REPORT

HW6

자바프로그래밍2

제출일 2023. 11.13

소속 컴퓨터공학과

학번 32183520

이름 이 주성

목표

- Command 패턴과 Strategy 패턴을 사용해 incremental find program 작성하기

Command 패턴

- 명령(Command)를 수신자(Receiver)와 호출자(Invoker) 사이에서 캡슐화함
    - 요청의 발신자와 수신자를 분리해 시스템을 느슨하게 결합
- 클라이언트는 Command 객체를 통해 다양한 작업을 수행한다.
- 요청을 대기열에 추가하거나 로깅해 실행 취소 가능한 작업을 지원한다.
- 이번 예제에서 **Command** 패턴을 적용해 사용자 입력에 따라 해당하는 **Command** 객체를 **Invoker**에 할당한 후 실행시킨다**.**

Strategy 패턴

- 클래스 또는 알고리즘을 런타임에 변경 가능도록 하는 패턴
- 적용하고 싶은 알고리즘들을 정의해두고 사용하고 싶을 때 쉽게 바꿔서 사용할 수 있다.
- 이번 예제에서는 **Finder**를 실행시간에 동적으로 정할 것이므로 **Finder**를 **Strategy** 패턴을 적용해 구현한다**.**

IFinderCommand 인터페이스

```
// Command interface
public interface IFinderCommand {
   List<PeriodicElement> execute(List<PeriodicElement> elements) throws
IOException;

   List<PeriodicElement> undo();
}
```

FinderCommand 추상 클래스

- IFinderCommand 구현
- 구체적인 Command 객체들의 undo 로직이 동일하므로 추상 클래스로 undo를 오버라이딩해 이 추상 클래스를 상속시켜 undo 관련 코드를 재사용한다.

```
// IFinderCommand를 구현하는 추상클래스
// Command 객체 모두가 동일한 undo 로직을 가지고 있으므로 추상클래스로 만들어 코드를 재사용한다.
public abstract class FinderCommand implements IFinderCommand {
   protected List<PeriodicElement> prevElements;

   // undo(): 이전의 prevElements를 리턴해준다.
   @Override
   public List<PeriodicElement> undo() {
       return prevElements;
   }
}
```

Concrete Command 객체

PhaseFinderCommand

- FinderCommand 추상 클래스 상속
- Receiver에 대한 참조를 가지고 있고 Receiver의 실제 작업 메서드인 find()를 함께 캡슐화한다.
- 입력받은 Phase로 PhaseFinder 생성

```
public class PhaseFinderCommand extends FinderCommand {
   // Receiver - Strategy Pattern 적용
   private IPeriodicElementFinder finder;

   @Override
         public   List<PeriodicElement>   execute(List<PeriodicElement> elements) throws IOException {
       // undo()를 위해 이전 상태 기록
```

```
        prevElements = elements;


        // phase를 입력받아 PhaseFinder 생성
         System.out.print("Please enter [phase] of PeriodicElement [e.g.
gas, liq, solid, artificial]: ");
        finder = new PhaseFinder(UserInput.getPhase());


        // 주어진 원소 리스트 (elements)에서 PhaseFinder를 이용해 조회 후 리턴
        return finder.find(elements);
    }
}
```

- UserInput.getPhase()

```
// 조회하고 싶은 phase 값 입력받기
public static String getPhase() {
    String input = null;

    while (true) {
        try {
            input = br.readLine();

            if (!input.isEmpty()) {
                for (Phase p : Phase.values()) {
                    if (p.toString().equals(input)) {
                        return input;
                    }
                }
                System.out.print("gas, liq, solid, artificial 중 하나를
입력해주세요: ");
            } else {
                System.out.print("다시 입력해주세요: ");
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

WeightFinderCommand

- FinderCommand 추상 클래스 상속
- Receiver에 대한 참조를 가지고 있고 Receiver의 실제 작업 메서드인 find()를 함께 캡슐화한다.
- 입력받은 질량 범위로 WeightFinder 생성

```
public class WeightFinderCommand extends FinderCommand {
    // Receiver - Strategy Pattern 적용
    private IPeriodicElementFinder finder;

    @Override
    public List<PeriodicElement> execute(List<PeriodicElement> elements) throws
IOException {
        // undo()를 위해 이전 상태 기록
        prevElements = elements;

