KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Visitor- 데이터 구조를 돌아다니면서 처리한다

[KB] IT's Your Life

KB 국민은행

Visitor 패턴

- 데이터 구조와 처리를 분리
- 데이터 구조 안을 돌아다니는 주체인 '방문자'를 나타내는 클래스를 준비하고 그 클래스에 처리를 맡김
- 새로운 처리를 추가하고 싶을 때는 새로운 '방문자'를 만들면 됨
- 데이터 구조 쪽에서는 문을 두드리는 '방문자'를 받아 줌

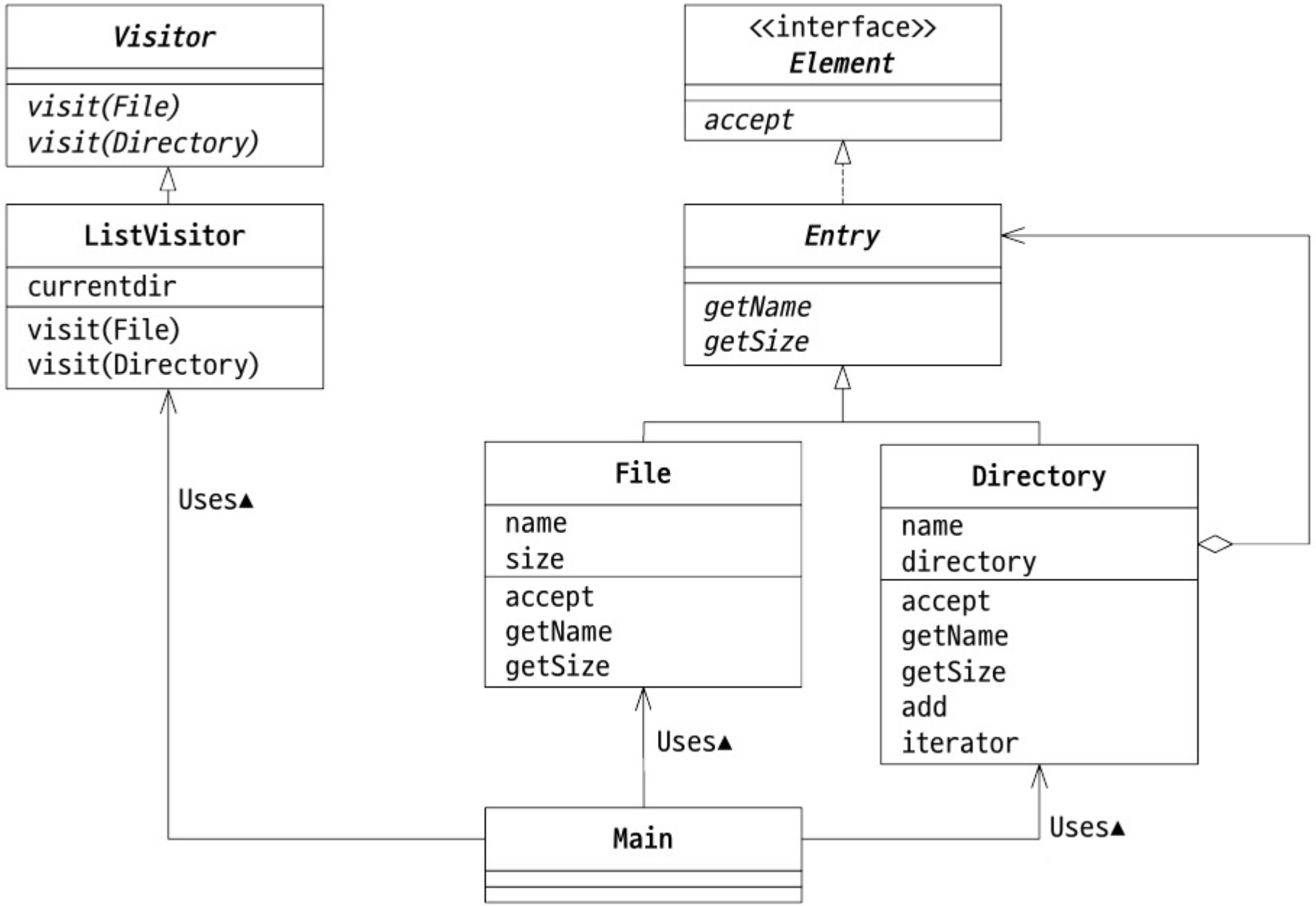
예제 프로그램

- 방문자를 돌아다니는 데이터구조로 Composite 패턴에서 등장한 파일과 디렉터리 사용
- 파일과 디렉터리로 구성된 데이터 구조 안을 방문자가 돌아다니면 파일 목록을 표시

