HW5

1분반

2024년 10월 23일

32211792

박재홍

Apthropod 클래스)

```java
public class Arthropod {
    private String arthropodname;  // 곤충의 이름을 저장하는 필드 (Your Code)
    private int bodyRegions;
    private int pairsOfAntennae;
    private RespirationType respiration;
    private MetamorphosisType metamorphosis;
    private int pairsOfWing;
    private int numberOfLegs;
    private String distinction;

    // ArthropodBuilder를 사용하여 Arthropod 객체를 생성하는 생성자
    public Arthropod(ArthropodBuilder builder){
        this.bodyRegions = builder.bodyRegions;
        this.pairsOfAntennae = builder.pairsOfAntennae;
        this.respiration = builder.respiration;
        this.metamorphosis = builder.metamorphosis;
        this.pairsOfWing = builder.pairsOfWing;
        this.numberOfLegs = builder.numberOfLegs;
        this.distinction = builder.distinction;
        this.arthropodname = builder.arthropodname; // 곤충의 이름을 builder에서 가져옴 (Your Code)
    }

    // Getter 메소드들: 각각의 필드에 대한 값을 반환
    public int getBodyRegions(){
        return bodyRegions;
    }

    public int getPairsOfAntennae(){
        return pairsOfAntennae;
    }

    public RespirationType getRespiration(){
        return respiration;
    }

    public MetamorphosisType getMetamorphosis() {
        return metamorphosis;
    }

    public int getPairsOfWing(){
        return pairsOfWing;
    }

    public int getNumberOfLegs(){
        return numberOfLegs;
    }

    public String getDistinction(){
        return distinction;
    }

    public String getArthropodname(){  // 곤충의 이름을 반환하는 메소드 (Your Code)
        return arthropodname;
    }

    // Arthropod 객체의 필드들을 String 형태로 반환
    @Override
    public String toString() {
        return "{" +
                " Arthropod name='" + getArthropodname() + "'" +  // 곤충의 이름 출력 (Your Code)
                ", bodyRegions='" + getBodyRegions() + "'" +
                ", pairsOfAntennae='" + getPairsOfAntennae() + "'" +
                ", respiration='" + getRespiration() + "'" +
                ", metamorphosis='" + getMetamorphosis() + "'" +
                ", pairsOfWing='" + getPairsOfWing() + "'" +
                ", numberOfLegs='" + getNumberOfLegs() + "'" +
                ", distinction='" + getDistinction() + "'" +
                "}";
    }
```

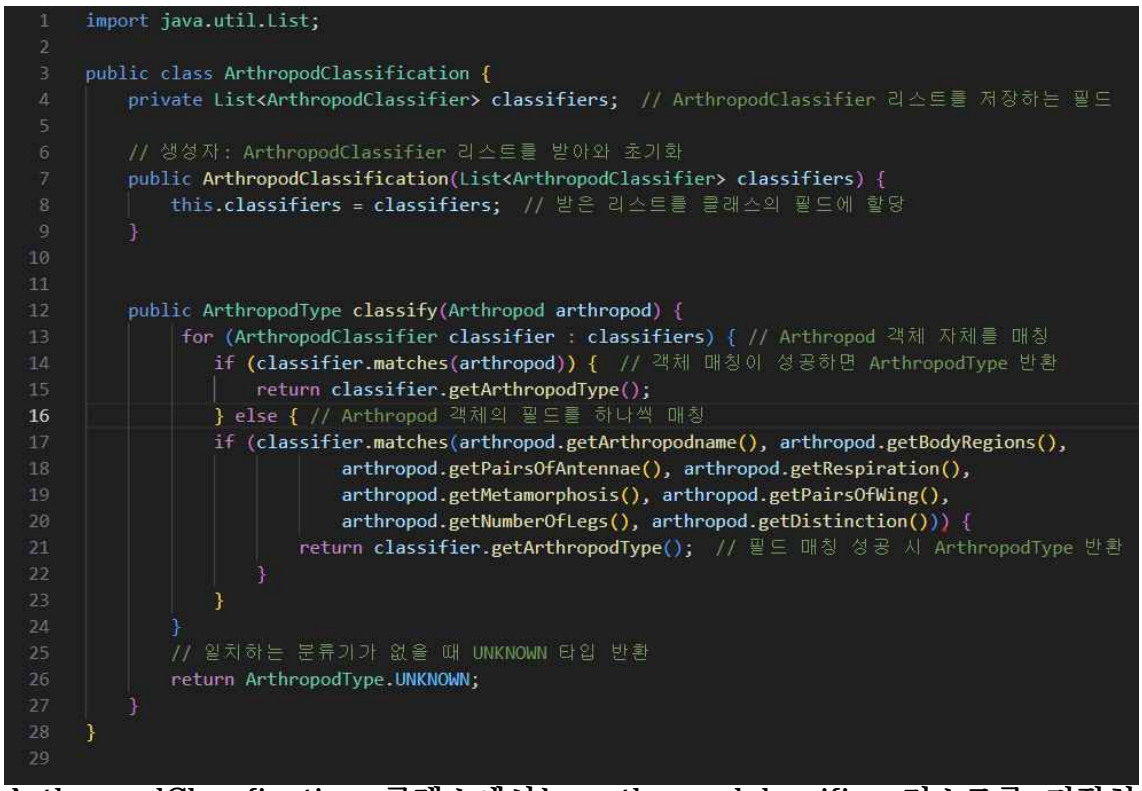
Arthropod 클래스에서는 각 변수들의 이름을 저장해주었고 arthropodbuilder를 통해 arthropod 객체를 생성하는 생성자를 선언했다. 그리고 각각의 변수들에 대한 값을 반환할 수 있도록 getter 메소드를 선언해주었고 arthropod 객체의 필드들을 string 형태로 반환 시켜주었다.