        // 질량 범위를 입력받아 WeightFinder 생성
        System.out.print("조회하고 싶은 원소의 질량 범위를 물결('~')을 기준으로
입력하세요. : ");
        finder = new WeightFinder(UserInput.getDoubleArray());

        // 주어진 원소 리스트(elements)에서 WeightFinder를 이용해 조회 후 리턴
        return finder.find(elements);
    }
}
```

- UserInput.getDoubleArray()

```
public static double[] getDoubleArray() {
    double[] numbers;

    try {
        // 공백을 기준으로 입력받아 int[]로 변환
        numbers = Arrays.stream(br.readLine().split("~"))
                .mapToDouble(Double::parseDouble)
                .toArray();

        return numbers;
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

나머지 구체 **Command** 객체들도 동일한 구조로 구현된다.

Receiver - Strategy Pattern 적용

IPeriodicElemnetFinder

- 원소 기호를 검색하기 위해 만든 인터페이스

- 어떤 속성으로 검색할지 Strategy Pattern으로 런타임에 결정하기 위해 인터페이스로 만듬

- 이 인터페이스를 구현한 xxxFinder가 오버라이딩한 find() 메소드로 검색

```
// Strategy Pattern
public interface IPeriodicElementFinder {
    // PeriodicElement 리스트를 찾는 find 메서드
    // 나중에 구체 클래스에서 오버라이딩해 사용
    List<PeriodicElement> find(List<PeriodicElement> elements);
}
```

6가지 **Finder**

- 6가지의 다른 기준으로 IPeriodicElementFinder를 구현한 Finder 클래스

NumberFinder

- 1~118 사이의 number를 배열 형태로 받아 일치하는 원소를 foundElements

리스트에 추가한다.

```
// 이름을 기준으로 검색하는 전략
public class NameFinder implements IPeriodicElementFinder {
    private String[] names;

    public NameFinder(String[] names) {
        this.names = names;
    }

    @Override
    public List<PeriodicElement> find(List<PeriodicElement> elements){
        List<PeriodicElement> foundElements = new ArrayList<>();

        // 일치하는 번호의 원소를 찾으면 foundElements에 추가
        for (PeriodicElement element : elements) {
                                    if    (Arrays.stream(names).anyMatch(s    ->
s.equals(element.getName()))) {
                foundElements.add(element);
            }
        }

        return foundElements;
    }
}
```

GroupFinder

- 1~18 혹은 0을 입력받는다.
- 0인 경우는 null인 group을 찾는다.

```
public class GroupFinder implements IPeriodicElementFinder {
    private int group;
```

```java
    public GroupFinder(int group) {
        this.group = group;
    }

    @Override
    public List<PeriodicElement> find(List<PeriodicElement> elements){
        List<PeriodicElement> foundElements = new ArrayList<>();

        // null인 경우
        if (group == 0) {
            for (PeriodicElement element : elements) {
                if (element.getGroup() >= 1 && element.getGroup() <= 18) {
                    continue;
                } else {
                    foundElements.add(element);
                }
            }
        } else {
            for (PeriodicElement element : elements) {
                if (element.getGroup() == group) {
                    foundElements.add(element);
                }
            }
        }

        return foundElements;
    }
}
```

