  
상하좌우 10씩 마진을 더한 값을 입력  
(※10mm는 임의로 지정한 절단공구 직경 값입니다.)

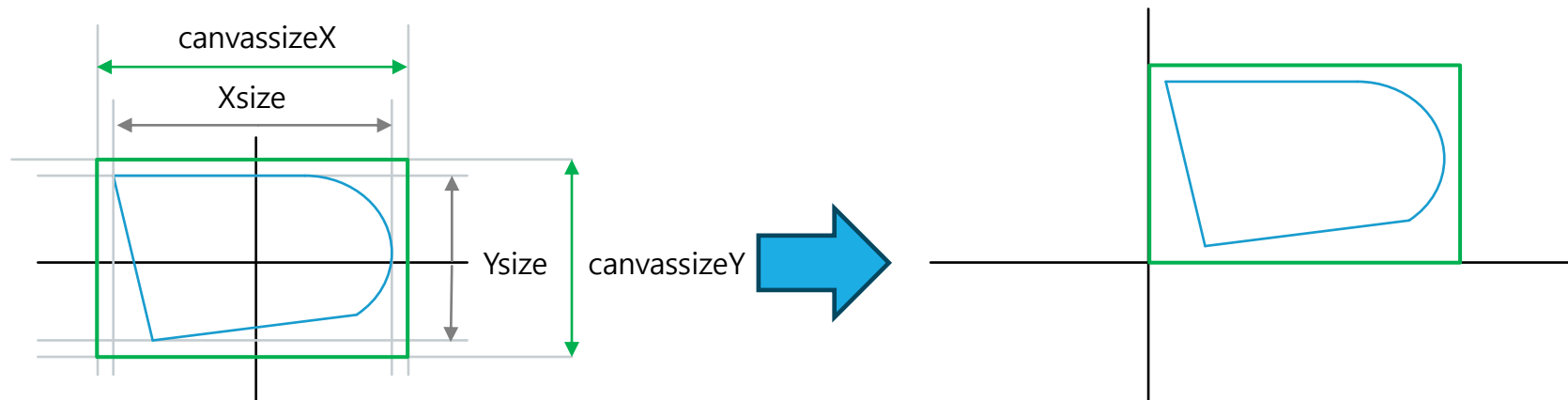


※참고

moveAll(finalList, -mainMinX+10, -mainMinY+10);

마진을 포함한 전체 도형을 X축, Y축 양수 영역으로 이동

cdimage.dispose();  
return finalList;



## 동작 상세 ④ DwgReader - 메서드 설명 ( parsDwg() STEP5 [4])

```
Xsize = mainMaxX - mainMinX;
```

```
Ysize = mainMaxY - mainMinY;
```

```
mainObject.canvassizeX = Xsize+20;
```

```
mainObject.canvassizeY = Ysize+20;
```

```
moveAll(finalList, -mainMinX+10, -mainMinY+10);
```

```
cadImage.dispose();  
return finalList;
```

열린 cadImage를 닫고 최종 List<FinalObject>를 return

## 동작 상세 ⑤ JavaScript - output.html

```
<div class="resultcontent">  
  <h1 class="resulttitle">업로드 결과</h1>
```

```
<canvas id="canvas"></canvas>
```

결과 그림을 표시한 canvas 위치

```
<div>  
  <h2>가공비용 계산</h2>
```

```
  <ul id="resultlist"></ul>  
  <div id="totalprice"></div>
```

가공요소별 가격 추가용 ul태그,  
가격 총합을 표시할 div태그

```
</div>
```

```
</div>
```



## 동작 상세 ⑤ JavaScript - <script> [1]

```
const canvas = document.getElementById("canvas");  
const ctx = canvas.getContext("2d");
```

canvas 태그 선택하여 변수로 저장

```
const jsonObject = JSON.parse(/*[[${finalObjectList}]]*/);  
const plateprice = /*[[${plateprice}]]*/ 0;  
const cutprice = /*[[${cutprice}]]*/ 0;  
const drillprice = /*[[${drillprice}]]*/ 0;
```

attribute로 받은 값 변수로 저장

```
const canvassizeX = jsonObject[0].canvassizeX;  
const canvassizeY = jsonObject[0].canvassizeY;  
const scaler = 1500 / jsonObject[0].canvassizeX;  
canvas.width = jsonObject[0].canvassizeX * scaler;  
canvas.height = jsonObject[0].canvassizeY * scaler;  
ctx.lineWidth = scaler * 1;
```