ArthropodBuilder 클래스)

```java
    // Builder 패턴을 사용하여 Arthropod 객체를 단계별로 생성할 수 있도록 하는 정적 내부 클래스
    public static class ArthropodBuilder {

        // 빌더 패턴에서 사용되는 필드들
        private int bodyRegions;
        private int pairsOfAntennae;
        private RespirationType respiration;
        private MetamorphosisType metamorphosis;
        private int pairsOfWing;
        private int numberOfLegs;
        private String distinction;
        private String arthropodname;  // 곤충 이름 (Your Code)

        // 곤충 이름을 설정하는 메소드
        public ArthropodBuilder setArthropodname(String arthropodname){
            this.arthropodname = arthropodname;
            return this;
        }

        // 각 필드에 대한 설정 메소드들
        public ArthropodBuilder setBodyRegions(int bodyRegions) {
            this.bodyRegions = bodyRegions;
            return this;
        }

        public ArthropodBuilder setPairsOfAntennae(int pairsOfAntennae){
            this.pairsOfAntennae = pairsOfAntennae;
            return this;
        }

        public ArthropodBuilder setRespiration(RespirationType respiration){
            this.respiration = respiration;
            return this;
        }

        public ArthropodBuilder setMetamorphosis(MetamorphosisType metamorphosis){
            this.metamorphosis = metamorphosis;
            return this;
        }

        public ArthropodBuilder setPairsOfWing(int pairsOfWing){
            this.pairsOfWing = pairsOfWing;
            return this;
        }

        public ArthropodBuilder setNumberOfLegs(int numberOfLegs){
            this.numberOfLegs = numberOfLegs;
            return this;
        }

        public ArthropodBuilder setDistinction(String distinction){
            this.distinction = distinction;
            return this;
        }

        // Builder 패턴에서 실제로 Arthropod 객체를 생성하는 메소드
        public Arthropod build() {
            return new Arthropod(this);  // 빌더 객체를 사용하여 Arthropod 객체 생성
        }
    }
}

```

Arthropod builder 클래스에서는 빌더 패턴에서 사용되는 필드들을 선언해주었고 각각의 필드들에 대해 설정 pathememf을 선언해주었고 마지막에 build 메소드를 통해 builder 패턴에서 실제로 arthropod 객체를 생성할 수 있도록 해주었다.