나머지 **finder**들도 같은 구조로 구현했다**.**

FinderInvoker

- 요청을 받아들이고 명령 인터페이스를 실행
- 명령이 어떻게 작동하는지에 대한 구체적인 내용 모름
- 단지 명령 인터페이스를 호출해 요청을 실행할 뿐
- setCommand 메서드로 Command 객체를 할당하고 execute 메서드로 해당 커맨드 객체를 실행시킨다.
- stack을 통해 이전 검색 기록을 관리한다. -> undo()를 위해
    - push()로 현재 명령 스택에 저장
    - 직전의 명령을 pop()으로 가져와 undo tlfgod
    - 직전의 명령이 없을 경우 빈 리스트 반환

```
// Invoker
public class FinderInvoker {
    // 설정할 Command 객체를 저장할 필드
    private IFinderCommand command;

    // Command 객체 기록을 가지고 있는 스택
    private final Stack<IFinderCommand> commandHistory = new Stack<>();

    // 실행할 Command 객체 설정
    public void setCommand(IFinderCommand command) {
        this.command = command;
    }

    // 설정한 Command 객체 실행
    public List<PeriodicElement> execute(List<PeriodicElement> elements)
throws IOException {
        // 현재 Command 객체를 실행
        List<PeriodicElement> foundElements = command.execute(elements);
        // 현재 Command 객체를 Command 객체 기록을 가진 스택에 추가
        commandHistory.push(command);

        // 실행 결과 반환
        return foundElements;
```

```
    }

    // 이전 Command를 취소하고 그 때의 결과를 반환
    public List<PeriodicElement> undo() {
        // Command 명령 기록이 있을 경우
        if (!commandHistory.isEmpty()) {
            // 스택에서 직전 명령어 가져오기
            IFinderCommand lastCommand = commandHistory.pop();
            // 직전 명령어의 undo 실행
            return lastCommand.undo();
        } else {    // Commamnd 스택이 비었을 경우, 즉 이전에 실행된 명령어가 없을
경우
            // 빈 리스트 반환
            return new ArrayList<>();
        }
    }
}
```

Main

1. csv에서 Periodic 리스트 추출
2. Invoker 생성
3. Command 객체 생성해서 FinderCommandDatabase에 저장
4. 원하는 Command 입력받기
5. Invoker에 해당 Command 객체 FinderCommandDatabase에서 불러와서 할당
6. invoker를 통해 execute or undo

```
public class Main {
    public static List<PeriodicElement> deepCopy(List<PeriodicElement>
list) {
        return new ArrayList<>(list);
    }

    public static void main(String[] args) throws IOException {
        // PeriodicElements.csv 파일을 load해서 PeriodicElement 리스트로 저장
        List<PeriodicElement> peList =
PeriodicElementImporter.loadCSV("PeriodicElements.csv");

        // invoker class
        FinderInvoker invoker = new FinderInvoker();
```

```
        // incremental finder
        FinderCommandDatabase database = new FinderCommandDatabase();
        database.addCommand("phase", new PhaseFinderCommand());
        database.addCommand("number", new NumberFinderCommand());
        database.addCommand("group", new GroupFinderCommand());
        database.addCommand("name", new NameFinderCommand());
        database.addCommand("period", new PeriodFinderCommand());
        database.addCommand("weight", new WeightFinderCommand());
        database.addCommand("symbol", new SymbolFinderCommand());

        // create foundList
        List<PeriodicElement> foundList = deepCopy(peList);

        do {
            // 원하는 Command 입력 받기 - 유효한 Coammand인지 검증까지 해줌
            System.out.print("Please enter command [e.g. number | name |
symbol | weight | period | group | phase | undo]: ");
            String commandName = UserInput.getCommandString();

            // 입력받은 명령이 "undo"일 경우
            if (commandName.equalsIgnoreCase("undo")) {
                // invoker의 undo 실행 -> 커맨드 패턴
                // 이전 명령어 스택에서 꺼내 undo를 실행한다.
                foundList = invoker.undo();

                // undo()의 결과로 빈 리스트가 반환되었다면 다시 deepCopy
                if (foundList == null || foundList.isEmpty()) {
                    foundList = deepCopy(peList);
                }

            } else {
                // invoker에 command 객체 설정
                invoker.setCommand(database.getCommand(commandName));

                // invoker의 execute()를 실행해 설정한 command 객체의
execute()를 실행시켜준다. -> 커맨드 패턴
                foundList = invoker.execute(foundList);
            }

            // 출력
            foundList.forEach(System.out::println);

        } while (!UserInput.getExitKey()); // exit 입력받으면 종료
    }
}
```