이름	설명
Visitor	파일이나 디렉터리를 방문하는 방문자를 나타내는 추상 클래스
Element	Visitor 클래스의 인스턴스를 받아들이는 데이터 구조를 나타내는 인터페이스
ListVisitor	Visitor 클래스의 하위 클래스로 파일이나 디렉터리 목록을 표시하는 클래스
Entry	File과 Directory의 상위 클래스가 되는 추상 클래스 (Acceptor 인터페이스 구현)
File	파일을 나타내는 클래스
Directory	디렉터리를 나타내는 클래스
Main	동작 테스트용 클래스

예제 프로그램

- 예제 프로그램 클래스 다이어그램



Element.java

```
public interface Element {
    public abstract void accept(Visitor v);
}
```

Entry.java

```
public abstract class Entry implements Element{
    public abstract String getName();   // 이름을 얻는다.
    public abstract int getSize();  // 크기를 얻는다.

    // 문자열 표현
    @Override
    public String toString() {
        return getName() + "(" + getSize() + ")";
    }
}
```

File.java

```
public class File extends Entry{
    private String name;
    private int size;

    public File(String name, int size) {
        this.name = name;
        this.size = size;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public int getSize() {
        return size;
    }

    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }
}
```

Directory.java

```
public class Directory extends Entry implements Iterable<Entry>{
    private String name;
    private List<Entry> directory = new ArrayList<>();

    public Directory(String name) {
        this.name = name;
    }

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

Directory.java

```
    @Override
    public int getSize() {
        int size = 0;
        for(Entry entry: directory) {
            size += entry.getSize();
        }
        return size;
    }

    public Entry add(Entry entry) {
        directory.add(entry);
        return this;
    }

    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }

    @Override
    public Iterator<Entry> iterator() {
        return directory.iterator();
    }
}
```

Visitor.java

```
public abstract class Visitor {
    public abstract void visit(File file);
    public abstract void visit(Directory directory);

}
```

ListVisitor.java

```
public class ListVisitor extends Visitor{
    // 현재 주목하는 디렉터리 이름
    private String currentdir = "";

    // File 방문 시
    @Override
    public void visit(File file) {
        System.out.println(currentdir + "/" + file);
    }