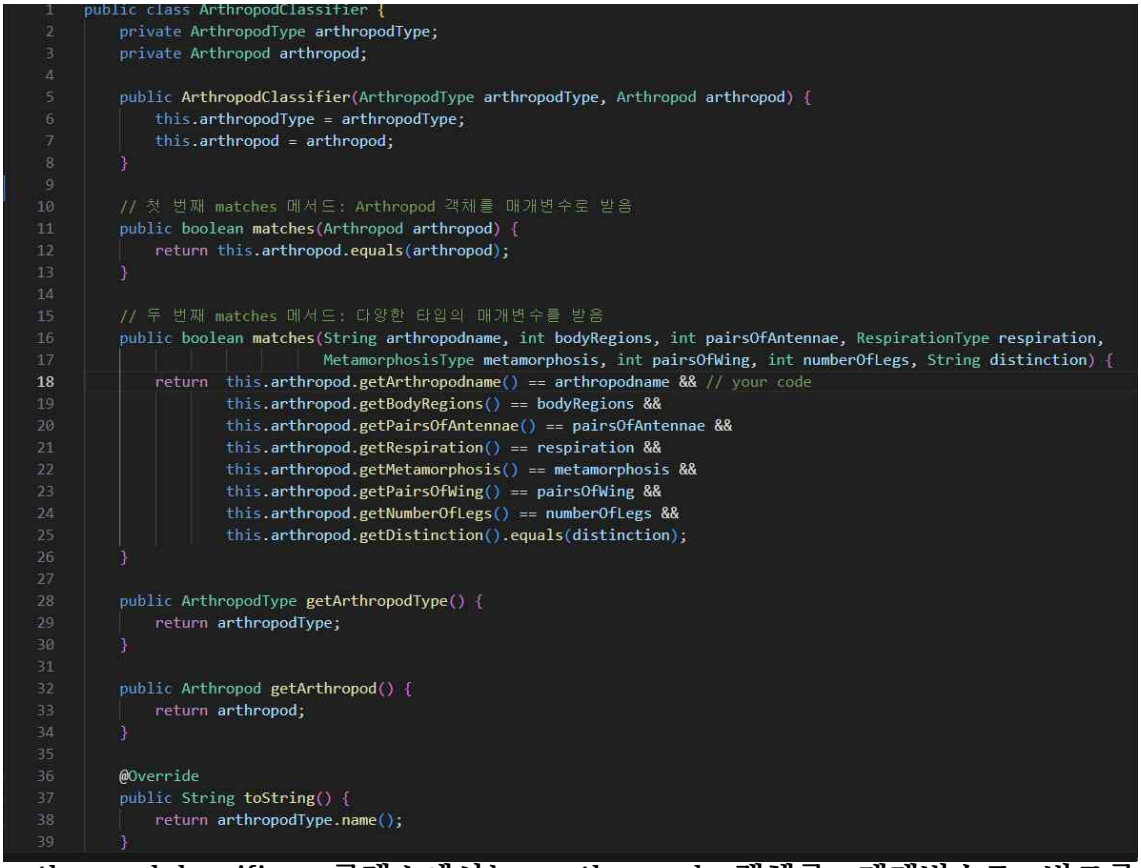
ArthropodClassfication 클래스)



```java
import java.util.List;

public class ArthropodClassification {
    private List<ArthropodClassifier> classifiers;  // ArthropodClassifier 리스트를 저장하는 필드

    // 생성자: ArthropodClassifier 리스트를 받아와 초기화
    public ArthropodClassification(List<ArthropodClassifier> classifiers) {
        this.classifiers = classifiers;  // 받은 리스트를 클래스의 필드에 할당
    }


    public ArthropodType classify(Arthropod arthropod) {
        for (ArthropodClassifier classifier : classifiers) { // Arthropod 객체 자체를 매칭
            if (classifier.matches(arthropod)) {  // 객체 매칭이 성공하면 ArthropodType 반환
                return classifier.getArthropodType();
            } else { // Arthropod 객체의 필드를 하나씩 매칭
            if (classifier.matches(arthropod.getArthropodname(), arthropod.getBodyRegions(),
                        arthropod.getPairsOfAntennae(), arthropod.getRespiration(),
                        arthropod.getMetamorphosis(), arthropod.getPairsOfWing(),
                        arthropod.getNumberOfLegs(), arthropod.getDistinction())) {
                    return classifier.getArthropodType();  // 필드 매칭 성공 시 ArthropodType 반환
                }
            }
        }
        // 일치하는 분류기가 없을 때 UNKNOWN 타입 반환
        return ArthropodType.UNKNOWN;
    }
}

```

ArthropodClassfication 클래스에서는 arthropodclassifier 리스트를 저장하는 필드를 선언해주었고 받은 리스트를 클래스의 필드에 할당할 수 있도록 생성자를 만들어주었고 그 후 처음엔 arthropod 객체 자체를 매칭시켜 성공하면 arthropodtype을 반환할 수 있게 해주었고 실패시 arthropod 객체의 필드를 하나씩 매칭시킬 수 있도록 해주었다. 매칭 성공시 동일하게 arthropodtype을 반환받을 수 있게 해주었고 일치하는게 없을시 UNKNOWN 타입으로 반환되게 했다.

ArthropodClassifier 클래스)



```java
public class ArthropodClassifier {
    private ArthropodType arthropodType;
    private Arthropod arthropod;

    public ArthropodClassifier(ArthropodType arthropodType, Arthropod arthropod) {
        this.arthropodType = arthropodType;
        this.arthropod = arthropod;
    }

    // 첫 번째 matches 메서드: Arthropod 객체를 매개변수로 받음
    public boolean matches(Arthropod arthropod) {
        return this.arthropod.equals(arthropod);
    }

    // 두 번째 matches 메서드: 다양한 타입의 매개변수를 받음
    public boolean matches(String arthropodname, int bodyRegions, int pairsOfAntennae, RespirationType respiration,
                           MetamorphosisType metamorphosis, int pairsOfWing, int numberOfLegs, String distinction) {
        return  this.arthropod.getArthropodname() == arthropodname && // your code
                this.arthropod.getBodyRegions() == bodyRegions &&
                this.arthropod.getPairsOfAntennae() == pairsOfAntennae &&
                this.arthropod.getRespiration() == respiration &&
                this.arthropod.getMetamorphosis() == metamorphosis &&
                this.arthropod.getPairsOfWing() == pairsOfWing &&
                this.arthropod.getNumberOfLegs() == numberOfLegs &&
                this.arthropod.getDistinction().equals(distinction);
    }

    public ArthropodType getArthropodType() {
        return arthropodType;
    }

    public Arthropod getArthropod() {
        return arthropod;
    }

    @Override
    public String toString() {
        return arthropodType.name();
    }
```

arthropodclassifier 클래스에서는 arthropod 객체를 매개변수로 받도록 matchs 메소드를 선언해주었고 그 다음은 다양한 타입의 매개변수를 받을 수 있도록 두 번째 matches 메소드를 선언해주었다.