실행결과

```
Please enter command [e.g. number | name | symbol | weight | period | group | phase | undo]: group
조회하고 싶은 원소의 그룹을 입력하세요. (0~18 사이의 값이어야 합니다. 0의 경우 null): 18
PeriodicElement [number=2, symbol=He, name=Helium, weight=4.002, period=1, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=10, symbol=Ne, name=Neon, weight=20.18, period=2, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=18, symbol=Ar, name=Argon, weight=39.948, period=3, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=36, symbol=Kr, name=Krypton, weight=83.798, period=4, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=54, symbol=Xe, name=Xenon, weight=131.293, period=5, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=86, symbol=Rn, name=Radon, weight=222.0, period=6, group=18, phase=gas, type=Alkali Metal]
PeriodicElement [number=118, symbol=Og, name=Oganesson, weight=294.0, period=7, group=18, phase=artificial, type=Noble Gas]
계속하고 싶다면 Enter를, 종료하고 싶다면 'exit'을 입력해주세요.
```

```
Please enter command [e.g. number | name | symbol | weight | period | group | phase | undo]: phase
Please enter [phase] of PeriodicElement [e.g. gas, liq, solid, artificial]: gas
PeriodicElement [number=2, symbol=He, name=Helium, weight=4.002, period=1, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=10, symbol=Ne, name=Neon, weight=20.18, period=2, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=18, symbol=Ar, name=Argon, weight=39.948, period=3, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=36, symbol=Kr, name=Krypton, weight=83.798, period=4, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=54, symbol=Xe, name=Xenon, weight=131.293, period=5, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=86, symbol=Rn, name=Radon, weight=222.0, period=6, group=18, phase=gas, type=Alkali Metal]
계속하고 싶다면 Enter를, 종료하고 싶다면 'exit'을 입력해주세요.
```

```
Please enter command [e.g. number | name | symbol | weight | period | group | phase | undo]: weight
조회하고 싶은 원소의 질량 범위를 물결('~')을 기준으로 입력하세요. : 1~50
PeriodicElement [number=2, symbol=He, name=Helium, weight=4.002, period=1, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=10, symbol=Ne, name=Neon, weight=20.18, period=2, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=18, symbol=Ar, name=Argon, weight=39.948, period=3, group=18, phase=gas, type=Noble Gas]
계속하고 싶다면 Enter를, 종료하고 싶다면 'exit'을 입력해주세요.
Please enter command [e.g. number | name | symbol | weight | period | group | phase | undo]: period
조회하고 싶은 원소의 주기를 입력하세요. (1~7 사이의 값이어야 합니다.): 2
PeriodicElement [number=10, symbol=Ne, name=Neon, weight=20.18, period=2, group=18, phase=gas, type=Noble Gas]
계속하고 싶다면 Enter를, 종료하고 싶다면 'exit'을 입력해주세요.
```

```
Please enter command [e.g. number | name | symbol | weight | period | group | phase | undo]: undo
PeriodicElement [number=2, symbol=He, name=Helium, weight=4.002, period=1, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=10, symbol=Ne, name=Neon, weight=20.18, period=2, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=18, symbol=Ar, name=Argon, weight=39.948, period=3, group=18, phase=gas, type=Noble Gas]
계속하고 싶다면 Enter를, 종료하고 싶다면 'exit'을 입력해주세요.
Please enter command [e.g. number | name | symbol | weight | period | group | phase | undo]: undo
PeriodicElement [number=2, symbol=He, name=Helium, weight=4.002, period=1, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=10, symbol=Ne, name=Neon, weight=20.18, period=2, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=18, symbol=Ar, name=Argon, weight=39.948, period=3, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=36, symbol=Kr, name=Krypton, weight=83.798, period=4, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=54, symbol=Xe, name=Xenon, weight=131.293, period=5, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=86, symbol=Rn, name=Radon, weight=222.0, period=6, group=18, phase=gas, type=Alkali Metal]
계속하고 싶다면 Enter를, 종료하고 싶다면 'exit'을 입력해주세요.
```

```
Please enter command [e.g. number | name | symbol | weight | period | group | phase | undo]: undo
PeriodicElement [number=2, symbol=He, name=Helium, weight=4.002, period=1, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=10, symbol=Ne, name=Neon, weight=20.18, period=2, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=18, symbol=Ar, name=Argon, weight=39.948, period=3, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=36, symbol=Kr, name=Krypton, weight=83.798, period=4, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=54, symbol=Xe, name=Xenon, weight=131.293, period=5, group=18, phase=gas, type=Noble Gas]
PeriodicElement [number=86, symbol=Rn, name=Radon, weight=222.0, period=6, group=18, phase=gas, type=Alkali Metal]
PeriodicElement [number=118, symbol=Og, name=Oganesson, weight=294.0, period=7, group=18, phase=artificial, type=Noble Gas]
계속하고 싶다면 Enter를, 종료하고 싶다면 'exit'을 입력해주세요.
```