    // Directory 방문 시
    @Override
    public void visit(Directory directory) {
        System.out.println(currentdir + "/" + directory);
        String savedir = currentdir;
        currentdir = currentdir + "/" + directory.getName();
        for(Entry entry: directory) {
            entry.accept(this);  // 재귀 호출!!!
        }
        currentdir = savedir;
    }
}
```

예제 프로그램

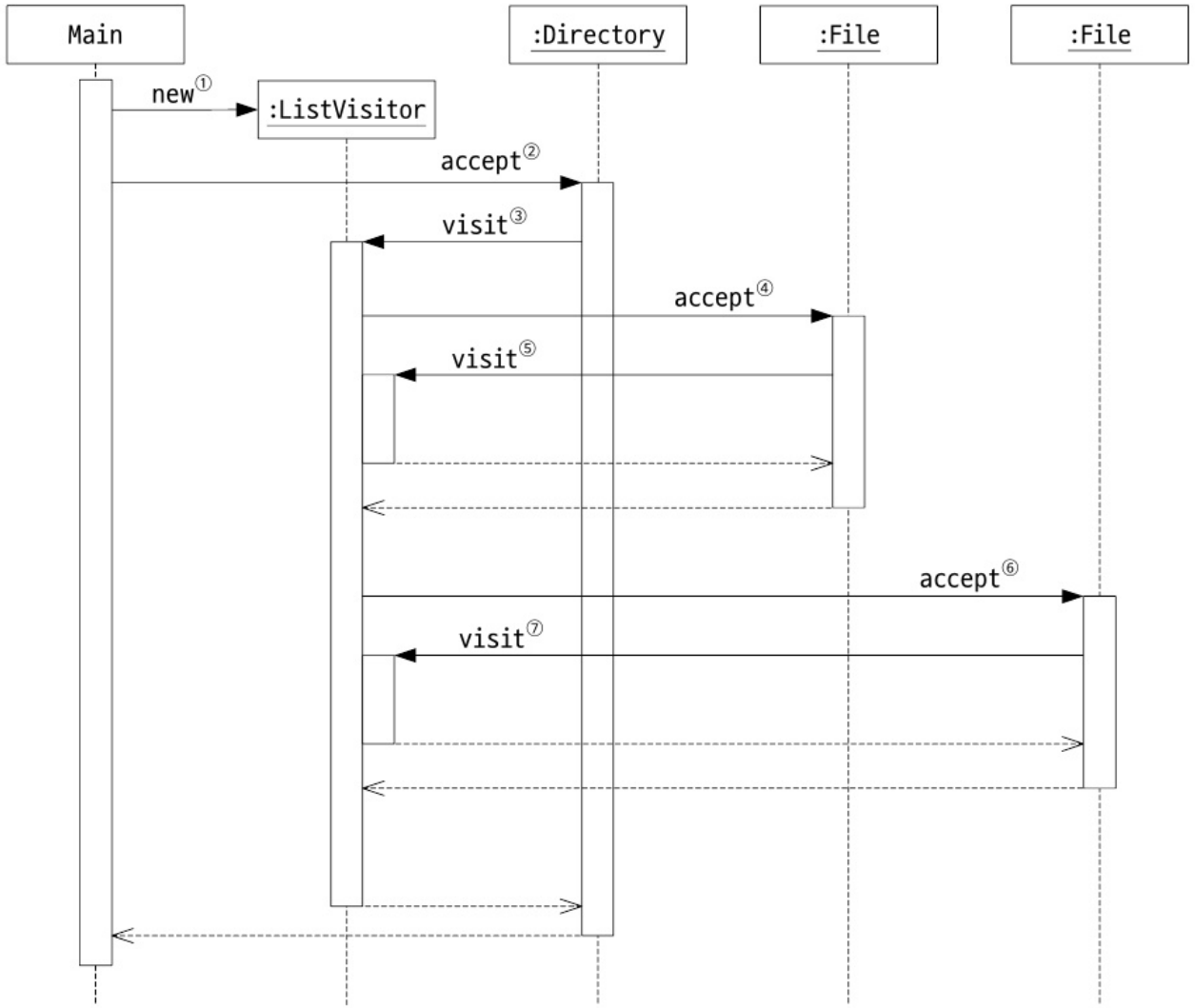
Main.java

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

        System.out.println("Making root entries...");
        Directory rootdir = new Directory("root");

        Directory bindir = new Directory("bin");
        Directory tmpdir = new Directory("tmp");
        Directory usrdir = new Directory("usr");

        rootdir.add(bindir);
        rootdir.add(tmpdir);
        rootdir.add(usrdir);

        bindir.add(new File("vi", 10000));
        bindir.add(new File("latex", 20000));

        rootdir.accept(new ListVisitor());
        System.out.println();
```

Main.java

```
        System.out.println("Making user entries...");
        Directory youngjin = new Directory("youngjin");
        Directory gildong = new Directory("gildong");
        Directory dojun = new Directory("dojun");

        usrdir.add(youngjin);
        usrdir.add(gildong);
        usrdir.add(dojun);

        youngjin.add(new File("diary.html", 100));
        youngjin.add(new File("Composite.java", 200));

        gildong.add(new File("memo.tex", 300));

        dojun.add(new File("game.doc", 400));
        dojun.add(new File("junk.mail", 500));

        rootdir.accept(new ListVisitor());
    }
}
```

- 예제 프로그램의 시퀀스 다이어그램 (하나의 디렉터리에 두 개의 파일이 있는 경우)



Visitor 패턴

Visitor 패턴의 클래스 다이어그램