ArthropodFactory 클래스)

```java
public class ArthropodFactory {

    public static Arthropod create(ArthropodType type){ // Arthropod 객체를 생성하는 정적 메소드. ArthropodType에 따라 적절한 Arthropod를 반환
        switch(type) { // ArthropodType에 따른 객체를 생성하여 반환하는 switch 문
            case ARACHNIDA :
            return new Arthropod.ArthropodBuilder()
            .setArthropodname(arthropodname:"Spiders") // your code
            .setBodyRegions(bodyRegions:2)
            .setPairsOfAntennae(pairsOfAntennae:0)
            .setRespiration(RespirationType.BOOK_LUNGS)
            .setMetamorphosis(MetamorphosisType.NONE)
            .setPairsOfWing(pairsOfWing:0)
            .setNumberOfLegs(numberOfLegs:8)
            .setDistinction(distinction:"")
            .build();

            case CHILOPODA :
            return new Arthropod.ArthropodBuilder()
            .setArthropodname(arthropodname:"Centipedes") // your code
            .setBodyRegions(bodyRegions:2)
            .setPairsOfAntennae(pairsOfAntennae:1)
            .setRespiration(RespirationType.TRACHEAL)
            .setMetamorphosis(MetamorphosisType.NONE)
            .setPairsOfWing(pairsOfWing:0)
            .setNumberOfLegs(numberOfLegs:30)
            .setDistinction(distinction:"1 pairs per segment")
            .build();

            case DIPLOPODA :
            return new Arthropod.ArthropodBuilder()
            .setArthropodname(arthropodname:"Milipedes") // your code
            .setBodyRegions(bodyRegions:2)
            .setPairsOfAntennae(pairsOfAntennae:1)
            .setRespiration(RespirationType.TRACHEAL)
            .setMetamorphosis(MetamorphosisType.NONE)
            .setPairsOfWing(pairsOfWing:0)
            .setNumberOfLegs(numberOfLegs:100)
            .setDistinction(distinction:"2 pair per segment")
            .build();
                case CRUSTACEA :
                return new Arthropod.ArthropodBuilder()
                .setArthropodname(arthropodname:"Crabs, Lobsters, etc.") // your code
                .setBodyRegions(bodyRegions:2)
                .setPairsOfAntennae(pairsOfAntennae:2)
                .setRespiration(RespirationType.GILLS)
                .setMetamorphosis(MetamorphosisType.VARIABLE)
                .setPairsOfWing(pairsOfWing:0)
                .setNumberOfLegs(numberOfLegs:10)
                .setDistinction(distinction:"")
                .build();

                case ODONATA :
                return new Arthropod.ArthropodBuilder()
                .setArthropodname(arthropodname:"Dragonflies, Damselflies") // your code
                .setBodyRegions(bodyRegions:3)
                .setPairsOfAntennae(pairsOfAntennae:1)
                .setRespiration(RespirationType.TRACHEAL)
                .setMetamorphosis(MetamorphosisType.INCOMPLETE)
                .setPairsOfWing(pairsOfWing:2)
                .setNumberOfLegs(numberOfLegs:6)
                .setDistinction(distinction:"wings membranous")
                .build();

                case ORTHOPTERA :
                return new Arthropod.ArthropodBuilder()
                .setArthropodname(arthropodname:"Grasshoppers, Crickets") // your code
                .setBodyRegions(bodyRegions:3)
                .setPairsOfAntennae(pairsOfAntennae:1)
                .setRespiration(RespirationType.TRACHEAL)
                .setMetamorphosis(MetamorphosisType.INCOMPLETE)
                .setPairsOfWing(pairsOfWing:2)
                .setNumberOfLegs(numberOfLegs:6)
                .setDistinction(distinction:"hind legs enlarged")
                .build();
```

```
77              case  DIPTERA :
78              return new Arthropod.ArthropodBuilder()
79              .setArthropodname(arthropodname:"Flies") // your code
80              .setBodyRegions(bodyRegions:3)
81              .setPairsOfAntennae(pairsOfAntennae:1)
82              .setRespiration(RespirationType.TRACHEAL)
83              .setMetamorphosis(MetamorphosisType.COMPLETE)
84              .setPairsOfWing(pairsOfWing:1)
85              .setNumberOfLegs(numberOfLegs:6)
86              .setDistinction(distinction:"")
87              .build();
88
89              case COLEOPTERA :
90              return new Arthropod.ArthropodBuilder()
91              .setArthropodname(arthropodname:"Beetle") // your code
92              .setBodyRegions(bodyRegions:3)
93              .setPairsOfAntennae(pairsOfAntennae:1)
94              .setRespiration(RespirationType.TRACHEAL)
95              .setMetamorphosis(MetamorphosisType.COMPLETE)
96              .setPairsOfWing(pairsOfWing:2)
97              .setNumberOfLegs(numberOfLegs:6)
98              .setDistinction(distinction:"hard exoskeleton and elytra(wing covers)")
99              .build();
100
101             case LEPIDOPTERA :
102             return new Arthropod.ArthropodBuilder()
103             .setArthropodname(arthropodname:"Butterflies, Moths") // your code
104             .setBodyRegions(bodyRegions:3)
105             .setPairsOfAntennae(pairsOfAntennae:1)
106             .setRespiration(RespirationType.TRACHEAL)
107             .setMetamorphosis(MetamorphosisType.COMPLETE)
108             .setPairsOfWing(pairsOfWing:2)
109             .setNumberOfLegs(numberOfLegs:6)
110             .setDistinction(distinction:"wings scaly")
111             .build();
112
113               case HYMENOPTERA :
114               return new Arthropod.ArthropodBuilder()
115               .setArthropodname(arthropodname:"Bees, Wasps, Ants") // your code
116               .setBodyRegions(bodyRegions:3)
117               .setPairsOfAntennae(pairsOfAntennae:1)
118               .setRespiration(RespirationType.TRACHEAL)
119               .setMetamorphosis(MetamorphosisType.COMPLETE)
120               .setPairsOfWing(pairsOfWing:2)
121               .setNumberOfLegs(numberOfLegs:6)
122               .setDistinction(distinction:"wings membranous")
123               .build();
124
125               default :  // UNKNOWN 타입에 대한 기본 처리
126               return new Arthropod.ArthropodBuilder()
127               .setArthropodname(arthropodname:"UNKNOWN") // your code
128               .setBodyRegions(bodyRegions:0)
129               .setPairsOfAntennae(pairsOfAntennae:0)
130               .setRespiration(RespirationType.UNKNOWN)
131               .setMetamorphosis(MetamorphosisType.UNKNOWN)
132               .setPairsOfWing(pairsOfWing:0)
133               .setNumberOfLegs(numberOfLegs:0)
134               .setDistinction(distinction:"UNKNOWN")
135               .build();
136           }
137       }
138   }
139
```