리스트 내 첫 객체의 값을 이용해  
canvas에 필요한 값들 설정  
(※ 1500은 임의로 지정한 비율값 기준입니다.)  
canvas의 1500 = 1500px  
FinalObject의 1500 = 1500mm  
크거나 작은 결과를 1500px 기준으로 맞추기위해  
scaler 값을 사용

```
let totalprice = 0;
```

총액 누적 계산용 변수

## 동작 상세 ⑤ JavaScript - <script> [2]

```
ctx.strokeStyle = "grey";  
ctx.strokeRect(...생략...);
```

리스트 내 값과 canvas의 strokeRect(), arc(), moveTo(), lineTo() 등을 사용해 도형 그리기

```
let container = document.getElementById("resultlist");  
const mainplate = document.createElement("li");  
resulttemp.textContent = ...생략...  
container.appendChild(resulttemp);
```

해당 요소에 대한 내용을 담은 <li>태그 작성 후 추가

```
totalprice += Math.ceil(jsonObject[i].totallength * cutprice);
```

...이하 리스트 객체 끝까지 반복문...

객체에 포함된 값과 attribute로 받은 단가정보로 금액을 계산해 총액에 누적

```
container = document.getElementById("totalprice");  
const totalpriceout = document.createElement("h2");  
totalpriceout.textContent = "총 합 : " + totalprice + "원";  
container.appendChild(totalpriceout);
```

총합 결과를 <h2>태그 작성 후 추가

## 동작 상세 ⑤ JavaScript - <script> [3]

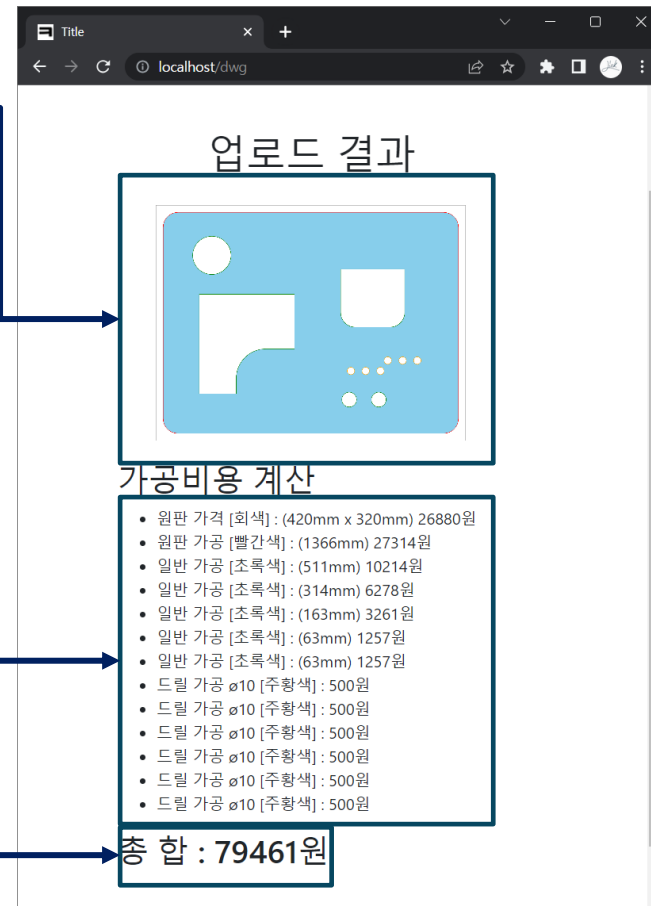
```
ctx.strokeStyle = "grey";
ctx.strokeRect(...생략...);
```

```
let container = document.getElementById("resultlist");
const mainplate = document.createElement("li");
resulttemp.textContent = ...생략...
container.appendChild(resulttemp);
```

```
totalprice += Math.ceil(jsonObject[i].totallength * cutprice);
```

## ...이하 리스트 객체 끝까지 반복문

```
container = document.getElementById("totalprice");
const totalpriceout = document.createElement("h2");
totalpriceout.textContent = "총 합 : " + totalprice + "원";
container.appendChild(totalpriceout);
```





Java Portfolio - 끝 -