arthropodfactory 클래스에서는 arthropod 객체를 생성하는 정적 메소드, arthropodtype에 따라 적절한 arthropod를 반환하는 create메소드를 선언해 주었고 switch문을 통해 arthropodtype에 따른 객체를 생성하여 반환해주도록 해주었다.

ArthropodImporter 클래스)

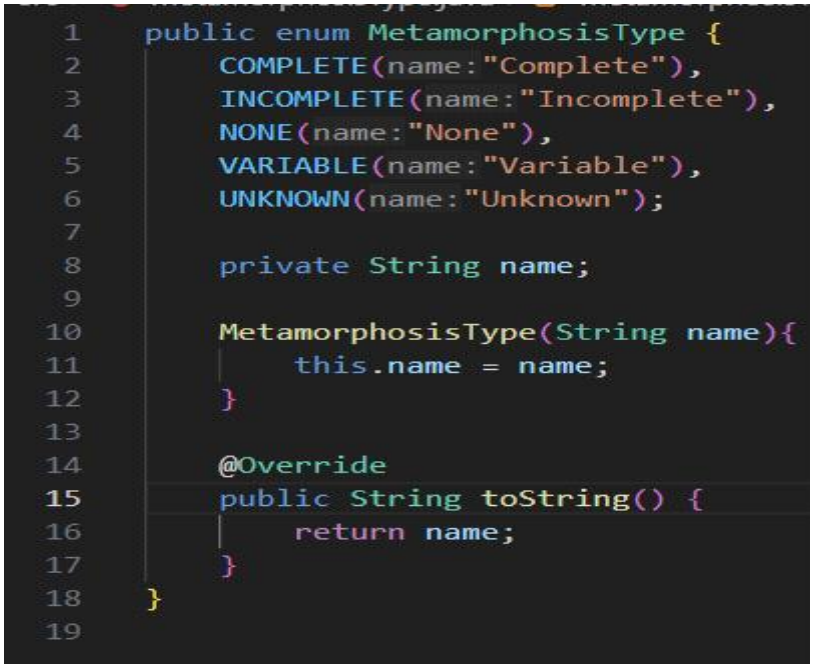
```java
import java.util.List;
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ArthropodImporter {

    // CSV 파일을 읽어 List<ArthropodClassifier>로 변환하는 메소드
    public static List<ArthropodClassifier> loadCSV(String filename) {
        // ArthropodClassifier 객체들을 저장할 리스트
        List<ArthropodClassifier> arthropods = new ArrayList<>();
        // ArthropodFactory 객체 생성
        ArthropodFactory arthropodFactory = new ArthropodFactory();

        // 파일을 읽기 위해 BufferedReader 사용
        try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
            String line;

            // CSV 파일의 각 줄을 읽음 (11개의 줄을 읽음)
            for(int i = 0; i < 11; i++) {
                // 한 줄을 읽어옴
                line = br.readLine();

                // 쉼표(,)를 기준으로 문자열을 나누어 배열로 변환
                String[] values = line.split(regex:",");

                // ArthropodType 열거형 값을 CSV에서 읽은 첫 번째 값으로 설정
                ArthropodType type = ArthropodType.valueOf(values[0].toUpperCase());

                // 임시 Arthropod 객체 생성
                Arthropod tmpArthropod;

                // ArthropodFactory를 사용하여 Arthropod 객체 생성
                tmpArthropod = arthropodFactory.create(type);

                // 생성된 Arthropod 객체를 ArthropodClassifier로 감싸서 리스트에 추가
                arthropods.add(new ArthropodClassifier(type, tmpArthropod));
            }
        } catch (IOException e) {
            // 파일 읽기 중 발생한 예외 처리
            e.printStackTrace();
        }

        // ArthropodClassifier 리스트 반환
        return arthropods;
    }
}
```

arthropodimporter 클래스에서는 csv파일을 읽어 list<arthropodclassifier>로 변환하는 메소드인 loadCSV 메소드를 선언해주었고 그 안에 객체들을 저장할 리스트를 선언해주고 파일을 읽기 위해 bufferedreader를 사용해주었다.

ArthropodType ENUM)

```java
public enum ArthropodType {
    ARACHNIDA(name:"Arachnida"),
    CHILOPODA(name:"Chilopoda"),
    DIPLOPODA(name:"Diplopoda"),
    CRUSTACEA(name:"Crustacea"),
    ORTHOPTERA(name:"Orthoptera"),
    ODONATA(name:"Odonata"),
    DIPTERA(name:"Diptera"),
    COLEOPTERA(name:"Coleoptera"),
    LEPIDOPTERA(name:"Lepidoptera"),
    HYMENOPTERA(name:"Hymenoptera"),
    UNKNOWN(name:"Unknown");

    private String name;

    ArthropodType(String name){
        this.name = name;
    }


    public String toString(){
        return name;
    }

}
```

MetamorphosisType ENUM)



```
public enum MetamorphosisType {
    COMPLETE(name:"Complete"),
    INCOMPLETE(name:"Incomplete"),
    NONE(name:"None"),
    VARIABLE(name:"Variable"),
    UNKNOWN(name:"Unknown");

    private String name;

    MetamorphosisType(String name){
        this.name = name;
    }

    @Override
    public String toString() {
        return name;
    }
}
```

RespirationType ENUM)

```
public enum RespirationType {
    TRACHEAL(name:"Tracheal"),
    BOOK_LUNGS(name:"Book Lungs"),
    GILLS(name:"Gills"),
    UNKNOWN(name:"Unknown");

    private String name;

    RespirationType(String name){
        this.name = name;
    }

    @Override
    public String toString() {
    return name;
        }
}
```

App 클래스)

```java
/* HW5
자바프로그래밍2_1분반
2024/10/23
32211792
박재홍 */

import java.util.*;
import java.util.stream.Collectors;

public class App {

    Run | Debug
    public static void main(String[] args) {

        // arthropods.csv 파일에서 곤충 데이터를 읽어와 ArthropodClassifier 리스트를 생성
        List<ArthropodClassifier> classifiers = ArthropodImporter.loadCSV(filename:"C:/Users/박재홍/source/java24/HW5/src/arthropods.csv");

        // ArthropodClassifier 리스트를 통해 ArthropodClassification 인스턴스를 생성
        ArthropodClassification classification = new ArthropodClassification(classifiers);

        // ArthropodBuilder를 사용하여 'Beetle'이라는 곤충 객체를 생성
        Arthropod beetle = new Arthropod.ArthropodBuilder()
        .setArthropodname(arthropodname:"Beetle")  // 곤충의 이름 설정 (Your Code)
        .setBodyRegions(bodyRegions:3)
        .setPairsOfAntennae(pairsOfAntennae:1)
        .setRespiration(RespirationType.TRACHEAL)
        .setMetamorphosis(MetamorphosisType.COMPLETE)
        .setPairsOfWing(pairsOfWing:2)
        .setNumberOfLegs(numberOfLegs:6)
        .setDistinction(distinction:"hard exoskeleton and elytra(wing covers)")
        .build();

        System.out.print(beetle);
        System.out.println(" => Arthropod Type: " + classification.classify(beetle));

        // ArthropodFactory를 사용하여 'Crustacea' 객체를 생성
        Arthropod crab = ArthropodFactory.create(ArthropodType.CRUSTACEA);
        System.out.println(crab);
        System.out.println(" => Arthropod Type: " + classification.classify(crab));
        System.out.println();
        System.out.println();

        // classifiers 리스트에 있는 ArthropodClassifier의 Arthropod 객체를 Arthropod 리스트로 변환
        List<Arthropod> arthropods = classifiers.stream()
            .map(classifier -> classifier.getArthropod())
            .collect(Collectors.toList());

        // arthropods 리스트에 있는 각 곤충 객체를 출력하고, 그들의 ArthropodType을 분류하여 출력
        for (Arthropod arthropod : arthropods) {
            System.out.println(arthropod);
            System.out.println(" => Arthropod Type: " + classification.classify(arthropod));
        }
    }
}

```

arthropod.csv 파일에서 곤충 데이터를 읽어와 arthropodclassifier 리스트를 생성해주었고 그 리스트들을 통해 arthropodclassification 인스턴스를 생성해주었다. 그리고 arthropodbuilder를 통해 beetle 이라는 곤충 객체를 생성해 출력해주었고 arthropodfactory를 통해 crustacea 라는 객체를 생성해주었다. 그리고 classifiers 리스트에 있는 arthropodclassifier의 arthropod 객체를 arthropod 리스트로 변환시켜주고 arthropods 리스트에 잇는 각 곤충 객체를 출력하고 그들의 arthropodtype를 분류하여 출력하도록 해주었다.

Your Code)

```
private String arthropodname;  // 곤충의 이름을 저장하는 필드 (Your Code)
```

```
this.arthropodname = builder.arthropodname; // 곤충의 이름을 builder에서 가져옴 (Your Code)
```

```
public String getArthropodname(){  // 곤충의 이름을 반환하는 메소드 (Your Code)
    return arthropodname;
}
```

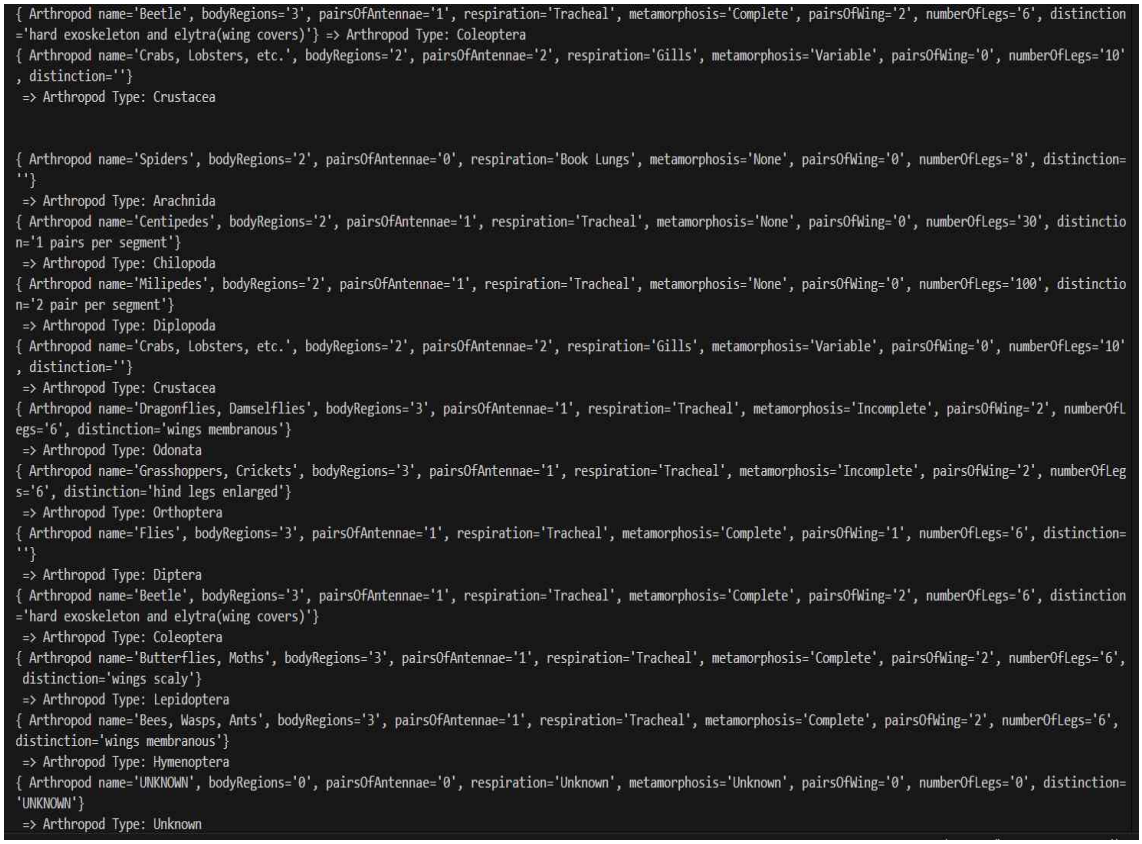
```
@Override
public String toString() {
    return "{" +
        " Arthropod name='" + getArthropodname() + "'" +  // 곤충의 이름 출력 (Your Code)
```

```
private String arthropodname;  // 곤충 이름 (Your Code)

// 곤충 이름을 설정하는 메소드
public ArthropodBuilder setArthropodname(String arthropodname){
    this.arthropodname = arthropodname;
    return this;
}
// 두 번째 matches 메서드: 다양한 타입의 매개변수를 받음
public boolean matches(String arthropodname, int bodyRegions, int pairsOfAntennae, RespirationType respiration,
                       MetamorphosisType metamorphosis, int pairsOfWing, int numberOfLegs, String distinction) {
   return  this.arthropod.getArthropodname() == arthropodname && // your code
```

```
public static Arthropod create(ArthropodType type){ // Arthropod
    switch(type) { // ArthropodType에 따른 객체를 생성하여 반환하
        case ARACHNIDA :
        return new Arthropod.ArthropodBuilder()
        .setArthropodname(arthropodname:"Spiders") // your code
        .setBodyRegions(bodyRegions:2)
        .setPairsOfAntennae(pairsOfAntennae:0)
        .setRespiration(RespirationType.BOOK_LUNGS)
        .setMetamorphosis(MetamorphosisType.NONE)
        .setPairsOfWing(pairsOfWing:0)
        .setNumberOfLegs(numberOfLegs:8)
        .setDistinction(distinction:"")
        .build();
```

중간중간에 보면 코드안에 Arthropodname이라는 것이 있는데 곤총의 종류에 따른 곤충의 이름들을 추가적으로 넣어주어서 출력시 곤충의 이름과 그에 대해 정보가 뜨도록 해주었다.

실행결과)



```
{ Arthropod name='Beetle', bodyRegions='3', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='Complete', pairsOfWing='2', numberOfLegs='6', distinction
='hard exoskeleton and elytra(wing covers)'} => Arthropod Type: Coleoptera
{ Arthropod name='Crabs, Lobsters, etc.', bodyRegions='2', pairsOfAntennae='2', respiration='Gills', metamorphosis='Variable', pairsOfWing='0', numberOfLegs='10'
, distinction=''}
 => Arthropod Type: Crustacea


{ Arthropod name='Spiders', bodyRegions='2', pairsOfAntennae='0', respiration='Book Lungs', metamorphosis='None', pairsOfWing='0', numberOfLegs='8', distinction=
''}
 => Arthropod Type: Arachnida
{ Arthropod name='Centipedes', bodyRegions='2', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='None', pairsOfWing='0', numberOfLegs='30', distinctio
n='1 pairs per segment'}
 => Arthropod Type: Chilopoda
{ Arthropod name='Milipedes', bodyRegions='2', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='None', pairsOfWing='0', numberOfLegs='100', distinctio
n='2 pair per segment'}
 => Arthropod Type: Diplopoda
{ Arthropod name='Crabs, Lobsters, etc.', bodyRegions='2', pairsOfAntennae='2', respiration='Gills', metamorphosis='Variable', pairsOfWing='0', numberOfLegs='10'
, distinction=''}
 => Arthropod Type: Crustacea
{ Arthropod name='Dragonflies, Damselflies', bodyRegions='3', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='Incomplete', pairsOfWing='2', numberOfL
egs='6', distinction='wings membranous'}
 => Arthropod Type: Odonata
{ Arthropod name='Grasshoppers, Crickets', bodyRegions='3', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='Incomplete', pairsOfWing='2', numberOfLeg
s='6', distinction='hind legs enlarged'}
 => Arthropod Type: Orthoptera
{ Arthropod name='Flies', bodyRegions='3', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='Complete', pairsOfWing='1', numberOfLegs='6', distinction=
''}
 => Arthropod Type: Diptera
{ Arthropod name='Beetle', bodyRegions='3', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='Complete', pairsOfWing='2', numberOfLegs='6', distinction
='hard exoskeleton and elytra(wing covers)'}
 => Arthropod Type: Coleoptera
{ Arthropod name='Butterflies, Moths', bodyRegions='3', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='Complete', pairsOfWing='2', numberOfLegs='6',
 distinction='wings scaly'}
 => Arthropod Type: Lepidoptera
{ Arthropod name='Bees, Wasps, Ants', bodyRegions='3', pairsOfAntennae='1', respiration='Tracheal', metamorphosis='Complete', pairsOfWing='2', numberOfLegs='6',
distinction='wings membranous'}
 => Arthropod Type: Hymenoptera
{ Arthropod name='UNKNOWN', bodyRegions='0', pairsOfAntennae='0', respiration='Unknown', metamorphosis='Unknown', pairsOfWing='0', numberOfLegs='0', distinction=
'UNKNOWN'}
 => Arthropod Type: Unknown
```